About Me

- Postdoc in AMPLab
  - Led initial development of MLlib
- Technical Advisor for Databricks
- Assistant Professor at UCLA
- Research interests include scalability and ease-of-use issues in statistical machine learning

MLlib: Spark's Machine Learning Library

Ameet Talwalkar
AMPCAMP 5
November 20, 2014

amplab UC Berkeley
databricks™
UCLA

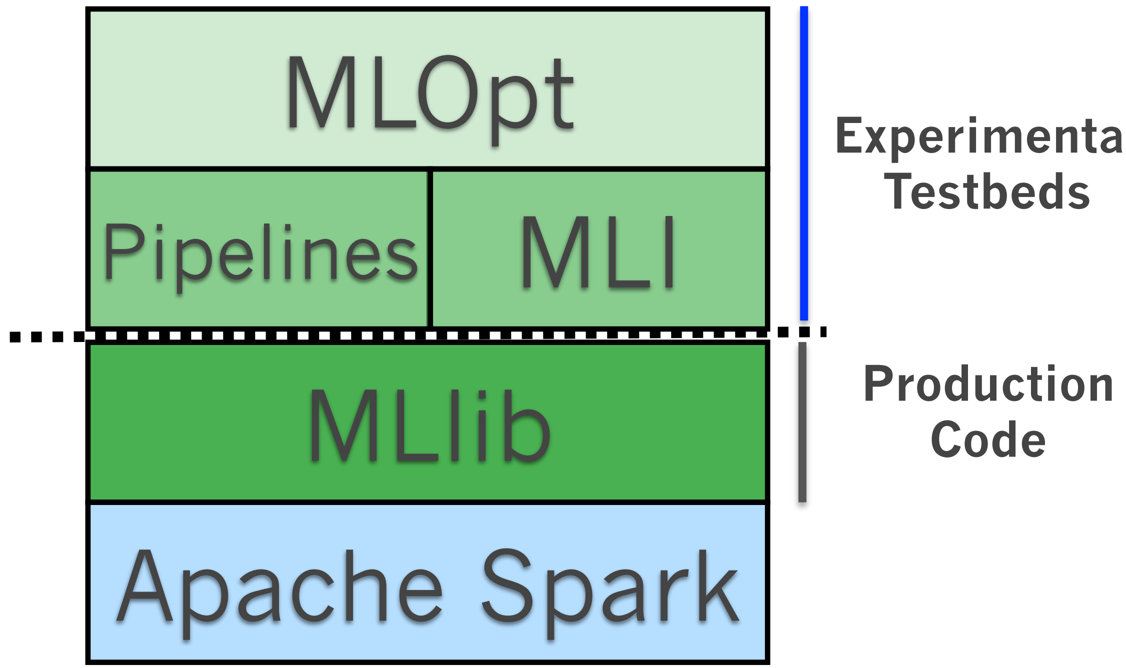
History and Overview

Example Applications

Ongoing Development

MLbase and MLlib

MLbase aims to simplify development and deployment of scalable ML pipelines



MLlib: Spark's core ML library

MLI, Pipelines: APIs to simplify ML development

- Tables, Matrices, Optimization, ML Pipelines

MLOpt: Declarative layer to automate hyperparameter tuning

History of MLlib

Initial Release

- Developed by MLbase team in AMPLab (11 contributors)
- Scala, Java
- Shipped with Spark v0.8 (Sep 2013)

15 months later…

- 80+ contributors from various organization
- Scala, Java, Python
- Latest release part of Spark v1.1 (Sep 2014)

What's in MLlib?

- Alternating Least Squares
- Lasso
- Ridge Regression
- Logistic Regression
- Decision Trees
- Naïve Bayes
- Support Vector Machines
- K-Means
- Gradient descent
- L-BFGS
- Random data generation
- Linear algebra
- Feature transformations
- Statistics: testing, correlation
- Evaluation metrics

Collaborative Filtering for Recommendation

Prediction

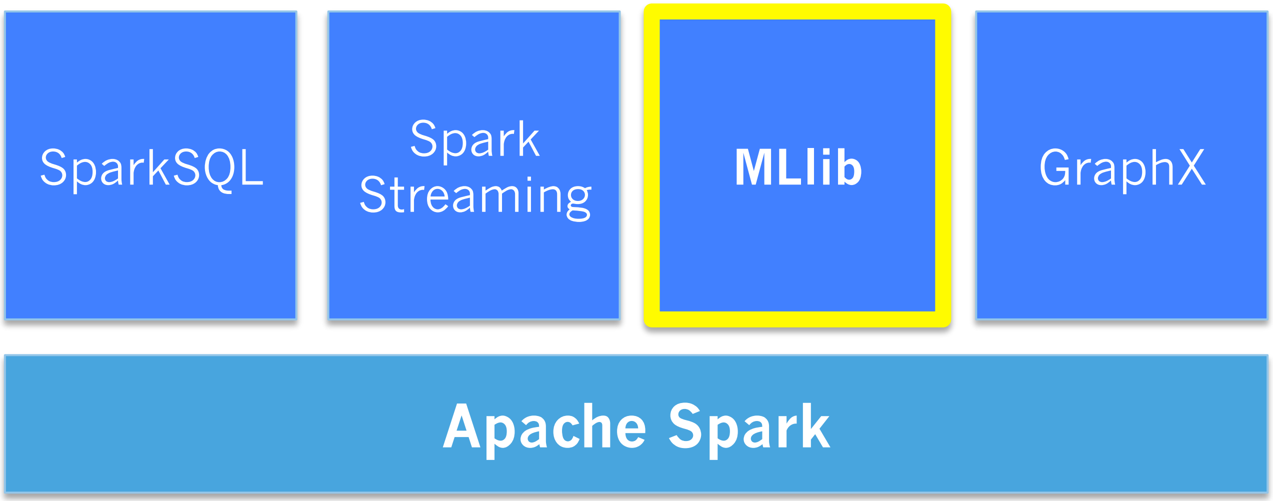
Clustering

Optimization

Many Utilities

Benefits of MLlib

- Part of Spark
  - Integrated data analysis workflow
  - Free performance gains



Benefits of MLlib

- Part of Spark
  - Integrated data analysis workflow
  - Free performance gains
- Scalable
- Python, Scala, Java APIs
- Broad coverage of applications & algorithms
- Rapid improvements in speed & robustness

