Evan Sparks and Ameet Talwalkar
UC Berkeley
Three Converging Trends
Three Converging Trends

Big Data
Three Converging Trends

Big Data

Distributed Computing

Amazon Web Services

Google Compute Engine

Rackspace

Microsoft Azure
Three Converging Trends

Big Data

Distributed Computing

Machine Learning
Three Converging Trends

Big Data

MLbase

Machine Learning

Distributed Computing
Vision
MLlib
MLI
ML Optimizer
Release Plan
Problem: Scalable implementations difficult for ML Developers...
Problem: Scalable implementations difficult for ML Developers...
Problem: Scalable implementations difficult for ML Developers...
Problem: ML is difficult for End Users...

Too many algorithms...
Problem: ML is difficult for End Users...

Too many algorithms...

Too many knobs...
Problem: ML is difficult for End Users…

Too many algorithms…

Too many knobs…

Difficult to debug…
Problem: ML is difficult for End Users...

- Too many algorithms...
- Too many knobs...
- Difficult to debug...
- Doesn’t scale...
Problem: ML is difficult for End Users...

- Too many algorithms...
- Too many knobs...
- Difficult to debug...
- Doesn’t scale...
- Doesn’t scale...
- Fast
- Reliable
- Provable
- Accurate

ML Developer
1. Easy scalable ML development (ML Developers)
2. User-friendly ML at scale (End Users)
1. Easy scalable ML development (*ML Developers*)
2. User-friendly ML at scale (*End Users*)

Along the way, we gain insight into data intensive computing
Matlab Stack
Matlab Stack

Single Machine
Matlab Stack

- Lapack: low-level Fortran linear algebra library
Matlab Stack

- **Lapack**: low-level Fortran linear algebra library
- **Matlab Interface**
  - Higher-level abstractions for data access / processing
  - More extensive functionality than Lapack
  - Leverages Lapack whenever possible
Matlab Stack

- Lapack: low-level Fortran linear algebra library
- Matlab Interface
  - Higher-level abstractions for data access / processing
  - More extensive functionality than Lapack
  - Leverages Lapack whenever possible
- Similar stories for R and Python
MLbase Stack

Matlab Interface

Lapack

Single Machine
MLbase Stack

Matlab Interface

Lapack

Single Machine

Runtime(s)
Spark: cluster computing system designed for iterative computation
Spark: cluster computing system designed for iterative computation

MLlib: low-level ML library in Spark
  - Callable from Scala, Java
MLbase Stack

Spark: cluster computing system designed for iterative computation

MLlib: low-level ML library in Spark
   ✦ Callable from Scala, Java

MLI: API / platform for feature extraction and algorithm development
   ✦ Includes higher-level functionality with faster dev cycle than MLlib
Spark: cluster computing system designed for iterative computation

MLlib: low-level ML library in Spark
  ✦ Callable from Scala, Java

MLI: API / platform for feature extraction and algorithm development
  ✦ Includes higher-level functionality with faster dev cycle than MLlib

ML Optimizer: automates model selection
  ✦ Solves a search problem over feature extractors and algorithms in MLI
MLbase Stack Status

- Spark
- MLlib
- MLI
- ML Optimizer

ML Developer

End User
MLbase Stack Status

- Spark
- MLlib
- MLI
- ML Optimizer

ML Developer

End User
MLbase Stack Status

Goal 1: Summer Release

ML Developer

ML Optimizer

MLI

MLlib

Spark

End User
MLbase Stack Status

Goal 1: Summer Release

Goal 2: Winter Release
Example: MLlib
Example: MLlib

- Goal: Classification of text file
Example: MLlib

```scala
def main(args: Array[String]) {
  val sc = new SparkContext("local", "SparkLR")

  // Load data from HDFS
  val data = sc.textFile(args(0))  // RDD[String]

  // User is responsible for formatting/featurizing/normalizing their RDD!
  val featurizedData: RDD[(Double, Array[Double])] = processData(data)
```

