Algorithms in Amazon SageMaker

Zohar Karnin, Amazon AI Labs

Edo Liberty, Bing Xiang, Baris Cuskon, Ramesh Nallapati, Phillip Gautier, Madhav Jha, Ran Ding, Tim Januschowski, David Selinas, Bernie Wang, Jan Gasthaus, Laurence Rouesnel, Amir Sadoughi, Piali Das, Julio Delgado Mangas, Yury Astashonok, Can Balioglu, Saswata Chakravarty, Alex Smola
1) ML Algorithms in The Cloud - New Challenges

2) SageMaker Algorithms - Architecture and Data Flow

3) Science of Streaming Algorithms – Advantages and Challenges

4) SageMaker Algorithms – Accurate, Fast, Scalable, and Easy to Use.

5) Deep dive – SageMaker K-Means
ML Algorithms in The Cloud – New Challenges
Lifecycle of a Machine Learning Project

Exploration → Training → Hosting → Exploration
Small Data - Machine Learning
Our Customers use ML at massive scale!

“Our data warehouse is 100TB and we are processing 2TB daily. We're running mostly gradient boosting (trees), LDA and K-Means clustering and collaborative filtering.”
Shahar Cizer Kobrinsky, VP Architecture

“We process 3 million ad requests a second, 100,000 features per request. That’s 250 trillion per day. Not your run of the mill Data science problem!”
Bill Simmons, CTO

“We collect 160M events daily in the ML pipeline and run training over the last 15 days and need it to complete in one hour. Effectively there's 100M features in the model”
Valentino Volonghi, CTO
Large Scale Machine Learning
Large Scale Machine Learning
Cost vs. Time

Minutes       Hours       Days       Weeks       Months

Single Machine
Cost vs. Time

- Ideal Case
- Single Machine

Cost Levels:
- $$$$$
- $$$
- $$
- $
Cost vs. Time

- Distributed, with Strong Machines
- Single Machine
- Ideal Case

Time:
- Minutes
- Hours
- Days
- Weeks
- Months

Cost:
- $$$$$
- $$$$
- $$$
- $$
- $
Cost vs. Time

- Ideal Case
- Single Machine
- Distributed, with Strong Machines
Model Selection
Incremental Training
Production Readiness

Investment vs. Data/Model Size

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
SageMaker Algorithms - Architecture and Data Flow
Streaming
Streaming

- Memory vs. Data Size
- Time/Cost vs. Data Size
Incremental Training
Incremental Training
GPU/CPU
Distributed

GPU State

GPU State

GPU State
Parameter Server – distributed (k,v) store.
Cost vs. Time

- Minutes
- Hours
- Days
- Weeks
- Months

- $$$$$
- $$$
- $$
- $$
- $
Cost vs. Time

- **Amazon SageMaker**
- **Best Alternative**
Production Readiness

Data/Model Size

Investment

Reasonable Investment Level

Unusable Data
Wasted opportunity
Production Readiness

Reasonable Investment Level

No unusable Data
No wasted opportunity

Investment

Data/Model Size
Science of Streaming Algorithms – Advantages and Challenges
Finding the exact median in a stream is impossible!
• After the seeing half the items, each one of them might still be the median.
• The algorithm must remember all of them.
• It cannot have a fixed memory footprint.
Gradient Descent

\[ f(x) = \frac{1}{n} \sum_{i=1}^{n} |x - x_i| \]

\[ m = \arg \min_{x} f(x) \]
Stochastic Gradient Descent

\[ f_i = |x_i - x| \quad \rightarrow \quad \mathbb{E}_i[f_i] = f \quad \rightarrow \quad \mathbb{E}_i[f'_i] = f' \]

\[ m_t = \begin{cases} 
m_{t-1} + \alpha / \sqrt{t} & \text{if } m_{t-1} < x_i \\
m_{t-1} - \alpha / \sqrt{t} & \text{if } m_{t-1} > x_i \end{cases} \]