History and Overview

Example Applications

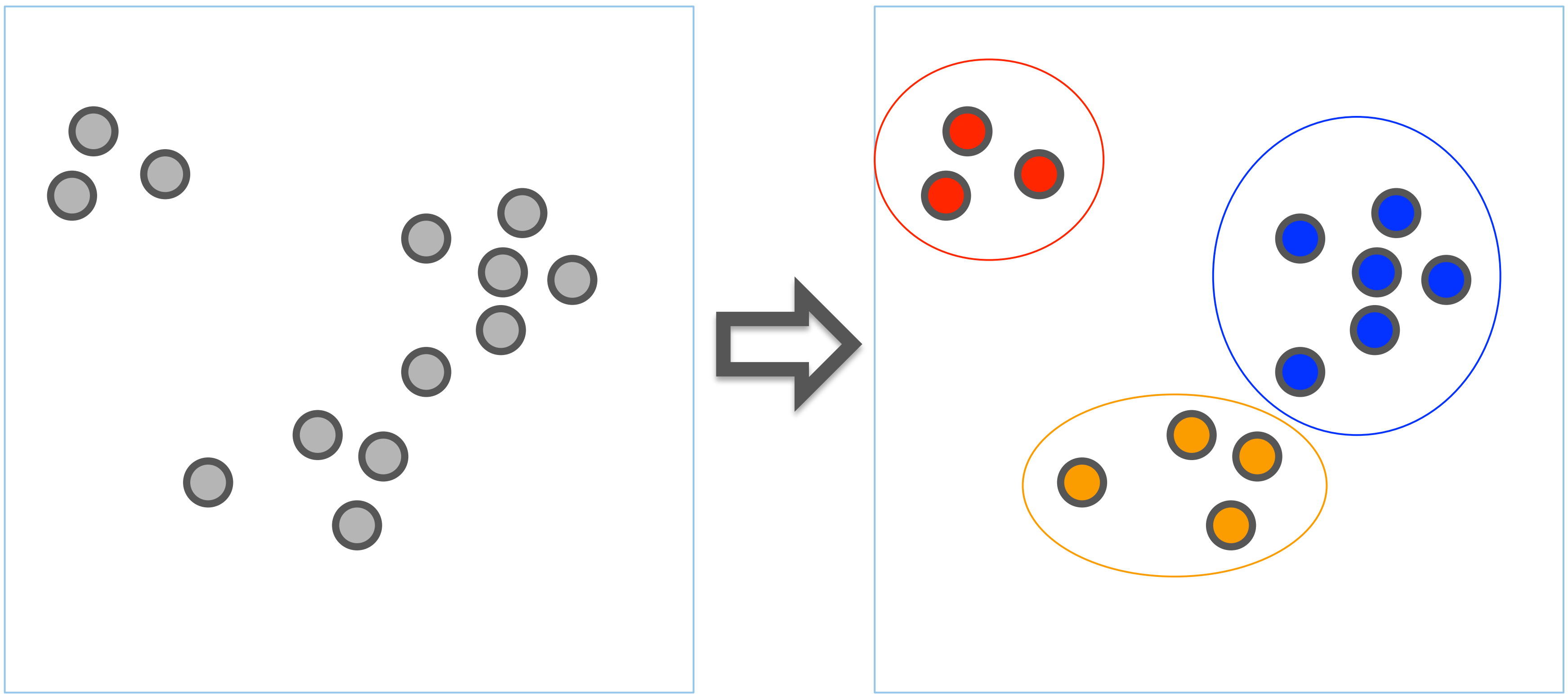
- Use Cases
- Distributed ML Challenges
- Code Examples