- Goal: Classification of text file
- Featurize data manually
**Example: MLlib**

- Goal: Classification of text file
- Featurize data manually
- Calls MLlib’s LR function

```scala
def main(args: Array[String]) {
  val sc = new SparkContext("local", "SparkLR")

  //Load data from HDFS
  val data = sc.textFile(args(0))  //RDD[String]

  //User is responsible for formatting/featurizing/normalizing their RDD!
  val featurizedData: RDD[(Double,Array[Double])] = processData(data)

  //Train the model using MLlib.
  val model = new LogisticRegressionLocalRandomSGD()
    .setStepSize(0.1)
    .setNumIterations(50)
    .train(featurizedData)
}
```
Example: MLI
Example: MLI

```
def main(args: Array[String]) {
    val mc = new MLContext("local", "MLILR")

    // Read in file from HDFS
    val rawTextTable = mc.csvFile(args(0), Seq("class","text"))

    // Run feature extraction
    val classes = rawTextTable(??, "class")
    val ngrams = tfIdf(nGrams(rawTextTable(??, "text"), n=2, top=30000))
    val featureizedTable = classes.zip(ngrams)
}
```

✨ Use built-in feature extraction functionality
Example: MLI

```scala
def main(args: Array[String]) {
  val mc = new MLContext("local", "MLILR")

  // Read in file from HDFS
  val rawTextTable = mc.csvFile(args(0), Seq("class","text"))

  // Run feature extraction
  val classes = rawTextTable(??, "class")
  val ngrams = tfIdf(nGrams(rawTextTable(??, "text"), n=2, top=30000))
  val featureizedTable = classes.zip(ngrams)

  // Classify the data using Logistic Regression.
  val lrModel = LogisticRegression(featureizedTable, stepSize=0.1, numIter=12)
}
```

- Use built-in feature extraction functionality
- MLI Logistic Regression leverages MLlib
Example: MLI

Use built-in feature extraction functionality

MLI Logistic Regression leverages MLlib

Extensions:
  - Embed in cross-validation routine
  - Use different feature extractors / algorithms or write new ones

```scala
def main(args: Array[String]) {
  val mc = new MLContext("local", "MLILR")
  //Read in file from HDFS
  val rawTextTable = mc.csvFile(args(0), Seq("class","text"))
  //Run feature extraction
  val classes = rawTextTable(??, "class")
  val ngrams = tfIdf(nGrams(rawTextTable(??, "text"), n=2, top=30000))
  val featureizedTable = classes.zip(ngrams)
  //Classify the data using Logistic Regression.
  val lrModel = LogisticRegression(featureizedTable, stepSize=0.1, numIter=12)
}
```
Example: ML Optimizer

\[
\begin{align*}
\text{var } X &= \text{load}("text_file", \text{2 to 10}) \\
\text{var } y &= \text{load}("text_file", \text{1}) \\
\text{var } (\text{fn-model, summary}) &= \text{doClassify}(X, y)
\end{align*}
\]

- User declaratively specifies task
- ML Optimizer searches through MLI
Vision
MLlib
MLI
ML Optimizer
Release Plan
Lay of the Land

Ease of use

Performance, Scalability
Matlab, R

Lay of the Land

Ease of use

Performance, Scalability
Matlab, R

Mahout

Ease of use

Performance, Scalability
Lay of the Land

Ease of use

Performance, Scalability

Matlab, R

Mahout

GraphLab, VW
Lay of the Land

Ease of use vs. Performance, Scalability

- Matlab, R
- Mahout
- MLlib
- GraphLab, VW
MLlib

**Classification:** Logistic Regression, Linear SVM (+L1, L2)

**Regression:** Linear Regression (+Lasso, Ridge)

**Collaborative Filtering:** Alternating Least Squares

**Clustering:** K-Means

**Optimization Primitives:** SGD, Parallel Gradient
**MLlib**

**Classification:** Logistic Regression, Linear SVM (+L1, L2)

**Regression:** Linear Regression (+Lasso, Ridge)

**Collaborative Filtering:** Alternating Least Squares

**Clustering:** K-Means

**Optimization Primitives:** SGD, Parallel Gradient

**Included within Spark codebase**
- Unlike Mahout/Hadoop
- Part of Spark 0.8 release
- Continued support via Spark project
MLlib Performance
MLlib Performance

- **Walltime**: elapsed time to execute task
MLlib Performance

- **Walltime**: elapsed time to execute task

- **Weak scaling**
  - fix problem size *per processor*
  - ideally: constant walltime as we grow cluster
MLlib Performance

- **Walltime**: elapsed time to execute task
- **Weak scaling**
  - fix problem size *per processor*
  - ideally: constant walltime as we grow cluster
- **Strong scaling**
  - fix total problem size
  - ideally: linear speed up as we grow cluster
MLlib Performance

