MLbase: A System for Distributed Machine Learning

Ameet Talwalkar

amplab UC Berkeley
databricks™
UCLA

Problem: Scalable implementations *difficult* for ML Developers…

MATLAB® The Language of Technical Computing

CHALLENGE: Can we simplify distributed ML development?

Problem: ML is *difficult* for End Users…

Too many algorithms

Too many knobs…

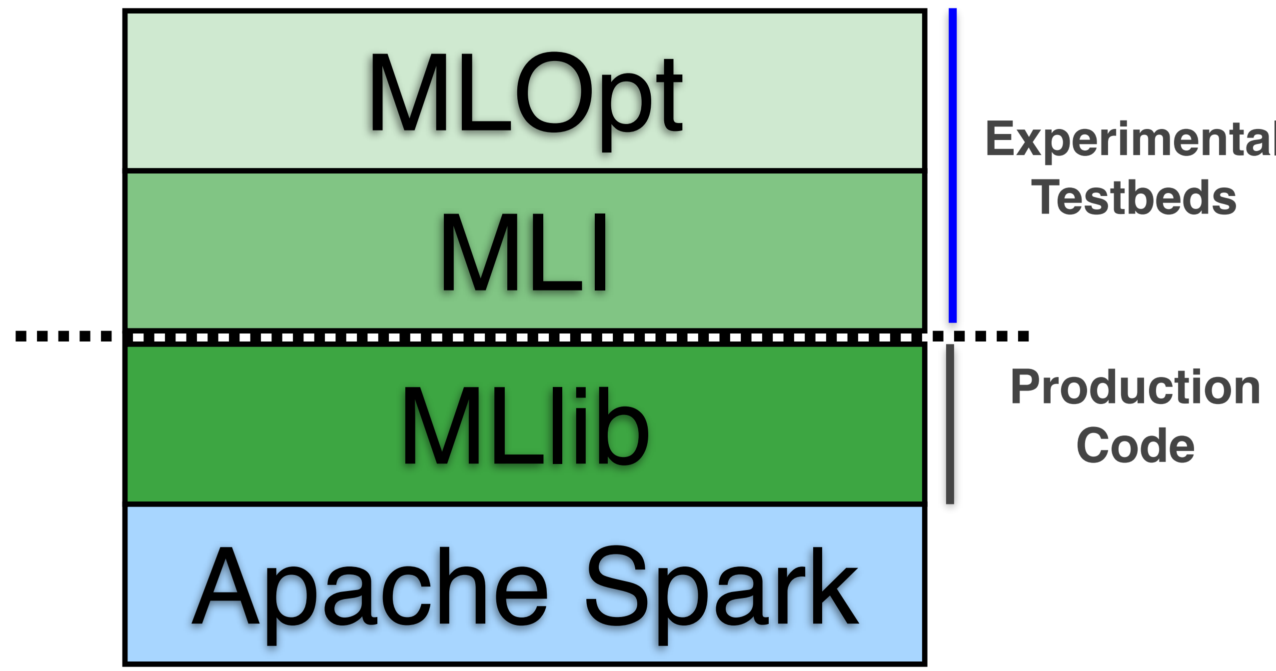
Too many ways to preprocess…

Difficult to debug

CHALLENGE: Can we automate ML pipeline construction?

MLbase

MLbase aims to simplify development and deployment of scalable ML pipelines



Spark: Cluster computing system designed for iterative computation (most active project in Apache Software Foundation)

MLlib: Spark's core ML library

MLI: API to simplify ML development

MLOpt: Declarative layer to automate hyperparameter tuning

Vision

MLlib / MLI

MLOpt

History of MLlib

Initial Release

- Developed by MLbase team in AMPLab
- 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