Ongoing Development

Clustering with K-Means

Given data points

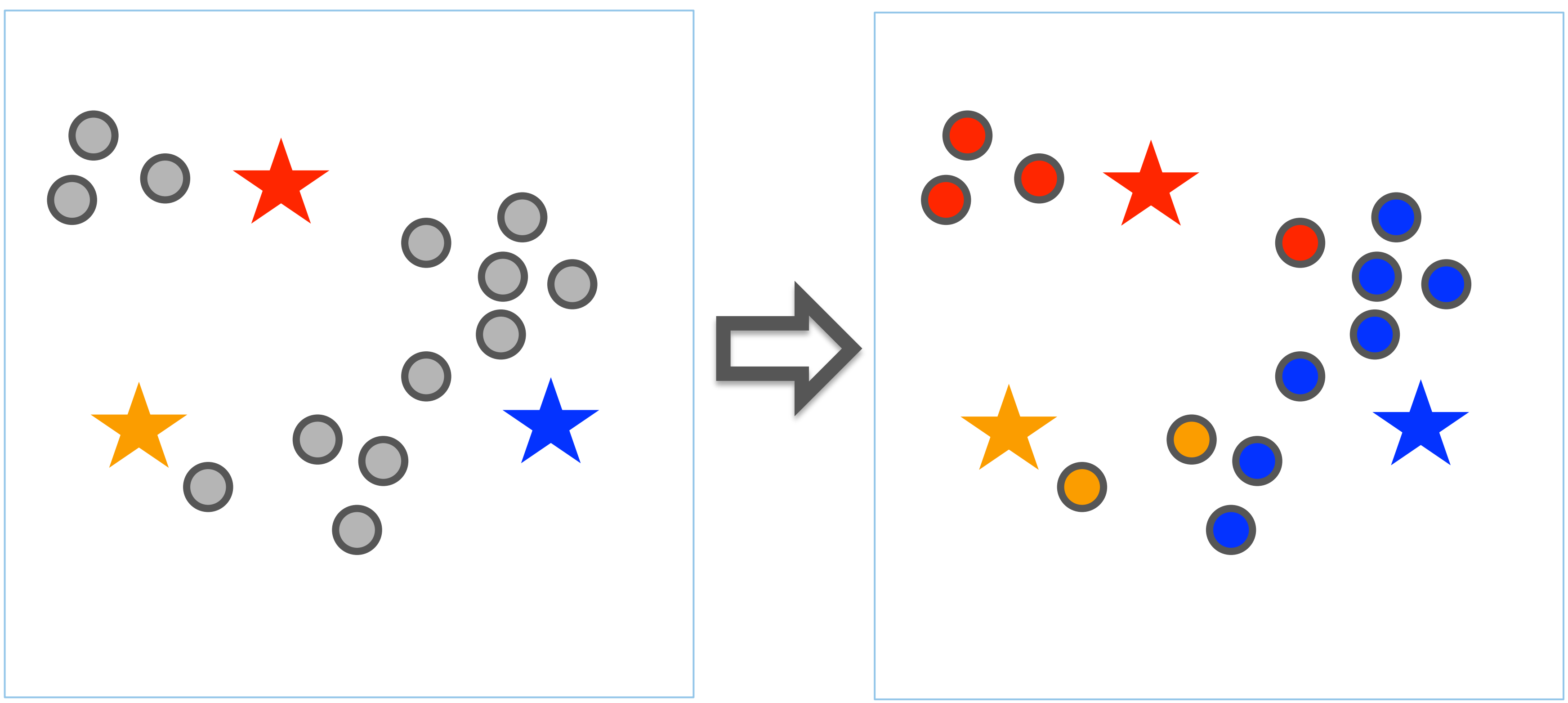
Find meaningful clusters



Clustering with K-Means

Choose cluster centers

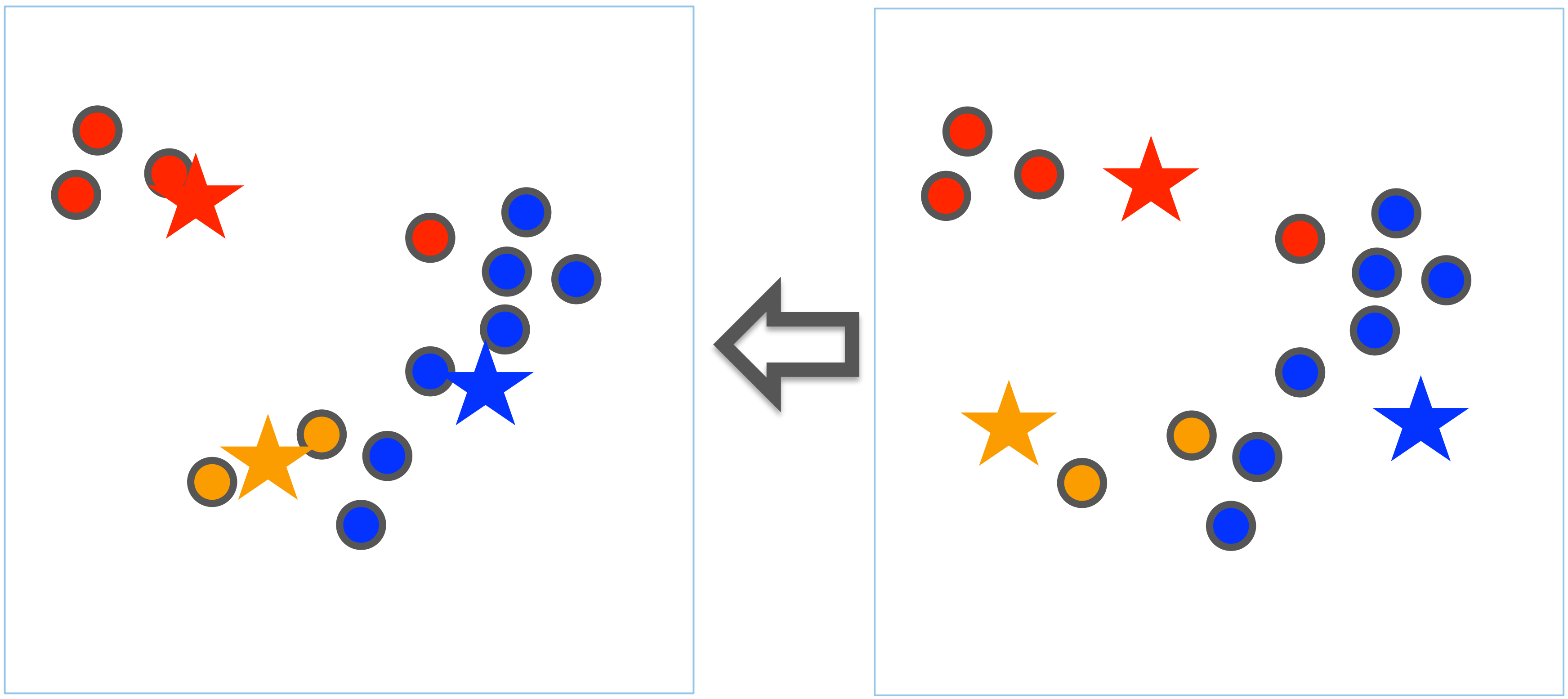
Assign points to clusters



Clustering with K-Means

Choose cluster centers

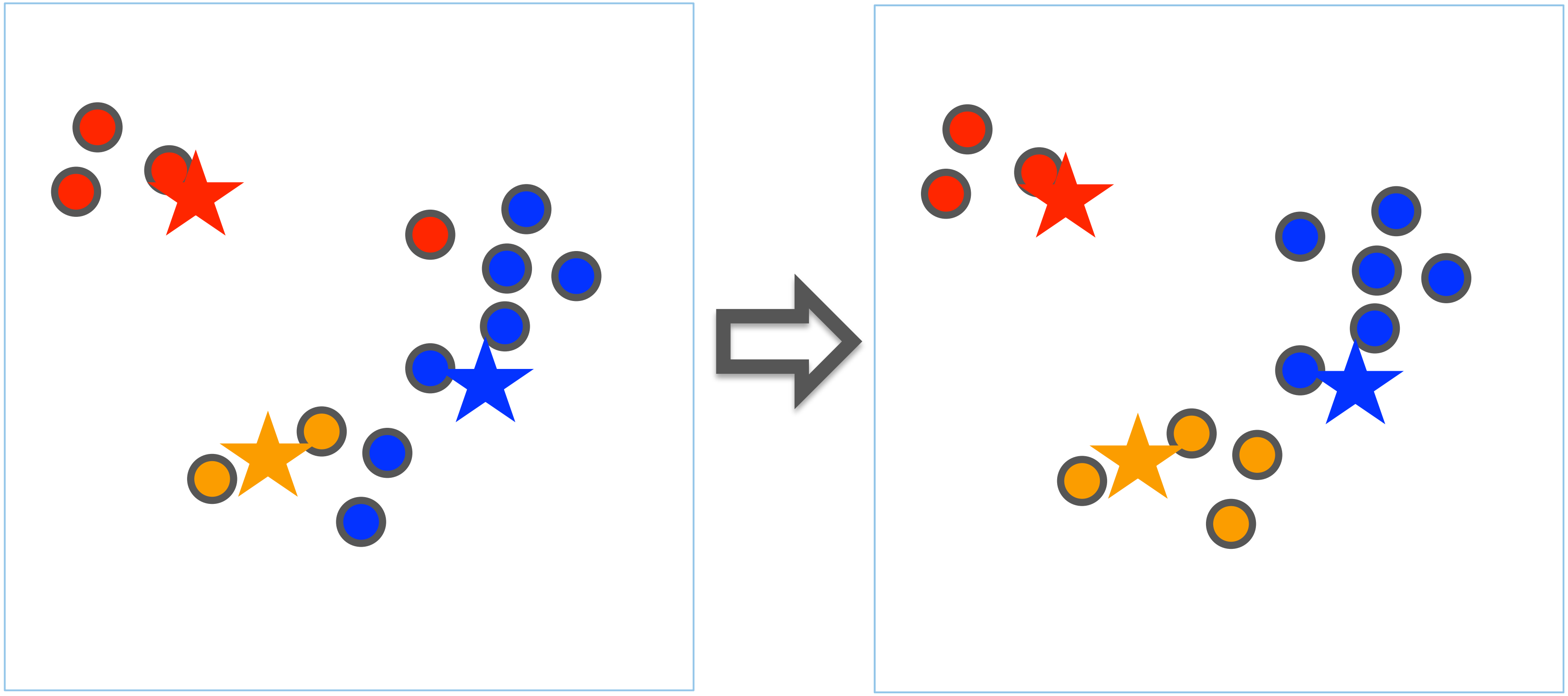
Assign points to clusters



Clustering with K-Means

Choose cluster centers

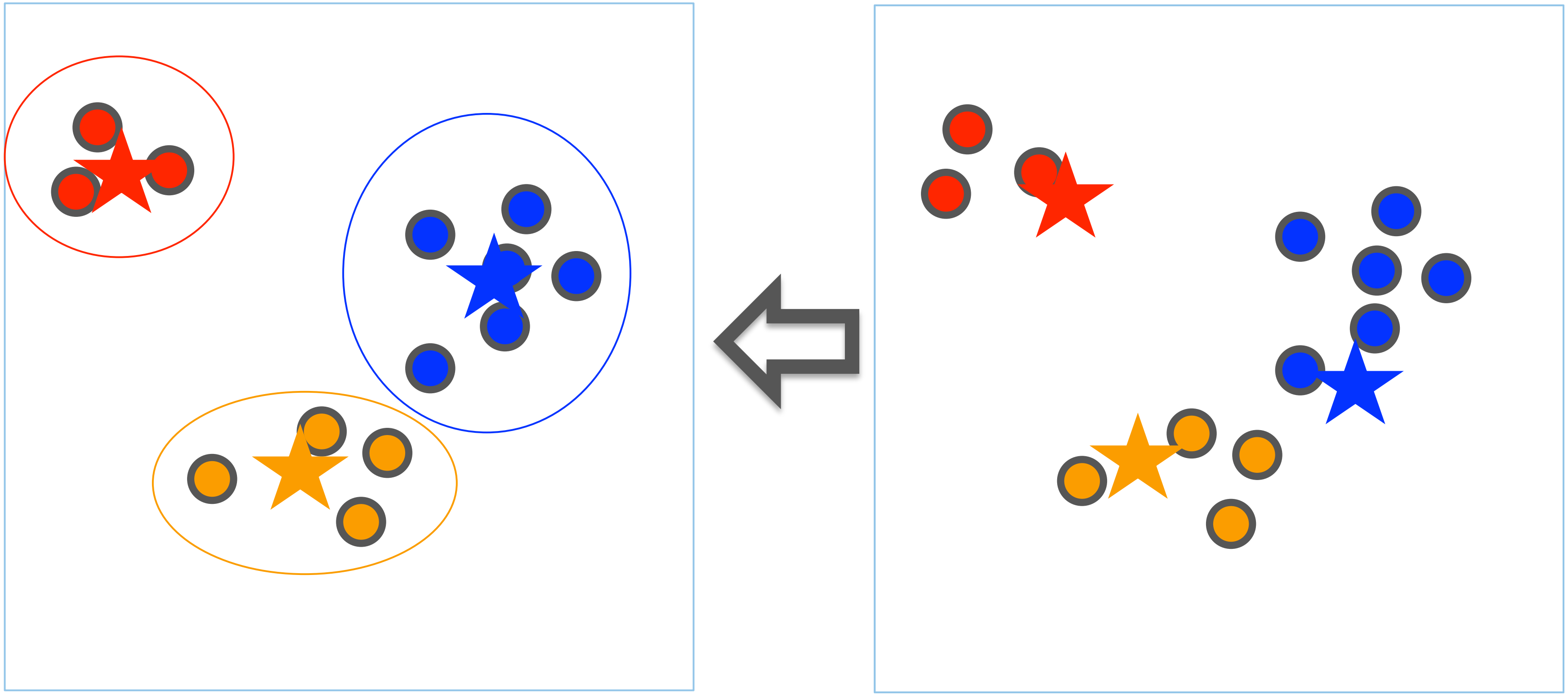
Assign points to clusters



Clustering with K-Means

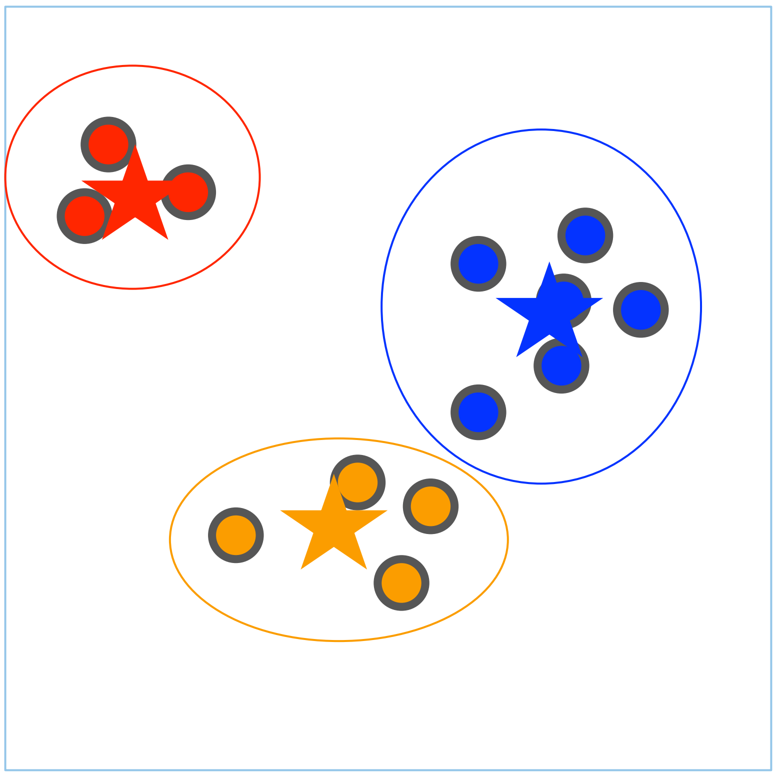
Choose cluster centers

Assign points to clusters



Clustering with K-Means

Data distributed by instance (point/row)



Smart initialization

Limited communication
(# clusters << # instances)

K-Means: Scala

```
// Load and parse data.
val data = sc.textFile("kmeans_data.txt")
val parsedData = data.map { x =>
   Vectors.dense(x.split(' ').map(_.toDouble))
}.cache()

// Cluster data into 5 classes using KMeans.
val clusters = KMeans.train(
   parsedData, k = 5, numIterations = 20)

// Evaluate clustering error.
val cost = clusters.computeCost(parsedData)
println("Sum of squared errors = " + cost)
```

K-Means: Python

```
# Load and parse data.
data = sc.textFile("kmeans_data.txt")
parsedData = data.map(lambda line:
    array([float(x) for x in line.split(' ‘)])).cache()

# Cluster data into 5 classes using KMeans.
clusters = KMeans.train(parsedData, k = 5, maxIterations = 20)

# Evaluate clustering error.
def error(point):
    center = clusters.centers[clusters.predict(point)]
    return sqrt(sum([x**2 for x in (point - center)]))

cost = parsedData.map(lambda point: error(point)) \
    .reduce(lambda x, y: x + y)
print("Sum of squared error = " + str(cost))
```