- **Walltime**: elapsed time to execute task

- **Weak scaling**
  - fix problem size *per processor*
  - ideally: constant walltime as we grow cluster

- **Strong scaling**
  - fix total problem size
  - ideally: linear speed up as we grow cluster

- **EC2 Experiments**
  - m2.4xlarge instances, up to 32 machine clusters
Logistic Regression - Weak Scaling
Logistic Regression - Weak Scaling

- Full dataset: 200K images, 160K dense features
Logistic Regression - Weak Scaling

- Full dataset: 200K images, 160K dense features
- Similar weak scaling
Logistic Regression - Weak Scaling

- Full dataset: 200K images, 160K dense features
- Similar weak scaling
- MLlib within a factor of 2 of VW’s walltime
Logistic Regression - Strong Scaling
Logistic Regression - Strong Scaling

✦ Fixed Dataset: 50K images, 160K dense features
Logistic Regression - Strong Scaling

- Fixed Dataset: 50K images, 160K dense features
- MLlib exhibits better scaling properties
Logistic Regression - Strong Scaling

- Fixed Dataset: 50K images, 160K dense features
- MLlib exhibits better scaling properties
- MLlib faster than VW with 16 and 32 machines
ALS - Walltime
ALS - Walltime

- Dataset: Scaled version of Netflix data (9X in size)
- Cluster: 9 machines
## ALS - Walltime

<table>
<thead>
<tr>
<th>System</th>
<th>Walltime (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matlab</td>
<td>15443</td>
</tr>
</tbody>
</table>

- Dataset: Scaled version of Netflix data (9X in size)
- Cluster: 9 machines
ALS - Walltime

<table>
<thead>
<tr>
<th>System</th>
<th>Walltime (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matlab</td>
<td>15443</td>
</tr>
<tr>
<td>Mahout</td>
<td>4206</td>
</tr>
</tbody>
</table>

- Dataset: Scaled version of Netflix data (9X in size)
- Cluster: 9 machines
ALS - Walltime

<table>
<thead>
<tr>
<th>System</th>
<th>Walltime (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matlab</td>
<td>15443</td>
</tr>
<tr>
<td>Mahout</td>
<td>4206</td>
</tr>
<tr>
<td>GraphLab</td>
<td>291</td>
</tr>
<tr>
<td>MLlib</td>
<td>481</td>
</tr>
</tbody>
</table>

- Dataset: Scaled version of Netflix data (9X in size)
- Cluster: 9 machines
- MLlib an order of magnitude faster than Mahout
- MLlib within factor of 2 of GraphLab
Deployment Considerations
Deployment Considerations

Vowpal Wabbit, GraphLab

- Data preparation specific to each program
- Non-trivial setup on cluster
- No fault tolerance
Deployment Considerations

Vowpal Wabbit, GraphLab
- Data preparation specific to each program
- Non-trivial setup on cluster
- No fault tolerance

MLlib
- Reads files from HDFS
- Launch/compile/run on cluster with a few commands
- RDD’s are fault tolerance
Vision
MLlib
MLI
ML Optimizer
Release Plan
Lay of the Land

Ease of use

Performance, Scalability

Matlab, R

Mahout

MLlib

GraphLab, VW
Lay of the Land

Ease of use

Performance, Scalability

Matlab, R

MLI

MLlib

Mahout

GraphLab, VW
Current Options
Current Options

+ Easy (Resembles math, limited / no set up cost)
+ Sufficient for prototyping / writing papers
  - Ad-hoc, non-scalable scripts
  - Loss of translation upon re-implementation
Current Options

+ Easy (Resembles math, limited / no set up cost)
+ Sufficient for prototyping / writing papers
  - Ad-hoc, non-scalable scripts
  - Loss of translation upon re-implementation
Current Options

+ Easy (Resembles math, limited / no set up cost)
+ Sufficient for prototyping / writing papers
  - Ad-hoc, non-scalable scripts
  - Loss of translation upon re-implementation

+ Scalable and (sometimes) fast
+ Existing open-source library of ML algorithms
  - Difficult to set up, extend
Examples
Examples

‘Distributed’ Divide-Factor-Combine (DFC)
- Initial studies in MATLAB (Not distributed)
- Distributed prototype involving compiled MATLAB
Examples

