THE MISSING PIECE IN COMPLEX ANALYTICS: SCALABLE, LOW LATENCY MODEL SERVING AND MANAGEMENT WITH VELOX

Daniel Crankshaw, Peter Bailis, Joseph Gonzalez, Haoyuan Li, Zhao Zhang, Ali Ghodsi, Michael Franklin, and Michael I. Jordan
UC Berkeley AMPLab

CIDR 2015
Talk Outline

• ML model management today
• Velox system architecture
• Key idea: Split model family
• Prediction serving
• Model management
• Next directions
Catify: Music for Cats
MODELING TASK

Rating

Songs
MODELING TASK

Ratings vs. Songs

Prediction
Data
Data → Model
Data → Training → Model
BERKELEY DATA ANALYTICS STACK (BDAS)

- Spark Streaming
- BlinkDB
- Spark SQL
- GraphX
- MLBase
- MLlib
- Spark
- Mesos
- Hadoop Yarn
- Tachyon
- HDFS, S3, ...
Catify: Music for Cats
Catify: Music for Cats

<table>
<thead>
<tr>
<th>CatID</th>
<th>Song</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>2.1</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>3.7</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>3.7</td>
</tr>
<tr>
<td>3</td>
<td>273</td>
<td>4.2</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1.9</td>
</tr>
</tbody>
</table>
Catify: Music for Cats

<table>
<thead>
<tr>
<th>CatID</th>
<th>Song</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>2.1</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>3.7</td>
</tr>
<tr>
<td>3</td>
<td>273</td>
<td>4.2</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1.9</td>
</tr>
</tbody>
</table>
Catify: Music for Cats

<table>
<thead>
<tr>
<th>CatID</th>
<th>Song</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>2.1</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>3.7</td>
</tr>
<tr>
<td>3</td>
<td>273</td>
<td>4.2</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1.9</td>
</tr>
</tbody>
</table>
Catify: Music for Cats

<table>
<thead>
<tr>
<th>CatID</th>
<th>Song</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>2.1</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>3.7</td>
</tr>
<tr>
<td>3</td>
<td>273</td>
<td>4.2</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1.9</td>
</tr>
</tbody>
</table>
Prediction Latency

Prediction Error

Lower is Better
Prediction Latency

Prediction Error

Lower is Better

Lower is Better
Online Retraining

- e.g.,

Prediction Latency

Lower is Better

Prediction Error

Lower is Better
Catify: Music for Cats
Catify: Music for Cats

Apache Web Server
Node.js App Server
MySQL

Tachyon + HDFS
Catify: Music for Cats

Apache Web Server

Node.js App Server

MySQL

Tachyon + HDFS

Spark

Pipeline
Catify: Music for Cats
Catify: Music for Cats

$O(\text{users} \times \text{songs})$
Catify: Music for Cats

NGINX
Node.js App Server
MySQL

PipeLine
Spark

Tachyon + HDFS
Catify: Music for Cats

NGINX

Node.js App Server

MySQL

New Model

Tachyon + HDFS

Pipeline

Spark

Catify: Music for Cats
Catify: Music for Cats

- Node.js App Server
- MySQL
- Tachyon + HDFS
- Training Data
- New Model
- Spark
**Prediction**

**Latency**

**Online Retraining**

*e.g., Spark*
Prediction Latency

Online Retraining
e.g.,

Spark

Prediction Error

Full pre-materialization
e.g.,

MySQL
What’s wrong?
What’s wrong?

1. Predictions have either:
What's wrong?

1. Predictions have either:
   a. High latency, low staleness
What’s wrong?

1. Predictions have either:
   a. High latency, low staleness
   b. Low latency, high staleness
What’s wrong?

1. Predictions have either:
   a. High latency, low staleness
   b. Low latency, high staleness

2. Limited optimization of model semantics
What’s wrong?

1. Predictions have either:
   a. High latency, low staleness
   b. Low latency, high staleness
2. Limited optimization of model semantics
3. Ad-hoc lifecycle management
Talk Outline

- ML model management today
- Velox system architecture
- Split model family
- Prediction serving
- Model management
- Next directions
Talk Outline

- ML model management today
- **Velox system architecture**
- Split model family
- Prediction serving
- Model management
- Next directions
Prediction Latency