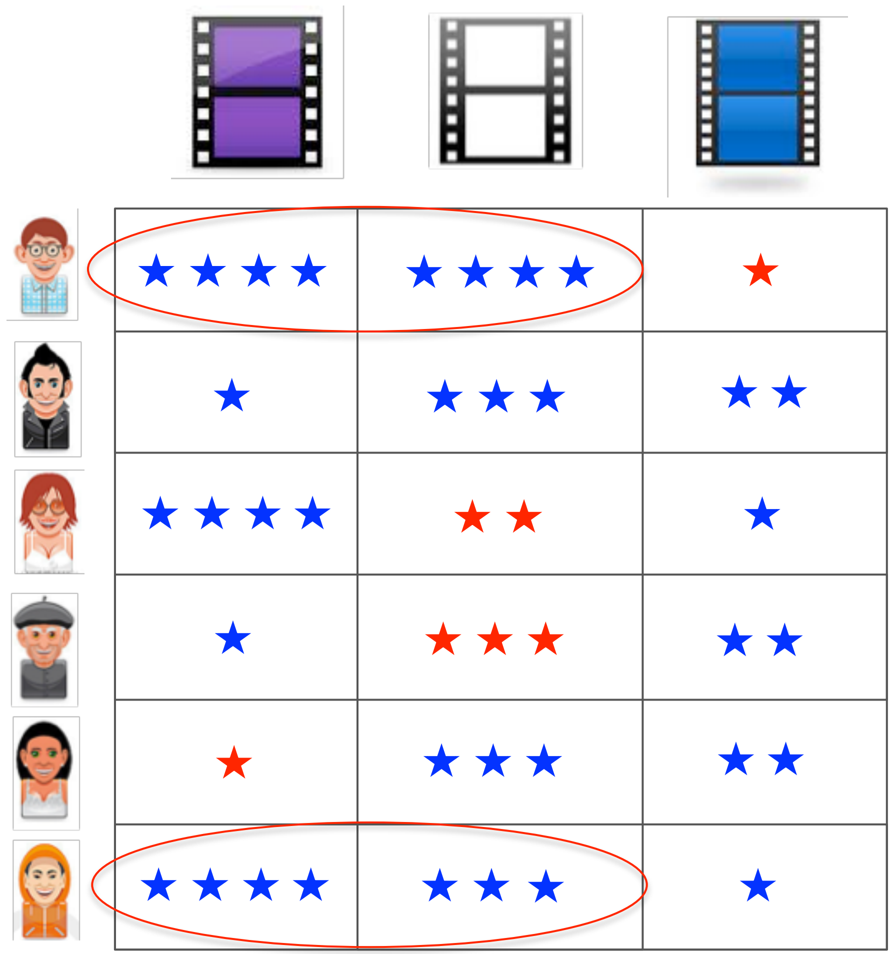
Recommendation

★★★★	★★★★	?
★	★★★	★★
★★★★		★
★		★★
	★★★	★★
★★★★	★★★	★

Goal: Recommend movies to users

Recommendation

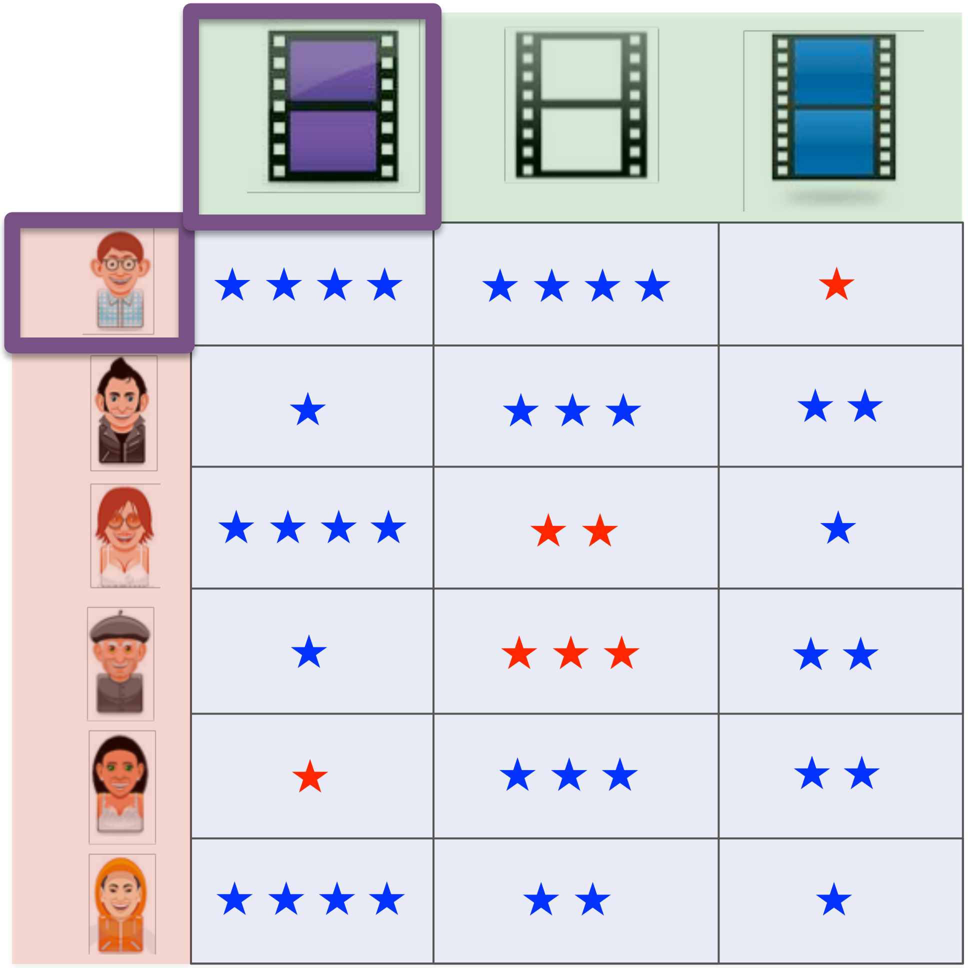
Collaborative filtering



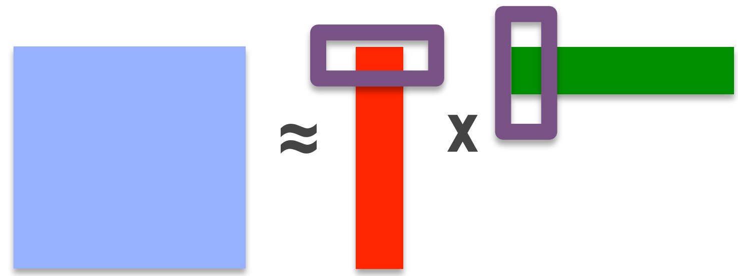
Goal: Recommend movies to users

Challenges:
- Defining similarity
- Dimensionality
  Millions of Users / Items
- Sparsity

Recommendation



Solution: Assume ratings are determined by a small number of factors.



25M Users, 100K Movies

→ 2.5 trillion ratings

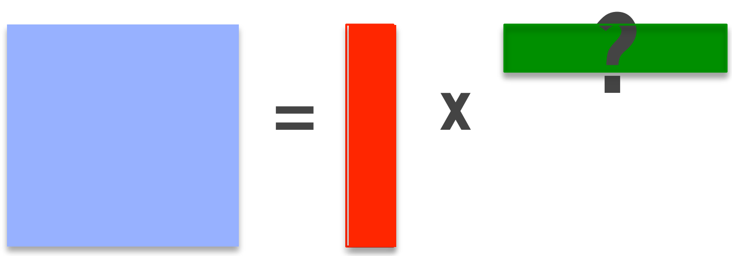
With 10 factors/user

→ 250M parameters

Recommendation with Alternating Least Squares (ALS)

<u>Algorithm</u>

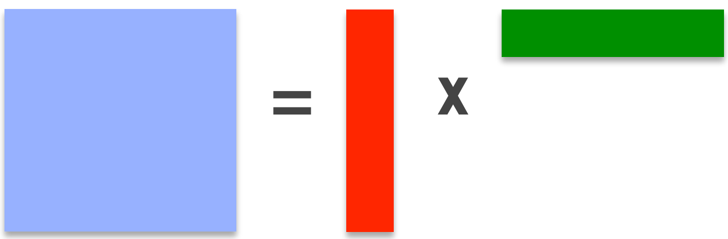
Alternating update of user/movie factors



Recommendation with Alternating Least Squares (ALS)

<u>Algorithm</u>

Alternating update of user/movie factors



Can update factors in parallel

Must be careful about communication

Recommendation with Alternating Least Squares (ALS)

```
// Load and parse the data
val data = sc.textFile("mllib/data/als/test.data")
val ratings = data.map(_.split(',') match {
    case Array(user, item, rate) =>
      Rating(user.toInt, item.toInt, rate.toDouble)
})

// Build the recommendation model using ALS
val model = ALS.train(
    ratings, rank = 10, numIterations = 20, regularizer = 0.01)

// Evaluate the model on rating data
val usersProducts = ratings.map { case Rating(user, product, rate) =>
  (user, product)
}
val predictions = model.predict(usersProducts)
```

ALS: Today's ML Exercise

- Load 1M/10M ratings from MovieLens
- Specify YOUR ratings on examples
- Split examples into training/validation
- Fit a model (Python or Scala)
- Improve model via parameter tuning
- Get YOUR recommendations