Frugal Streaming for Estimating Quantiles: One (or two) memory suffices: Qiang Ma, S. Muthukrishnan, Mark Sandler
Frugal Streaming for Estimating Quantiles: One (or two) memory suffices: Qiang Ma, S. Muthukrishnan, Mark Sandler

\[ m_t = \begin{cases} 
  m_{t-1} + \frac{\alpha}{\sqrt{t}} & \text{if } m_{t-1} < x_i \\
  m_{t-1} - \frac{\alpha}{\sqrt{t}} & \text{if } m_{t-1} > x_i 
\end{cases} \]
Frugal Streaming for Estimating Quantiles: One (or two) memory suffices: Qiang Ma, S. Muthukrishnan, Mark Sandler
Median - Sampling Algorithm

Sampling Algorithm:
1) Reservoir Sample k points from the data
2) Return the median of the sample
Median – Sketching Algorithm

Sketching Algorithm
1) Too complex to explain here...
2) Optimal Quantile Approximation in Streams; Zohar Karnin, Kevin Lang, Edo Liberty
Framework for Streaming Algorithms

\[ x_1, x_2, \ldots, x_n \rightarrow \frac{1}{n} \sum_{i=1}^{n} x_i \]
Average: Single Machine

```
running_sum = 0.0
items_seen = 0
```

```python
Initialize()
```

```
for x in stream:
    running_sum += x
    items_seen += 1
```

```python
update(x)
```

```
average = running_sum / items_seen
```

```python
finalize()
```
Average: Single Machine

```python
state.initialize()
for x in stream:
    state.update(x)
return state.finalize()
```
Average: Distributed

In parallel for w in workers:
  state[w].initialize()
  for x in stream[w]:
    state[w].update(x)
Average: Distributed

In parallel for w in workers:
    state[w].initialize()
    for x in stream[w]:
        state[w].update(x)

    running_sum += state[w].running_sum
    items_seen += state[w].items_seen

    average = running_sum / items_seen

initialize()
merge(state[w])
finalize()
Average: Distributed

In parallel for w in workers:
    state[w].initialize()
    for x in stream[w]:
        state[w].update(x)

master_state.initialize()
for w in workers:
    master_state.merge(state[w])
return master_state.finalize()
Distributed Learning

**Fully mergeable**

PCA: requires $E[x_t x_t^T]$

Convex SGD

Non-convex SGD

**Not mergeable**
Mergeable
Not mergeable
Distributed SGD

local_state.initialize()

for batch in stream:
    local_state.pull()
    delta = local_state.update(batch)
    local_state.push(delta)

return local_state.finalize()
HPO, State > Model
HPO, State > Model

Logistic Regression: Tune threshold for Acc, F1, p@r, etc.
K-Means: state.finalize(k)
SageMaker Algorithms – Accurate, Fast, Scalable, and Easy to Use.
## Algorithms - Example Usage

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Function</th>
<th>Example Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Learner</td>
<td>Classification and regression, these are the most popular ML algorithms used today.</td>
<td>• Estimating click probability for online advisements (for a customer)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Directing a customer’s inbound phone call to relevant agents</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Deciding whether a login event is legitimate.</td>
</tr>
<tr>
<td>Boosted Decision Trees (XGBoost)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Factorization Machines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K-Nearest Neighbor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K-means</td>
<td>Clustering</td>
<td>• Grouping similar events/document/images together</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
<td>• Reduce Dimensionality of data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Explore main factors/trends in data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Visualization</td>
</tr>
<tr>
<td>Neural Topic Modelling</td>
<td>Topic Modeling</td>
<td>• Maps documents into distribution over topics</td>
</tr>
<tr>
<td>Spectral LDA</td>
<td></td>
<td>• Discover dominant topics in your text corpus</td>
</tr>
<tr>
<td>Blazing Text</td>
<td>Word Embedding</td>
<td>• Feature Engineering for text</td>
</tr>
<tr>
<td>DeepAR</td>
<td>Time-series Forecasting</td>
<td>• Predict the number of page views you’ll get in an hour (and the number of servers you’ll need to host them!)</td>
</tr>
<tr>
<td>Image Classification</td>
<td>Classification of Images</td>
<td>• Detect quality assurance issues in manufactured goods using images.</td>
</tr>
<tr>
<td>Sequence to Sequence</td>
<td>Learn mapping between pairs of sequences</td>
<td>• Translating text between different languages.</td>
</tr>
</tbody>
</table>
Linear Learner

Regression:
Estimate a real valued function

\[ \hat{y} = \langle x, w \rangle + b \]