‘Distributed’ Divide-Factor-Combine (DFC)
✦ Initial studies in MATLAB (Not distributed)
✦ Distributed prototype involving compiled MATLAB

Mahout ALS with Early Stopping
✦ Theory: simple if-statement (3 lines of code)
Examples

‘Distributed’ Divide-Factor-Combine (DFC)
- Initial studies in MATLAB (*Not distributed*)
- Distributed prototype involving compiled MATLAB

Mahout ALS with Early Stopping
- Theory: simple if-statement (3 lines of code)
- Practice: sift through 7 files, nearly 1K lines of code
Insight: Programming Abstractions
Insight: Programming Abstractions

- Shield ML Developers from low-details: provide familiar mathematical operators in distributed setting

- ML Developer API (MLI)
Insight: Programming Abstractions

✧ Shield ML Developers from low-details: provide familiar mathematical operators in distributed setting

✧ ML Developer API (MLI)
  ✧ Table Computation: MLTable
  ✧ Linear Algebra: MLSubMatrix
  ✧ Optimization Primitives: MLSolve
Insight: Programming Abstractions

- Shield ML Developers from low-details: provide familiar mathematical operators in distributed setting

- ML Developer API (**MLI**)
  - Table Computation: **MLTable**
  - Linear Algebra: **MLSubMatrix**
  - Optimization Primitives: **MLSolve**

- MLI Examples:
  - DFC: \(~50\) lines of code
Insight: Programming Abstractions

✦ Shield ML Developers from low-details: provide familiar mathematical operators in distributed setting

✦ ML Developer API (**MLI**)
  ✦ Table Computation: **MLTable**
  ✦ Linear Algebra: **MLSubMatrix**
  ✦ Optimization Primitives: **MLSolve**

✦ MLI Examples:
  ✦ DFC: ~50 lines of code
  ✦ ALS: early stopping in 3 lines; < 40 lines total
Lines of Code
# Lines of Code

## Logistic Regression

<table>
<thead>
<tr>
<th>System</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matlab</td>
<td>11</td>
</tr>
</tbody>
</table>

## Alternating Least Squares

<table>
<thead>
<tr>
<th>System</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matlab</td>
<td>20</td>
</tr>
</tbody>
</table>
# Lines of Code

## Logistic Regression

<table>
<thead>
<tr>
<th>System</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matlab</td>
<td>11</td>
</tr>
<tr>
<td>Vowpal Wabbit</td>
<td>721</td>
</tr>
</tbody>
</table>

## Alternating Least Squares

<table>
<thead>
<tr>
<th>System</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matlab</td>
<td>20</td>
</tr>
<tr>
<td>Mahout</td>
<td>865</td>
</tr>
<tr>
<td>GraphLab</td>
<td>383</td>
</tr>
</tbody>
</table>
# Lines of Code

## Logistic Regression

<table>
<thead>
<tr>
<th>System</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matlab</td>
<td>11</td>
</tr>
<tr>
<td>Vowpal Wabbit</td>
<td>721</td>
</tr>
<tr>
<td>MLI</td>
<td>55</td>
</tr>
</tbody>
</table>

## Alternating Least Squares

<table>
<thead>
<tr>
<th>System</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matlab</td>
<td>20</td>
</tr>
<tr>
<td>Mahout</td>
<td>865</td>
</tr>
<tr>
<td>GraphLab</td>
<td>383</td>
</tr>
<tr>
<td>MLI</td>
<td>32</td>
</tr>
</tbody>
</table>
MLI Details
MLI Details

OLD
val x: RDD[Array[Double]]
MLI Details

OLD
val x: RDD[Array[Double]]
val x: RDD[spark.util.Vector]
MLI Details

OLD

val x: RDD[Array[Double]]
val x: RDD[spark.util.Vector]
val x: RDD[breeze.linalg.Vector]
MLI Details

OLD
val x: RDD[Array[Double]]
val x: RDD[spark.util.Vector]
val x: RDD[breeze.linalg.Vector]
val x: RDD[BIDMat.SMat]
OLD
val x: RDD[Array[Double]]
val x: RDD[spark.rdd.Vector]
val x: RDD[breeze.linalg.Vector]
val x: RDD[BIDMat.SMat]
MLI Details

OLD
val x: RDD[Array[Double]]
val x: RDD[spark.ml.Vector]
val x: RDD[breeze.linalg.Vector]
val x: RDD[BIDMat.SMat]