History and Overview

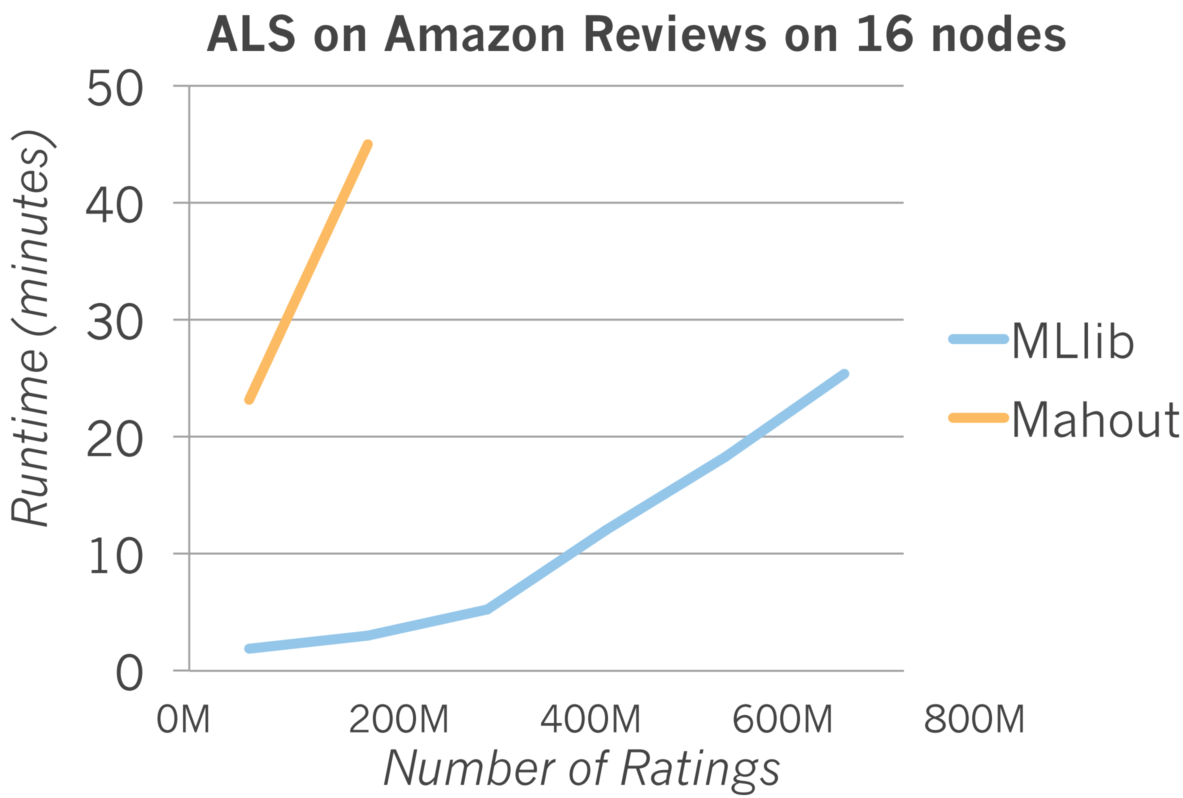
Example Applications

Ongoing Development

- Performance
- New APIs

Performance

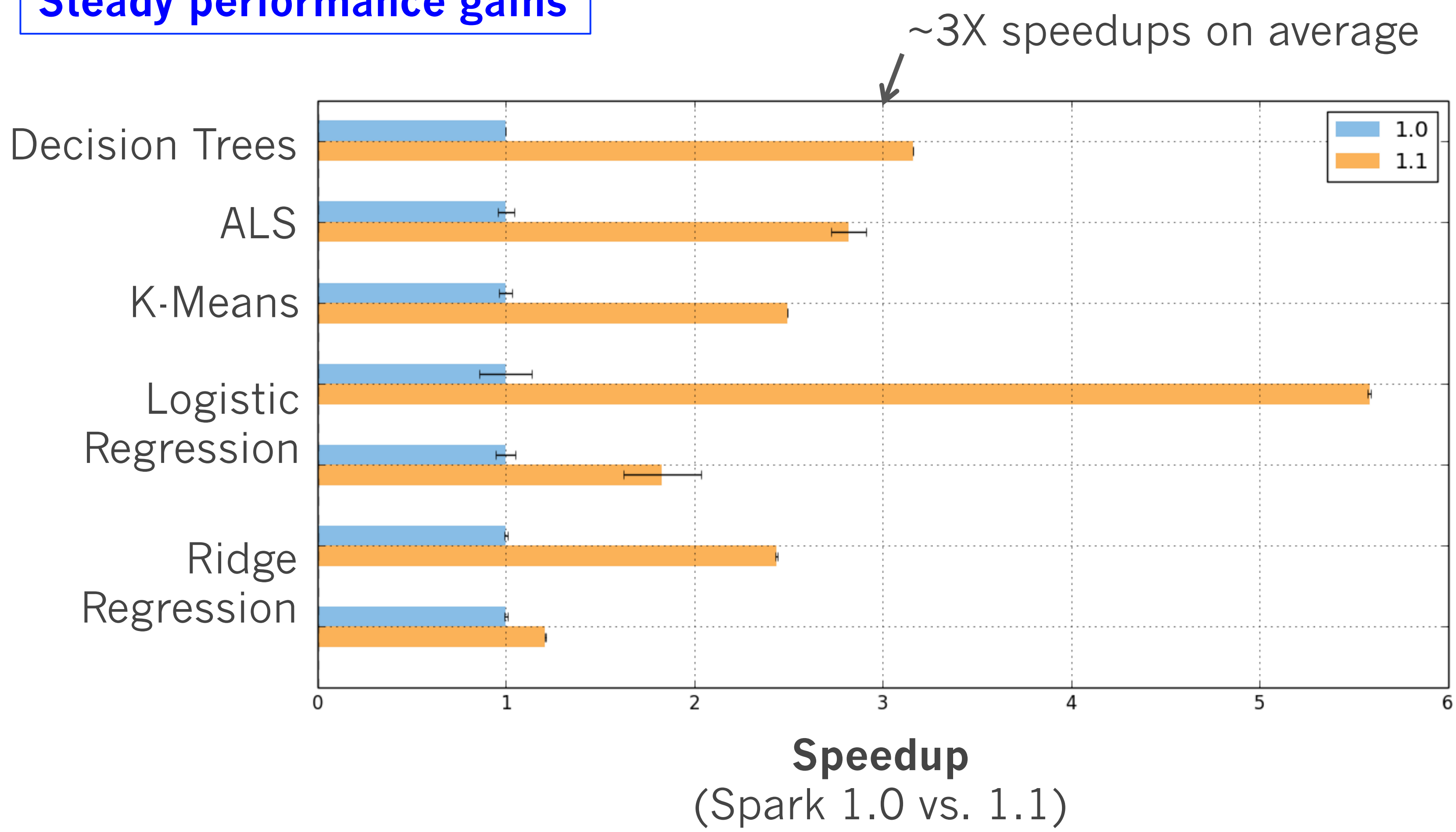
Spark: 10-100X faster than Hadoop & Mahout



On a dataset with 660M users, 2.4M items, and 3.5B ratings
MLlib runs in 40 minutes with 50 nodes