Online Retraining
e.g., Spark

Prediction Error

Full pre-materialization
e.g., MySQL
Online Retraining

Full pre-materialization

e.g., Spark
e.g., VELOX

e.g., MySQL®
VELOX GOALS
VELOX GOALS

1. Low latency and low error predictions
VELOX GOALS

1. Low latency and low error predictions
2. Cross-cutting model-specific optimizations
VELOX GOALS

1. Low latency and low error predictions
2. Cross-cutting model-specific optimizations
3. Unified system eases operation
Prediction Error

Online Retraining
e.g.,
Spark

Full pre-materialization
e.g.,
VELOX
MySQL
key idea: split model into staleness insensitive and staleness sensitive components

Online Retraining

Prediction Error

Prediction Latency

VELOX

Full pre-materialization e.g.,

Spark

MySQL®
key idea: 
**split model** into 
staleness insensitive 
and 
staleness sensitive 
components
Prediction Error

Online Retraining
e.g., Spark

BATCH
INCREMENTAL

key idea: split model into staleness insensitive and staleness sensitive components

Full pre-materialization e.g., MySQL

VELOX
BERKELEY DATA ANALYTICS STACK (BDAS)

- Spark Streaming
- BlinkDB
- Spark SQL
- GraphX
- MLbase
- MLlib
- Spark
- Mesos
- Hadoop Yarn
- Tachyon
  - HDFS, S3, …
THE MISSING PIECE

Training

- Spark Streaming
- BlinkDB
- Spark SQL
- Graph X
- MLbase
- ML library

Spark

Mesos          Hadoop Yarn

Tachyon

HDFS, S3, ...
THE MISSING PIECE

Training
- Spark Streaming
- BlinkDB
- Spark SQL
- Graph X

Management + Serving
- MLbase
- ML library

Spark

Mesos

Hadoop Yarn

Tachyon
HDFS, S3, …
THE MISSING PIECE

Training
- Spark Streaming
- BlinkDB
- Spark SQL
- Graph X
- MLbase
- ML library

Management + Serving
- Velox

Spark

Mesos

Hadoop Yarn

Tachyon

HDFS, S3, …
THE MISSING PIECE

Training

Spark Streaming
BlinkDB
Spark SQL
Graph X
MLbase
ML library

Spark

Management + Serving

Velox

Mesos

Hadoop Yarn

Tachyon
HDFS, S3, …
THE MISSING PIECE

Training
- Spark Streaming
- BlinkDB
- Spark SQL
- Graph X
- MLbase
- ML library

Management + Serving
- Velox
  - Model Manager

Mesos
- Hadoop Yarn

Tachyon
- HDFS, S3, …
THE MISSING PIECE

Training
- Spark Streaming
- BlinkDB
- Spark SQL
- Graph X
- MLbase
- ML library
- Spark

Management + Serving
- Velox
  - Model Manager
  - Prediction Service

Mesos

Hadoop Yarn

Tachyon

HDFS, S3, …
PREDICTION SERVICE
PREDICTION SERVICE

1. Implements model serving API
PREDICTION SERVICE

1. Implements model serving API
2. Low latency; < 10ms response time
PREDICTION SERVICE

1. Implements model serving API
2. Low latency; < 10ms response time
3. “Fuzzy” materialized view of model state
PREDICTION SERVICE

1. Implements model serving API
2. Low latency; < 10ms response time
3. “Fuzzy” materialized view of model state

MODEL MANAGER
PREDICTION SERVICE

1. Implements model serving API
2. Low latency; < 10ms response time
3. “Fuzzy” materialized view of model state

MODEL MANAGER

1. Maintains models via online and batch retraining
PREDICTION SERVICE

1. Implements model serving API
2. Low latency; < 10ms response time
3. “Fuzzy” materialized view of model state

MODEL MANAGER

1. Maintains models via online and batch retraining
2. Stores model catalog, metadata, versioning
PREDICTION SERVICE

1. Implements model serving API
2. Low latency; < 10ms response time
3. “Fuzzy” materialized view of model state

MODEL MANAGER

1. Maintains models via online and batch retraining
2. Stores model catalog, metadata, versioning
3. Contains library of standard models + custom API
Talk Outline

• ML model management today
• **Velox system architecture**
  • Key idea: Split model family
  • Prediction serving
• Model management
• Next directions
Talk Outline

• ML model management today
• Velox system architecture
• **Key idea: Split model family**
• Prediction serving
• Model management
• Next directions
PERSONALIZED MODELING
PERSONALIZED MODELING
PERSONALIZED MODELING

A Separate Model for Each User?
PERSONALIZED MODELING

A Separate Model for Each User?