NEW
val x: MLTable
MLI Details

OLD
val x: RDD[Array[Double]]
val x: RDD[spark.ml.Vector]
val x: RDD[breeze.linalg.Vector]
val x: RDD[BIDMat.SMat]

NEW
val x: MLTable

✦ Generic interface for feature extraction
✦ Common interface to support an optimizer
✦ Abstract interface for arbitrary backends
MLTable

- **Flexibility when loading data**
  - e.g., CSV, JSON, XML
  - Heterogenous data across columns
  - Missing Data
  - Feature extraction

- **Common Interface**
  - Supports MapReduce and Relational Operators

- Inspired by DataFrames (R) and Pandas (Python)
Feature Extraction

```scala
def main(args: Array[String]) {
  val mc = new MLContext("local")

  // Read in table from file on HDFS.
  val rawTextTable = mc.textFile(args(0))

  // Run feature extraction on the raw text - get the top 30000 bigrams.
  val featurizedTable = tfIdf(nGrams(rawTextTable, n=2, top=30000))

  // Cluster the data using K-Means.
  val kMeansModel = KMeans(featurizedTable, k=50)
}
```
MLSubMatrix

✧ **Linear algebra on local partitions**
  ✧ E.g., matrix-vector operations for mini-batch logistic regression
  ✧ E.g., solving linear system of equations for Alternating Least Squares

✧ **Sparse and Dense Matrix Support**
Alternating Least Squares

object BroadcastALS extends Algorithm {
  def train(trainData: MLNumericTable, trainDataTrans: MLNumericTable,
            m: Int, n: Int, k: Int, lambda: Int, maxIter: Int): ALSModel = {
    val lambI = MLSubMatrix.eye(k).mul(lambda)
    var U = MLSubMatrix.rand(m, k)
    var V = MLSubMatrix.rand(n, k)
    var U_b = trainData.context.broadcast(U)
    var V_b = trainData.context.broadcast(V)
    for (iter <- 0 until maxIter) {
      U = trainData.matrixBatchMap(localALS(_, U_b.value, lambI, k))
      U_b = trainData.context.broadcast(U)
      V = trainDataTrans.matrixBatchMap(localALS(_, V_b.value, lambI, k))
      V_b = trainData.context.broadcast(V)
    }
    new ALSModel(U, V)
  }

  def localALS(trainDataPart: MLSubMatrix, Y: MLSubMatrix, lambI: MLSubMatrix, k: Int){
    var localX = MLSubMatrix.zeros(trainDataPart.numRows, k)
    for (i <- 0 until trainDataPart.numRows) {
      val q = trainDataPart.rowID(i)
      val nz_inds = trainDataPart.nzCols(q)
      val Yq = Y(trainDataPart.nzCols(q), ??)
      localX(i, ??) = ((Yq.transpose times Yq) + lambI)
                      .solve(Yq.transpose times trainDataPart(q, nz_inds).transpose)
    }
    return localX
  }
}
Distributed implementations of common optimization patterns
- E.g., Stochastic Gradient Descent: Applicable to summable ML losses
- E.g., LBFGS: An approximate 2nd-order optimization method
- E.g., ADMM: Decomposition / coordination procedure
Logistic Regression

object LogisticRegression extends Algorithm {
  def sigmoid(z: Scalar) = 1.0 / (1.0 + exp(-1.0*z))

  def gradientFunction(w: MLSubMatrix, x: MLSubMatrix, y: Scalar): MLSubMatrix = {
    x.transpose * (sigmoid(x dot w) - y)
  }

  def train(data: MLNumericTable, p: LogRegParams): LogRegModel = {
    val d = data.numCols
    val params = SGDParams(initweights = MLSubMatrix.zeros(d, 1),
      maxIterations = p.maxIter, learningRate = p.learningRate,
      gradientFunction = gradientFunction)
    val weights = SGD(data, params)
    new LogRegModel(weights)
  }
}

Fig. 13: Logistic Regression Code in MATLAB (top) and MLI (middle, bottom).
MLI Functionality