Binary Classification:
Predict a 0/1 class

\[ \hat{y} = \langle x, w \rangle > t \, ? \, 0:1 \]

Multiclass Classification:
Predict one of \( k \) classes

\[ \hat{y} = \text{argmax}_i \langle x, w_i \rangle \]
Linear Learner

>8x speedup over naïve parallel training!

\[
\begin{align*}
    w_1 &= \min_w \sum_i L_1(w^T x_i, y_i) + \alpha_1 \|w\|_1 + \beta_1 \|w\|_2 \\
    \vdots & \quad \vdots \\
    w_k &= \min_w \sum_i L_k(w^T x_i, y_i) + \alpha_k \|w\|_1 + \beta_k \|w\|_2
\end{align*}
\]

Select model with best validation performance
## Linear Learner

<table>
<thead>
<tr>
<th>Regression (mean squared error)</th>
<th>SageMaker</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.02</td>
<td>1.06</td>
<td></td>
</tr>
<tr>
<td>1.09</td>
<td>1.02</td>
<td></td>
</tr>
<tr>
<td>0.332</td>
<td>0.183</td>
<td></td>
</tr>
<tr>
<td>0.086</td>
<td>0.129</td>
<td></td>
</tr>
<tr>
<td>83.3</td>
<td>84.5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classification (F1 Score)</th>
<th>SageMaker</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.980</td>
<td>0.981</td>
<td></td>
</tr>
<tr>
<td>0.870</td>
<td>0.930</td>
<td></td>
</tr>
<tr>
<td>0.997</td>
<td>0.997</td>
<td></td>
</tr>
<tr>
<td>0.978</td>
<td>0.964</td>
<td></td>
</tr>
<tr>
<td>0.914</td>
<td>0.859</td>
<td></td>
</tr>
<tr>
<td>0.470</td>
<td>0.472</td>
<td></td>
</tr>
<tr>
<td>0.903</td>
<td>0.908</td>
<td></td>
</tr>
<tr>
<td>0.508</td>
<td>0.508</td>
<td></td>
</tr>
</tbody>
</table>

### 30 GB datasets for web-spam and web-url classification

![Graph showing cost in dollars and billable time in minutes for different datasets and classification algorithms.]
Boosted Decision Trees

XGBoost is one of the most commonly used implementations of boosted decision trees in the world.

It is now available in Amazon SageMaker!
## Factorization Machines

\[
\hat{y} = w_0 + \langle w_1, x \rangle + \sum_{i,j > i} x_i x_j \langle v_i, v_j \rangle
\]

<table>
<thead>
<tr>
<th></th>
<th>Log_loss</th>
<th>F1 Score</th>
<th>Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>SageMaker (single pass)</td>
<td>0.494</td>
<td>0.277</td>
<td>820</td>
</tr>
<tr>
<td>Other (10 Iter)</td>
<td>0.516</td>
<td>0.190</td>
<td>650</td>
</tr>
<tr>
<td>Other (20 Iter)</td>
<td>0.507</td>
<td>0.254</td>
<td>1300</td>
</tr>
<tr>
<td>Other (50 Iter)</td>
<td>0.481</td>
<td>0.313</td>
<td>3250</td>
</tr>
</tbody>
</table>

Click Prediction 1 TB advertising dataset, m4.4xlarge machines, perfect scaling.

![Cost vs Billable Time](image-url)
K-Means Clustering

\[ \| x_i - \mu_j \| \]

\[ \frac{1}{n} \sum_i \min_j \| x_i - \mu_j \|^2 \]
## K-Means Clustering

<table>
<thead>
<tr>
<th>Method</th>
<th>Accurate?</th>
<th>Passes</th>
<th>Efficient Tuning</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lloyds [1]</td>
<td>Yes*</td>
<td>5-10</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>K-Means ++ [2]</td>
<td>Yes</td>
<td>k+5 to k+10</td>
<td>No</td>
<td>scikit-learn</td>
</tr>
<tr>
<td>K-Means</td>
<td></td>
<td>[3]</td>
<td>Yes</td>
<td>7-12</td>
</tr>
<tr>
<td>Online [4]</td>
<td>No</td>
<td>1</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Streaming [5,6]</td>
<td>No</td>
<td>1</td>
<td>No</td>
<td>Impractical</td>
</tr>
<tr>
<td>Webscale [7]</td>
<td>No</td>
<td>1</td>
<td>No</td>
<td>spark streaming</td>
</tr>
<tr>
<td>Coreset [8]</td>
<td>No</td>
<td>1</td>
<td>Yes</td>
<td>Impractical</td>
</tr>
<tr>
<td>SageMaker</td>
<td>Yes</td>
<td>1</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