Computationally Inefficient
many complex models
PERSONALIZED MODELING

A Separate Model for Each User?

Computationally Inefficient
many complex models

Statistically Inefficient
not enough data per user
Input (Song) → Rating
Input (Song) → Split → Rating
Input (Song) → Split → Rating
PERSONALIZED MODELING

Input (Song)

Personalized User Model
PERSONALIZED MODELING

Shared Basis Feature Model

Input (Song)

Personalized User Model
PERSONALIZED MODELING

Shared Basis Feature Model
Trained across users
Changes Slowly

Input (Song)

Personalized User Model
PERSONALIZED MODELING

Shared Basis Feature Model
- Trained across users
- Changes Slowly

Input (Song)

Personalized User Model
- Trained for each user
- Changes Quickly

Diagram showing the flow of input (Song) through a shared basis feature model, which changes slowly when trained across users, and a personalized user model, which changes quickly when trained for each user.
SPLIT MODEL FORMULATION

Shared Basis Feature Model

Input (Song)

Personalized User Model
SPLIT MODEL FORMULATION

Shared Basis Feature Model

Personalized User Model

Input (Song)
SPLIT MODEL FORMULATION

Shared Basis Feature Model

Input (Song)

Personalized User Model

Meow
SPLIT MODEL FORMULATION

Shared Basis Feature Model

Personalized User Model

Input (Song)
MATHEMATICAL FORMULATION

Input (Song)
MATHEMATICAL FORMULATION

Input (Song) $x$
MATHEMATICAL FORMULATION

Input
(Song)

\( x \)

Shared Basis
Feature Models
MATHEMATICAL FORMULATION

Input (Song)

\( x \)

\[ f(x; \theta) \]

Shared Basis Feature Models
MATHEMATICAL FORMULATION

Input (Song)

$x$

$f(x; \theta)$

Changes slowly

Shared Basis Feature Models
MATHEMATICAL FORMULATION

Input (Song) $x$

$f(x; \theta)$

Changes slowly

Shared Basis Feature Models

Personalized User Model
MATHEMATICAL FORMULATION

Input (Song) $x$

Shared Basis Feature Models

Changes slowly

$f(x; \theta) \cdot w_u$

Personalized User Model
MATHEMATICAL FORMULATION

Input (Song) $x$

\[ f(x; \theta) \cdot w_u \]

Changes slowly
Shared Basis Feature Models

Personalized User Model
Highly dynamic
MATHEMATICAL FORMULATION

\[ f(x; \theta) \cdot w_u = \text{Rating} \]

Input (Song) \( x \)

Changes slowly

Shared Basis Feature Models

Personalized User Model

Highly dynamic
MATHEMATICAL FORMULATION

\[ f(\mathbf{x}; \theta) \cdot w_u = \text{Rating} \]

- Input (Song) \( \mathbf{x} \)
- Changes slowly
- Shared Basis Feature Models
- Personalized User Model
- Highly dynamic
- Terrible
Talk Outline

• ML model management today
• Velox system architecture
• **Key idea: Split model family**
• Prediction serving
• Model management
• Next directions
Talk Outline

• ML model management today
• Velox system architecture
• Key idea: Split model family
• **Prediction serving**
• Model management
• Next directions
System Architecture

- **Training**
  - Spark
  - BlinkDB
  - Shark
  - GraphX
  - MLbase
  - ML library

- **Management + Serving**
  - Velox
    - Model Manager
    - Prediction Service

- **System Infrastructure**
  - Mesos
  - Hadoop Yarn
  - Mesos

- **Storage**
  - Tachyon
  - HDFS, S3, ...
PREDICTION API

Simple point queries:

GET /velox/catify/predict?userid=22&song=27632
PREDICTION API

Simple point queries:

```
GET /velox/catify/predict?userid=22&song=27632
```

More complex ordering queries:

```
GET /velox/catify/predict_top_k?userid=22&k=100
```
def predict( u: UUID, x: Context )

\[ w_u \cdot f(x; \theta) \]

PREDICTIONS
def predict(u: UUID, x: Context)

Look up user weight

$w_u \cdot f(x; \theta)$
def predict( u: UUID, x: Context )

$w_u \cdot f(x; \theta)$

Look up user weight

Primary key lookup
def predict( u: UUID, x: Context )