**Regression:** Linear Regression (+Lasso, Ridge)

**Collaborative Filtering:** Alternating Least Squares

**Clustering:** K-Means

**Classification:** Logistic Regression, Linear SVM (+L1, L2)

**Optimization Primitives:** SGD, Parallel Gradient
**MLI Functionality**

**Regression:** Linear Regression (+Lasso, Ridge)

**Collaborative Filtering:** Alternating Least Squares, [DFC]

**Clustering:** K-Means, [DP-Means]

**Classification:** Logistic Regression, Linear SVM (+L1, L2), Multinominal Regression, [Naive Bayes, Decision Trees]

**Optimization Primitives:** SGD, Parallel Gradient, Local SGD, [L-BFGS, ADMM, Adagrad]

**Feature Extraction:** Principal Component Analysis (PCA), N-grams, feature cleaning / normalization

**ML Tools:** Cross Validation, Evaluation Metrics
Vision
MLlib
MLI
ML Optimizer
Release Plan
What you *want* to do

Build a Classifier for X
<table>
<thead>
<tr>
<th>What you want to do</th>
<th>What you have to do</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build a Classifier for X</td>
<td>Learn the internals of ML classification algorithms, sampling, feature selection, X-validation, ....</td>
</tr>
<tr>
<td></td>
<td>Potentially learn Spark/Hadoop/...</td>
</tr>
<tr>
<td></td>
<td>Implement 3-4 algorithms</td>
</tr>
<tr>
<td></td>
<td>Implement grid-search to find the right algorithm parameters</td>
</tr>
<tr>
<td></td>
<td>Implement validation algorithms</td>
</tr>
<tr>
<td></td>
<td>Experiment with different sampling-sizes, algorithms, features</td>
</tr>
<tr>
<td></td>
<td>....</td>
</tr>
</tbody>
</table>
What you want to do

Build a Classifier for X

What you have to do

- Learn the internals of ML classification algorithms, sampling, feature selection, X-validation,....
- Potentially learn Spark/Hadoop/...
- Implement 3-4 algorithms
- Implement grid-search to find the right algorithm parameters
- Implement validation algorithms
- Experiment with different sampling-sizes, algorithms, features
- ....

and in the end

Ask For Help
Insight: A Declarative Approach

- End Users tell the system what they want, not how to get it
Insight: A Declarative Approach

- End Users tell the system what they want, not how to get it

SQL  Result

MQL  Model
Insight: A Declarative Approach

_end_ Users tell the system what they want, not how to get it

Example: Supervised Classification

```javascript
var X = load("als_clinical", 2 to 10)
var y = load("als_clinical", 1)
var (fn-model, summary) = doClassify(X, y)
```
Insight: A Declarative Approach

End Users tell the system what they want, not how to get it

Example: Supervised Classification

```javascript
var X = load("als_clinical", 2 to 10);
var y = load("als_clinical", 1);
var (fn-model, summary) = doClassify(X, y);
```
ML Optimizer: A Search Problem

✦ System is responsible for searching through model space

✦ Opportunities for physical optimization
Observation: We tend to be I/O bound during model training.
Systems Optimization of Model Search

Observation:
We tend to be I/O bound during model training.

- Idea from databases – shared cursor!

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>Dog</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>Cat</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>Cat</td>
</tr>
<tr>
<td>2</td>
<td>d</td>
<td>Cat</td>
</tr>
<tr>
<td>3</td>
<td>e</td>
<td>Dog</td>
</tr>
<tr>
<td>3</td>
<td>f</td>
<td>Horse</td>
</tr>
<tr>
<td>4</td>
<td>g</td>
<td>Monkey</td>
</tr>
</tbody>
</table>
Systems Optimization of Model Search

Observation:
We tend to be I/O bound during model training.

✧ Idea from databases – shared cursor!

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>Dog</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>Cat</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>Cat</td>
</tr>
<tr>
<td>2</td>
<td>d</td>
<td>Cat</td>
</tr>
<tr>
<td>3</td>
<td>e</td>
<td>Dog</td>
</tr>
<tr>
<td>3</td>
<td>f</td>
<td>Horse</td>
</tr>
<tr>
<td>4</td>
<td>g</td>
<td>Monkey</td>
</tr>
</tbody>
</table>
Systems Optimization of Model Search

Observation:
We tend to be I/O bound during model training.

✧ Idea from databases – shared cursor!

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>Dog</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>Cat</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>Cat</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>d</td>
<td>Cat</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>e</td>
<td>Dog</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>f</td>
<td>Horse</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>g</td>
<td>Monkey</td>
<td></td>
</tr>
</tbody>
</table>
Systems Optimization of Model Search

Observation:
We tend to be I/O bound during model training.