Performance

Steady performance gains



Algorithms

In Spark 1.2

- Random Forests: ensembles of Decision Trees
- Boosting

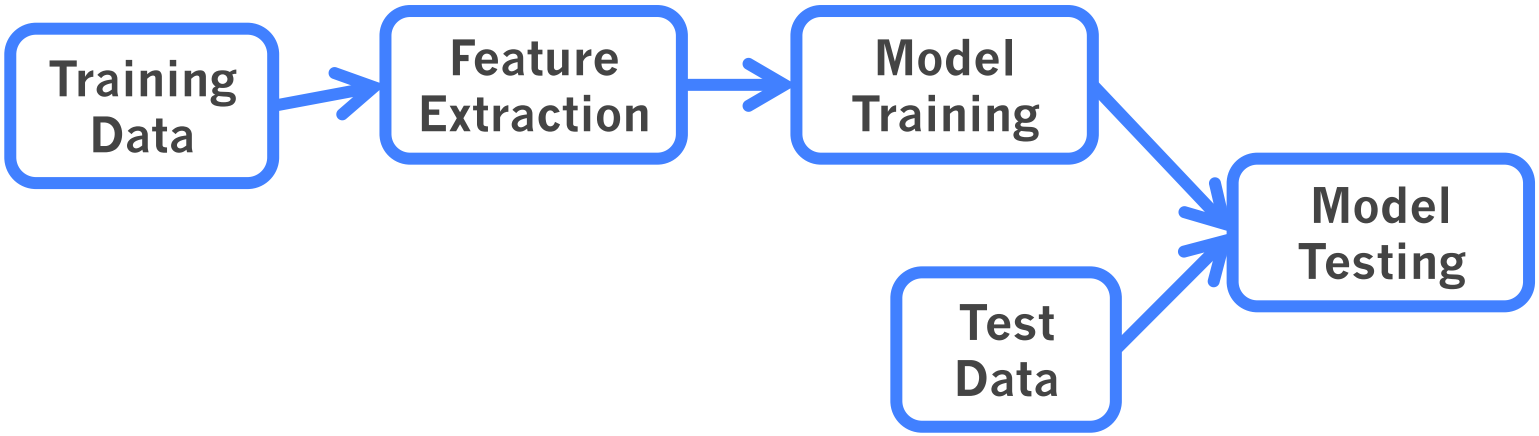
Under development

- Topic modeling
- (many others)

Many others!

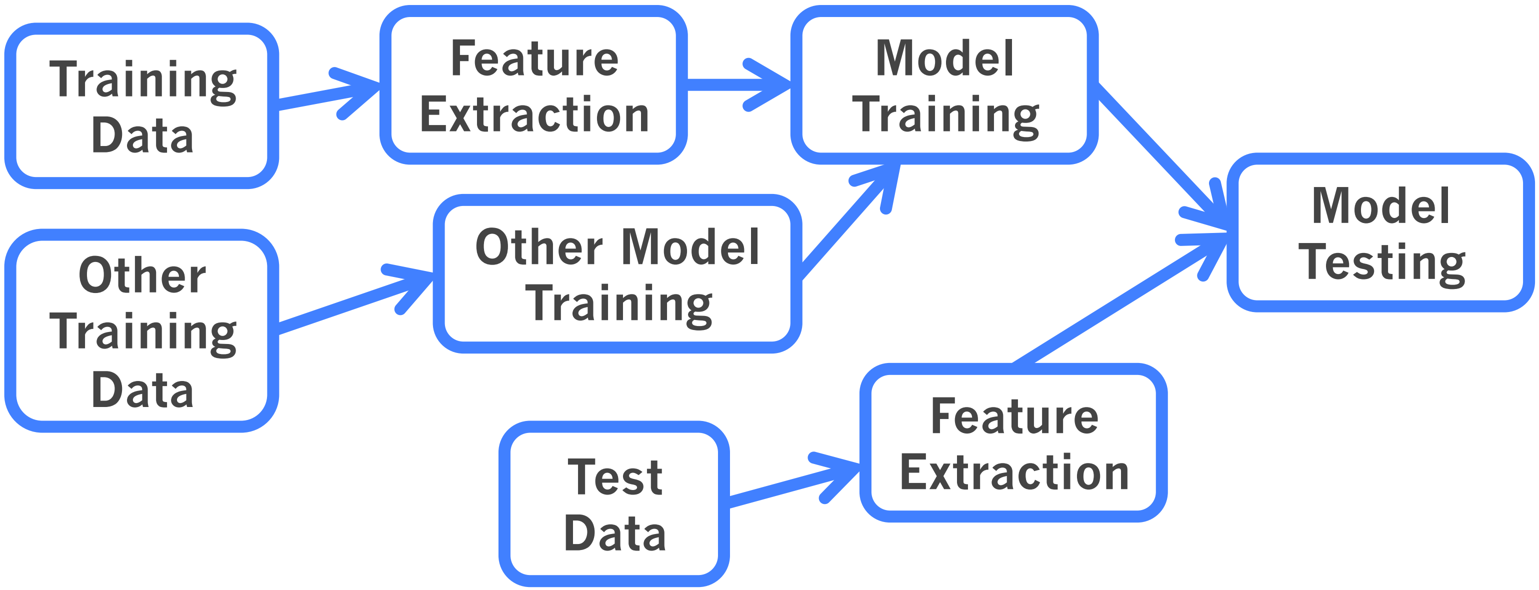
ML Pipelines

Typical ML workflow



ML Pipelines

Typical ML workflow *is complex.*



ML Pipelines

Typical ML workflow *is complex.*

Pipelines in 1.2 (alpha release)

- Easy workflow construction
- Standardized interface for model tuning
- Testing & failing early

Inspired by MLbase / Pipelines Project (see Evan's talk)

Collaboration with Databricks

MLbase / MLOpt aims to autotune these pipelines

Datasets

ML pipelines require *Datasets*

- Handle many data types (features)
- Keep metadata about features
- Select subsets of features for different parts of pipeline
- Join groups of features

ML Dataset = SchemaRDD

Further Integration with SparkSQL

Inspired by MLbase / MLI API

Resources

MLlib Programming Guide
spark.apache.org/docs/latest/mllib-guide.html

Databricks training info
databricks.com/spark-training

Spark user lists & community
spark.apache.org/community.html

edX MOOC on Scalable Machine Learning
www.edx.org/course/uc-berkeleyx/uc-berkeleyx-cs190-1x-scalable-machine-6066

4-day BIDS minicourse in January