[4] Liberty et. al., 2015  
K-Means Clustering

<table>
<thead>
<tr>
<th></th>
<th>$k$</th>
<th>SageMaker</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Text 1.2GB</strong></td>
<td>10</td>
<td>1.18E3</td>
<td>1.18E3</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>1.00E3</td>
<td>9.77E2</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>9.18E2</td>
<td>9.03E2</td>
</tr>
<tr>
<td><strong>Images 9GB</strong></td>
<td>10</td>
<td>3.29E2</td>
<td>3.28E2</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>2.72E2</td>
<td>2.71E2</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>2.17E2</td>
<td>Failed</td>
</tr>
<tr>
<td><strong>Videos 27GB</strong></td>
<td>10</td>
<td>2.19E2</td>
<td>2.18E2</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>2.03E2</td>
<td>2.02E2</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>1.86E2</td>
<td>1.85E2</td>
</tr>
<tr>
<td><strong>Advertising 127GB</strong></td>
<td>10</td>
<td>1.72E7</td>
<td>Failed</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>1.30E7</td>
<td>Failed</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>1.03E7</td>
<td>Failed</td>
</tr>
<tr>
<td><strong>Synthetic 1100GB</strong></td>
<td>10</td>
<td>3.81E7</td>
<td>Failed</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>3.51E7</td>
<td>Failed</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>2.81E7</td>
<td>Failed</td>
</tr>
</tbody>
</table>

Running Time vs. Number of Clusters

~10x Faster!
Principal Component Analysis (PCA)

\[ \left\| x_i - P(x_i) \right\| \]

\[ \frac{1}{n} \sum_i \left\| x_i - P(x_i) \right\|^2 \]
Principal Component Analysis (PCA)

More than 10x faster at a fraction the cost!

Cost vs. Time

Throughput and Scalability

Cost in Dollars

Billable time in Minutes

Cost in Dollars

Mb/Sec/Machine

Number of Machines

other  sagemaker-deterministic  sagemaker-randomized
Neural Topic Modeling

- Perplexity vs. Number of Topic
  - (~200K documents, ~100K vocabulary)
Time Series Forecasting

### Mean absolute percentage error

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Mean absolute percentage error</th>
<th>P90 Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Traffic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hourly occupancy rate of 963 bay area freeways</td>
<td>0.14</td>
<td>0.13</td>
</tr>
<tr>
<td><strong>Electricity</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Electricity use of 370 homes over time</td>
<td>0.07</td>
<td>0.08</td>
</tr>
<tr>
<td><strong>Pageviews</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page view hits of websites</td>
<td>10k</td>
<td>180k</td>
</tr>
<tr>
<td>10k</td>
<td>0.32</td>
<td>0.29</td>
</tr>
<tr>
<td>180k</td>
<td>0.32</td>
<td>0.31</td>
</tr>
</tbody>
</table>

One hour on p2.xlarge, $1
Pipe Mode (launched May 23rd)

Job Execution Time

Job Startup Time

Throughput

PCA

K-Means
Mxnet as an engine
Deep Dive: SageMaker K-Means
Lloyd's Algorithms

\[
\min_{c,A} \sum_{i=1}^{n} |x_i - c_{A(i)}|^2
\]

Fix A, optimize c

\[c_j = E_{x \in A^{-1}(j)}[x]\]

Fix c, optimize A

\[A(i) = \min_j |x_i - c_j|^2\]
K-Means++

$$\Pr[c_{next} = x] \propto \min_j |x - c_j|^2$$
K-Means++

\[
\Pr[c_{next} = x] \propto \min_j |x - c_j|^2
\]
K-Means++

\[ \Pr[c_{next} = x] \propto \min_j |x - c_j|^2 \]
K-Means++

$$\Pr[c_{next} = x] \propto \min_j |x - c_j|^2$$
Online K-Means

For batch in data:
  Assign points in batch to clusters
  Move centers to new average
Online K-Means