✦ Idea from databases – shared cursor!
✦ Single pass over the data, many models trained
Systems Optimization of Model Search

Observation:
We tend to be I/O bound during model training.

✦ Idea from databases – shared cursor!
✦ Single pass over the data, many models trained
✦ Example – Logistic Regression via SGD

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>Dog</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>Cat</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>Cat</td>
</tr>
<tr>
<td>2</td>
<td>d</td>
<td>Cat</td>
</tr>
<tr>
<td>3</td>
<td>e</td>
<td>Dog</td>
</tr>
<tr>
<td>3</td>
<td>f</td>
<td>Horse</td>
</tr>
<tr>
<td>4</td>
<td>g</td>
<td>Monkey</td>
</tr>
</tbody>
</table>
Relationship with MLI

ML Developer

ML Optimizer

MLI

MLlib

Spark

End User

ML Library

DMX Runtime

Master Server

Meta-Data

Statistics

User Declarative ML Task

ML Contract +

Code

Optimizer

Parser

Executor/Monitoring

result (e.g., fn-model & summary)
MLI provides **common interface** for all algorithms
Relationship with MLI

- MLI provides **common interface** for all algorithms
- **Contracts**: Meta-data for algorithms written against MLI
MLI provides **common interface** for all algorithms

**Contracts**: Meta-data for algorithms written against MLI

- Type (e.g., classification)
- Parameters
- Runtime (e.g., $O(n)$)
- Input-Specification
- Output-Specification
- ...

End User
Vision
MLlib
MLI
ML Optimizer
Release Plan
Contributors

- John Duchi
- Michael Franklin
- Joseph Gonzalez
- Rean Griffith
- Michael Jordan
- Tim Kraska
- Xinghao Pan
- Virginia Smith
- Shivaram Venkatarataram
- Matei Zaharia
Contributors

✦ John Duchi
✦ Michael Franklin
✦ Joseph Gonzalez
✦ Rean Griffith
✦ Michael Jordan
✦ Tim Kraska
✦ Xinghao Pan
✦ Virginia Smith
✦ Shivaram Venkataram
✦ Matei Zaharia
First Release (Summer)

ML Optimizer

MLI

MLlib

Spark

ML Developer

End User
First Release (Summer)

- **MLlib**: low-level ML library and underlying kernels
  - Callable from Scala, Java
  - Included as part of Spark

- **MLI**: API for feature extraction and ML algorithms
  - Platform for ML development
  - Includes more extensive library and with faster dev-cycle than MLlib
Second Release (Winter)

ML Optimizer
MLI
MLlib
Spark

ML Developer
End User

result (e.g., fn-model & summary)
Second Release (Winter)

- **ML Optimizer**: automated model selection
  - Search problem over feature extractors and algorithms in MLI
  - Contracts
  - Restricted query language (MQL)
Second Release (Winter)

- **ML Optimizer**: automated model selection
  - Search problem over feature extractors and algorithms in MLI
  - Contracts
  - Restricted query language (MQL)

- **Feature extraction** for image data
Future Directions
Future Directions

- **Identify minimal set of ML operators**
  - Expose internals of ML algorithms to optimizer
Future Directions

✧ Identify minimal set of ML operators
  ✧ Expose internals of ML algorithms to optimizer

✧ **Unified language** for End Users and ML Developers
Future Directions

✧ **Identify minimal set of ML operators**
  ✧ Expose internals of ML algorithms to optimizer

✧ **Unified language** for End Users and ML Developers

✧ Plug-ins to **Python, R**
Future Directions

- **Identify minimal set of ML operators**
  - Expose internals of ML algorithms to optimizer

- **Unified language** for End Users and ML Developers

- Plug-ins to **Python, R**

- **Visualization** for unsupervised learning and exploration
Future Directions

- **Identify minimal set of ML operators**
  - Expose internals of ML algorithms to optimizer

- **Unified language** for End Users and ML Developers

- Plug-ins to **Python, R**

- **Visualization** for unsupervised learning and exploration

- **Advanced ML capabilities**
  - Time-series algorithms
  - Graphical models
  - Advanced Optimization (e.g., asynchronous computation)
  - Online updates
  - Sampling for efficiency
Contributions encouraged!

MLbase

www.mlbase.org

Berkeley, CA
August 29-30