Look up user weight
Primary key lookup
Partition query by user: always local

\[ \mathcal{W}_u \cdot f(x; \theta) \]
def predict( u: UUID, x: Context )

$w_u \cdot f(x; \theta)$

Compute Features

user independent
def predict(u: UUID, x: Context)

Feature computation could be costly

\[\mathbf{w}_u \cdot f(x; \theta)\]

user independent
def predict( u: UUID, x: Context )

Feature computation could be costly

Compute Features

Cache features for reuse across users

\( \mathbf{w}_u \cdot f(x; \theta) \)

user independent
Feature caching leads to order-of-magnitude reduction in latency.
TOP-K QUERIES

Query predicate to pre-filter candidate set

All Songs
TOP-K QUERIES

Query predicate to pre-filter candidate set

All Songs ➔ Playlist Keywords
TOP-K QUERIES

Query predicate to pre-filter candidate set

All Songs \rightarrow Playlist Keywords \rightarrow Candidate Songs
TOP-K QUERIES

Query predicate to pre-filter candidate set

All Songs → Playlist Keywords → Candidate Songs

Score and rank all candidates
TOP-K QUERIES

Query predicate to pre-filter candidate set

By exploiting split model design we can leverage:

Score and rank all candidates
TOP-K QUERIES

Query predicate to pre-filter candidate set

By exploiting split model design we can leverage:

A. Shrivastava, P. Li. “Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS).” NIPS'14 Best Paper
TOP-K QUERIES

Query predicate to pre-filter candidate set

By exploiting split model design we can leverage:

A. Shrivastava, P. Li. “Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS).” NIPS’14 Best Paper


All Songs → Playlist Keywords → Candidate Songs

Score and rank all candidates
Talk Outline

- ML model management today
- Velox system architecture
- Key idea: Split model family
- Prediction serving
- Model management
- Next directions
Talk Outline

• ML model management today
• Velox system architecture
• Key idea: Split model family
• Prediction serving
• Model management
• Next directions
System Architecture

Training
- Spark
- BlinkDB
- Shark
- Graph X
- MLbase
- ML library

Management + Serving
- Velox
  - Model Manager
  - Prediction Service

Mesos
- Hadoop Yarn
- Mesos

Tachyon
- HDFS, S3, …
1. Online and offline model training
2. Sample bias problem
FEEDBACK API

Simple direct value feedback:

```
POST /velox/catify/observe?userid=22&song=27&score=3.7
```
FEEDBACK API

Simple direct value feedback:

POST /velox/catify/observe?userid=22&song=27&score=3.7

Online Learning

Continuously update user models in Velox
FEEDBACK API

Simple direct value feedback:

POST /velox/catify/observe?userid=22&song=27&score=3.7

Online Learning

Continuously update user models in Velox

Offline Learning

Logged to DFS for feature learning in Spark
FEEDBACK API

Simple direct value feedback:

```
POST /velox/catify/observe?userid=22&song=27&score=3.7
```

**Online Learning**

Continuously update user models in Velox

**Offline Learning**

Logged to DFS for feature learning in Spark

**Evaluation**

Continuously assess model performance
def observe(u: UUID, x: Context, y: Score)

\[ w_u \cdot f(x; \theta) \]
ONLINE LEARNING

def observe(u: UUID, x: Context, y: Score)

Update $w_u$ with new training point

$w_u \cdot f(x; \theta)$
def observe(u: UUID, x: Context, y: Score)

Update $w_u$ with new training point

Stochastic gradient descent

$w_u \cdot f(x; \theta)$
ONLINE LEARNING

```python
def observe(u: UUID, x: Context, y: Score):
  w_u with new training point

  \[ w_u \cdot f(x; \theta) \]

  Stochastic gradient descent

  Incremental linear algebra
```
def retrain(trainingData: RDD)

\[ w_u \cdot f(x; \theta) \]

Efficient batch training using Spark

Spark Based Training Algs.
def retrain(trainingData: RDD)

\[ \mathbf{w}_u \cdot f(x; \theta) \]

Efficient batch training using Spark

When do we retrain?
OFFLINE LEARNING

def retrain(trainingData: RDD)

\[ w_u \cdot f(x; \theta) \]

Efficient batch training using Spark

When do we retrain?

Periodically

Spark Based Training Algs.
def retrain(trainingData: RDD)

\[ w_u \cdot f(x; \theta) \]

Efficient batch training using Spark

When do we retrain?