For batch in data:
Assign points in batch to clusters
Move centers to new average
Online K-Means

For batch in data:
Assign points in batch to clusters
Move centers to new average
Online K-Means

For batch in data:
Assign points in batch to clusters
Move centers to new average
Online K-Means

For batch in data:
  Assign points in batch to clusters
  Move centers to new average

Cons
• Sub-optimal Convergence
Online K-Means

For batch in data:
Assign points in batch to clusters
Move centers to new average

Cons
• Sub-optimal Convergence
• Sensitive to initialization
Core Set

Represent $n$ points as $r \ll n$ weighted points
Core Set

Represent n points as r ≪ n weighted points

Practical Heuristic
For r>k, centers of (crude) r-means ≈ core set for k-means
Core Set

Represent \( n \) points as \( r \ll n \) weighted points

Solves issue #1 in online K-Means
(suboptimal convergence)
Adding Centers Over Time

Add more centers over time, based on cost = \( \min_j |x - c_j|^2 \)
Adding Centers Over Time

Add more centers over time, based on cost = \( \min_j |x - c_j|^2 \)
Adding Centers Over Time

Add more centers over time, based on cost = \( \min_j |x - c_j|^2 \)
Adding Centers Over Time

Add more centers over time, based on cost = \( \min_j |x - c_j|^2 \)
Adding Centers Over Time

Add more centers over time, based on cost = \( \min_j |x - c_j|^2 \)

Overcomes poor initializations
Adding Centers Over Time

Add more centers over time, based on cost = $\min_j |x - c_j|^2$

Overcomes poor initializations

We already have r slots...
Start with r centers, regularly throw away least useful centers
# Experiments

<table>
<thead>
<tr>
<th></th>
<th>k</th>
<th>SageMaker</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Text 1.2GB</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.18E3</td>
<td>1.18E3</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>1.00E3</td>
<td>9.77E2</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>9.18.E2</td>
<td>9.03E2</td>
<td></td>
</tr>
<tr>
<td><strong>Images 9GB</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3.29E2</td>
<td>3.28E2</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>2.72E2</td>
<td>2.71E2</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>2.17E2</td>
<td>Failed</td>
<td></td>
</tr>
<tr>
<td><strong>Videos 27GB</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2.19E2</td>
<td>2.18E2</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>2.03E2</td>
<td>2.02E2</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>1.86E2</td>
<td>1.85E2</td>
<td></td>
</tr>
<tr>
<td><strong>Advertising 127GB</strong></td>
<td></td>
<td>Failed</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.72E7</td>
<td>Failed</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>1.30E7</td>
<td>Failed</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>1.03E7</td>
<td>Failed</td>
<td></td>
</tr>
<tr>
<td><strong>Synthetic 1100GB</strong></td>
<td></td>
<td>Failed</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3.81E7</td>
<td>Failed</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>3.51E7</td>
<td>Failed</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>2.81E7</td>
<td>Failed</td>
<td></td>
</tr>
</tbody>
</table>

---

Running Time vs. Number of Clusters

~10x Faster!

Other = Lloyds + kmeans || Requires 7-12 passes over the data

Online k-means achieves significantly worse loss
Using Amazon SageMaker Algorithms on AWS
From Amazon SageMaker Notebooks

```python
import boto3
import sagemaker

sess = sagemaker.Session()

pca = sagemaker.estimator.Estimator(containers[boto3.Session().region_name],
                                     role,
                                     train_instance_count=1,
                                     train_instance_type='ml.c4.xlarge',
                                     output_path=output_location,
                                     sagemaker_session=sess)

pca.set_hyperparameters(feature_dim=50000,
                         num_components=10,
                         subtract_mean=True,
                         algorithm_mode='randomized',
                         mini_batch_size=200)

pca.fit({'train': s3_train_data})

pca_predictor = pca.deploy(initial_instance_count=1,
                             instance_type='ml.c4.xlarge')
```
From Command Line

profile=<your_profile>
arn_role=<your_arn_role>
training_image=382416733822.dkr.ecr.us-east-1.amazonaws.com/kmeans:1
training_job_name=clutering_text_documents_`date '+%Y-%m-%d-%H-%M-%S'`
aws --profile $profile \
  --region us-east-1 \n  sagemaker create-training-job \
    --training-job-name $training_job_name \n    --algorithm-specification TrainingImage=$training_image,TrainingInputMode=File \n    --hyper-parameters k=10,feature_dim=1024,mini_batch_size=1000 \n    --role-arn $arn_role \n    --input-data-config '{"channelName": "train", "DataSource": {"S3DataSource": {"S3DataType": "S3Prefix", "S3Uri": ":/kmeans_demo/train", "S3DataDistributionType": "ShardedByS3Key"}, "CompressionType": "None", "RecordWrapperType": "None"},"ChannelName": "train", "DataSource": {"S3DataSource": {"S3DataType": ":/kmeans_demo/train", "S3DataDistributionType": "ShardedByS3Key"}, "CompressionType": "None", "RecordWrapperType": "None"},"ChannelName": "train", "DataSource": {"S3DataSource": {"S3DataType": ":/kmeans_demo/train", "S3DataDistributionType": "ShardedByS3Key"}, "CompressionType": "None", "RecordWrapperType": "None"}'} \
    --output-data-config S3OutputPath=s3://training_output/$training_job_name \
    --resource-config InstanceCount=2,InstanceType=ml.c4.8xlarge,VolumeSizeInGB=50 \
    --stopping-condition MaxRuntimeInSeconds=3600
# Python/PySpark Example

```python
from sagemaker_pyspark import SageMakerEstimator

features = spark.read.parquet('s3://<bucket>/<dataset>

algorithm = SageMakerEstimator(
    trainingImage=ntm_container,
    modelImage=ntm_container,
    trainingInstanceType='ml.p3.8xlarge',
    trainingInstanceCount=16,
    endpointInstanceType='ml.c5.2xlarge',
    endpointInitialInstanceCount=4,
    hyperParameters={
        "num_topics": "100",
        "feature_dim": "250000",
        "mini_batch_size": "10000",
    },
    sagemakerRole=IAMRole(role_arn)
)

model = algorithm.fit(features)
```

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
SageMaker + Spark =

// Scala Example
import com.amazonaws.services.sagemaker.sparksdk.{IAMRole, SageMakerEstimator}

val features = spark.read.parquet("s3://<bucket>/<dataset>")

val algorithm = new SageMakerEstimator(
    trainingImage = ntm_container,
    modelImage = ntm_container,
    trainingInstanceType = "ml.p3.8xlarge",
    trainingInstanceCount = 16,
    endpointInstanceType = "ml.c5.2xlarge",
    endpointInitialInstanceCount = 4,
    hyperParameters = Map(
        "num_topics" -> "100",
        "feature_dim" -> "250000",
        "mini_batch_size" -> "10000"
    ),
    sagemakerRole = IAMRole(roleArn)
)

val model = estimator.fit(features)
SageMaker + Spark =

1. Load and transform data
2. Generate Features
3. Train a model using SageMaker
4. Generate predictions
5. Use/save the predictions

Runs on your EMR cluster (compute heavy – e.g. 16 x m4.4xlarge)

Uses your algorithm

Saves the model to S3

Model Artifact

Amazon EMR

Amazon SageMaker

SageMaker Managed Training Cluster
(optimized for your algorithm e.g. 4 x p3.2xlarge)

Uses your algorithm

SageMaker Managed Hosting Cluster
(optimized for your algorithm e.g. 8 x c5.2xlarge)

Amazon ECR

Uses your algorithm
Amazon SageMaker - Try It Out

Dashboard
- Notebook instances
- Jobs

Resources
- Models
- Endpoint configuration
- Endpoints

Overview
- Notebook instance
  - Explore AWS data in your notebooks, and use algorithms to create models via training jobs.
  - Create notebook instance

- Jobs
  - Track training jobs at your desk or remotely. Leverage high-performance AWS algorithms.
  - View jobs

- Models
  - Create models for hosting from job outputs, or import externally trained models into Amazon SageMaker.
  - View models

- Endpoint
  - Deploy endpoints for developers to use in production. A/B Test model variants via an endpoint.
  - View endpoints