Periodically

Trigger by the evaluation system
Velox keeps models updated at low latency.
Sample Bias: model affects the training data.
ALWAYS SERVE THE BEST SONG?
ALWAYS SERVE THE BEST SONG?
VELOX SOLUTION

With prob. $1 - \epsilon$ serve the best predicted song
VELOX SOLUTION

With prob. $1 - \epsilon$ serve the best predicted song
VELOX SOLUTION

With prob. $1 - \epsilon$ serve the best predicted song
With prob. $\epsilon$ pick a \textit{random} song
VELOX SOLUTION

With prob. $1 - \epsilon$ serve the best predicted song
With prob. $\epsilon$ pick a random song
VELOX SOLUTION

With prob. $1 - \epsilon$ serve the best predicted song
With prob. $\epsilon$ pick a *random* song

Epsilon Greedy
VELOX SOLUTION

With prob. $1 - \epsilon$ serve the best predicted song
With prob. $\epsilon$ pick a random song

Epsilon Greedy

Active Learning

Opportunity to explore new systems for this emerging analytics workload
Talk Outline

- ML model management today
- Velox system architecture
- Key idea: Split model family
- Prediction serving
- **Model management**
- Next directions
Talk Outline

• ML model management today
• Velox system architecture
• Split model family
• Prediction serving
• Model management
• Next directions
OPEN CHALLENGES FOR DATABASE SYSTEMS
OPEN CHALLENGES FOR DATABASE SYSTEMS

- Going beyond the split model family
OPEN CHALLENGES FOR DATABASE SYSTEMS

• Going beyond the split model family
  • logical model pipeline language
OPEN CHALLENGES FOR DATABASE SYSTEMS

• Going beyond the split model family
  • logical model pipeline language

• More generic training pipelines
OPEN CHALLENGES FOR DATABASE SYSTEMS

• Going beyond the split model family
  • logical model pipeline language

• More generic training pipelines
  • standard set of physical operators
OPEN CHALLENGES FOR DATABASE SYSTEMS

- Going beyond the split model family
  - logical model pipeline language

- More generic training pipelines
  - standard set of physical operators

- Automatically choose split for online & offline training
OPEN CHALLENGES FOR DATABASE SYSTEMS

• Going beyond the split model family
  • logical model pipeline language

• More generic training pipelines
  • standard set of physical operators

• Automatically choose split for online & offline training
  • view maintenance and query optimization
OPEN CHALLENGES FOR DATABASE SYSTEMS

• Going beyond the split model family
  • logical model pipeline language

• More generic training pipelines
  • standard set of physical operators

• Automatically choose split for online & offline training
  • view maintenance and query optimization

• Ensure user privacy
OPEN CHALLENGES FOR DATABASE SYSTEMS

• Going beyond the split model family
  • logical model pipeline language

• More generic training pipelines
  • standard set of physical operators

• Automatically choose split for online & offline training
  • view maintenance and query optimization

• Ensure user privacy
  • Privacy-Preserving DBMS
Data
Data → Model
Data → Training → Model
The future of research in scalable learning systems will be in the integration of the learning lifecycle:
THE MISSING PIECE

Training

- Spark Streaming
- BlinkDB
- Spark SQL
- GraphX
- MLbase
- ML library

Management + Serving

Velox

- Model Manager
- Prediction Service

Spark

Mesos

Hadoop Yarn

Tachyon

HDFS, S3, …
key idea: **split model** into 
staleness insensitive 
and 
staleness sensitive 
components 

Full pre-materialization 
e.g.,

**VELOX**
SUMMARY
Today: model training and serving relies on ad-hoc, manual processes spread across multiple systems.
Today: model training and serving relies on ad-hoc, manual processes spread across multiple systems. The Velox system automatically maintains multiple models while providing low latency, scalable, and personalized predictions.
Today: model training and serving relies on ad-hoc, manual processes spread across multiple systems

The Velox system automatically maintains multiple models while providing low latency, scalable, and personalized predictions

Velox is coming soon as part of BDAS
Today: model training and serving relies on ad-hoc, manual processes spread across multiple systems.

The Velox system automatically maintains multiple models while providing low latency, scalable, and personalized predictions.

Velox is coming soon as part of BDAS.

https://amplab.cs.berkeley.edu/projects/velox/
QUESTIONS?