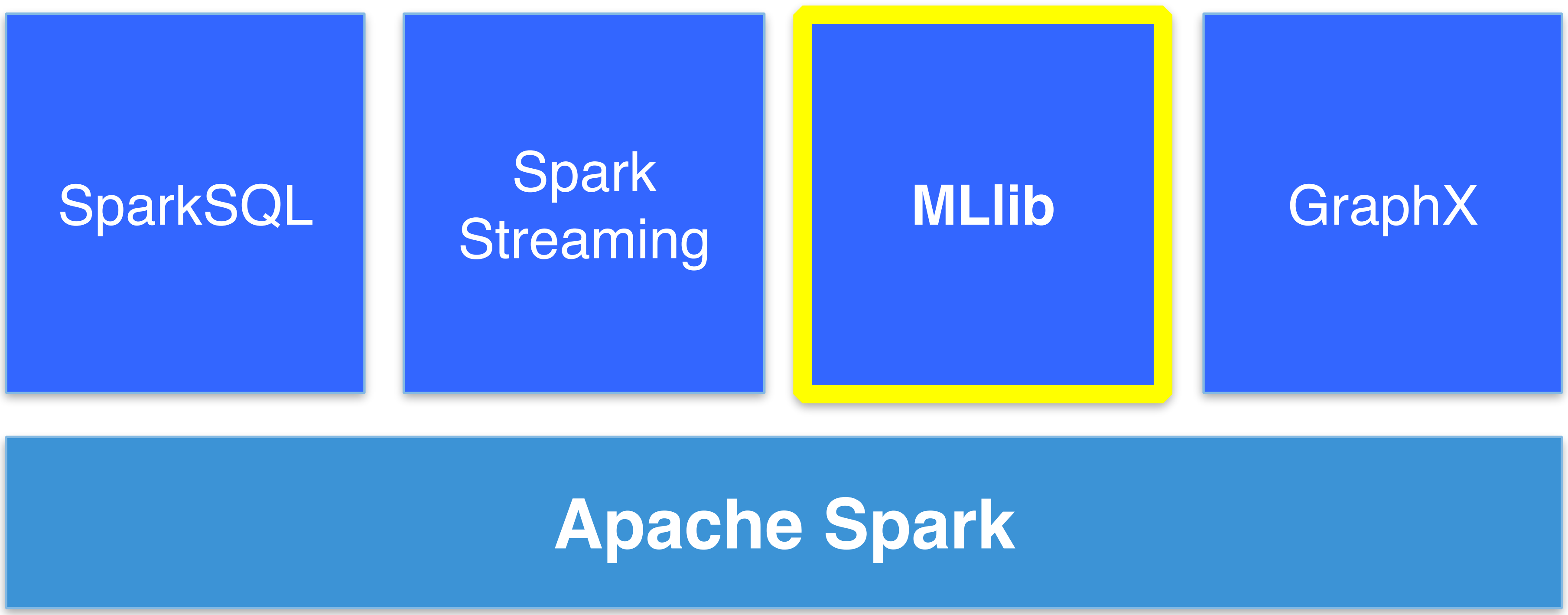
Clustering

Optimization Primitives

Many Utilities

Benefits of MLlib

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

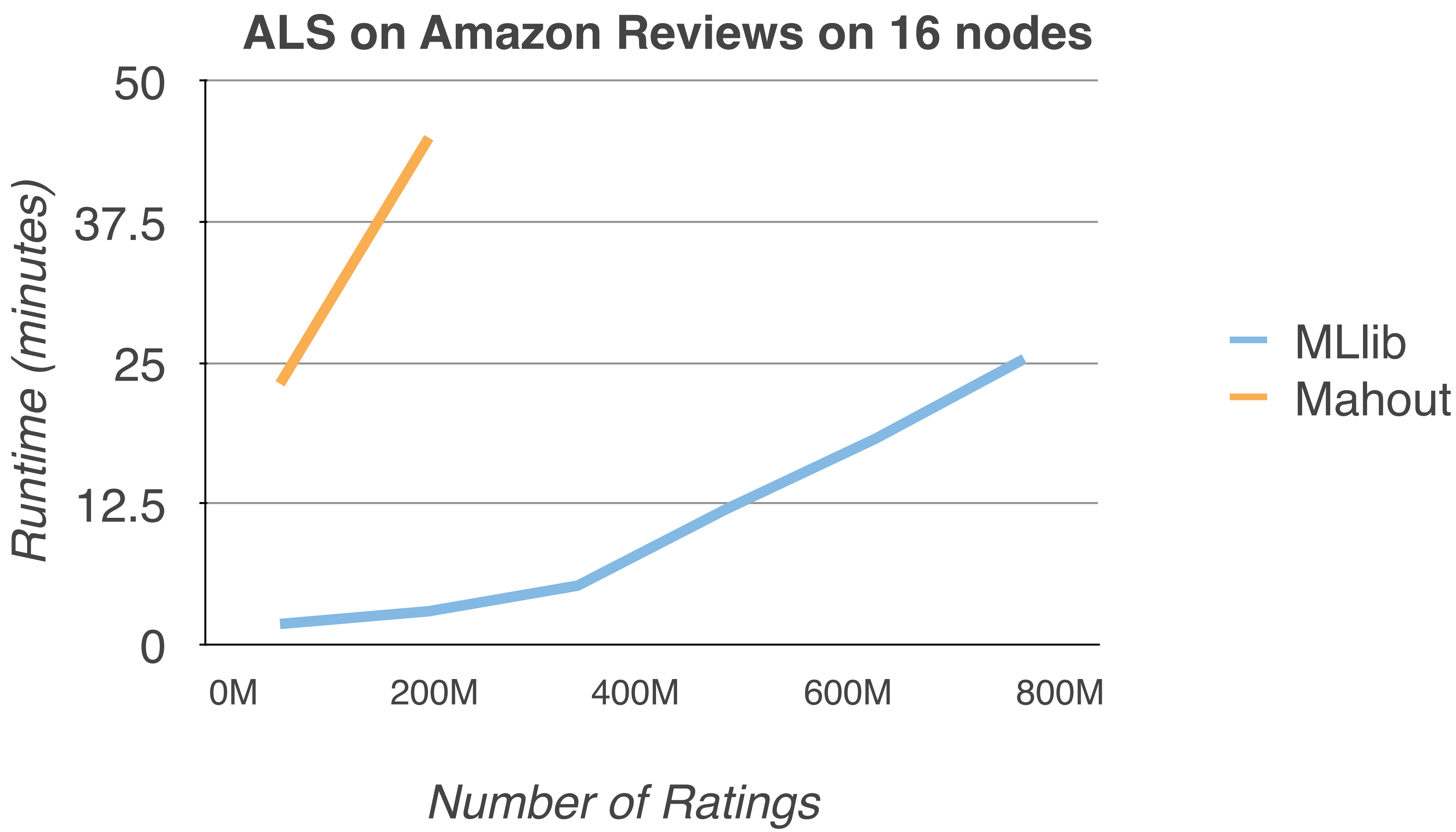


Benefits of MLlib

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

Performance

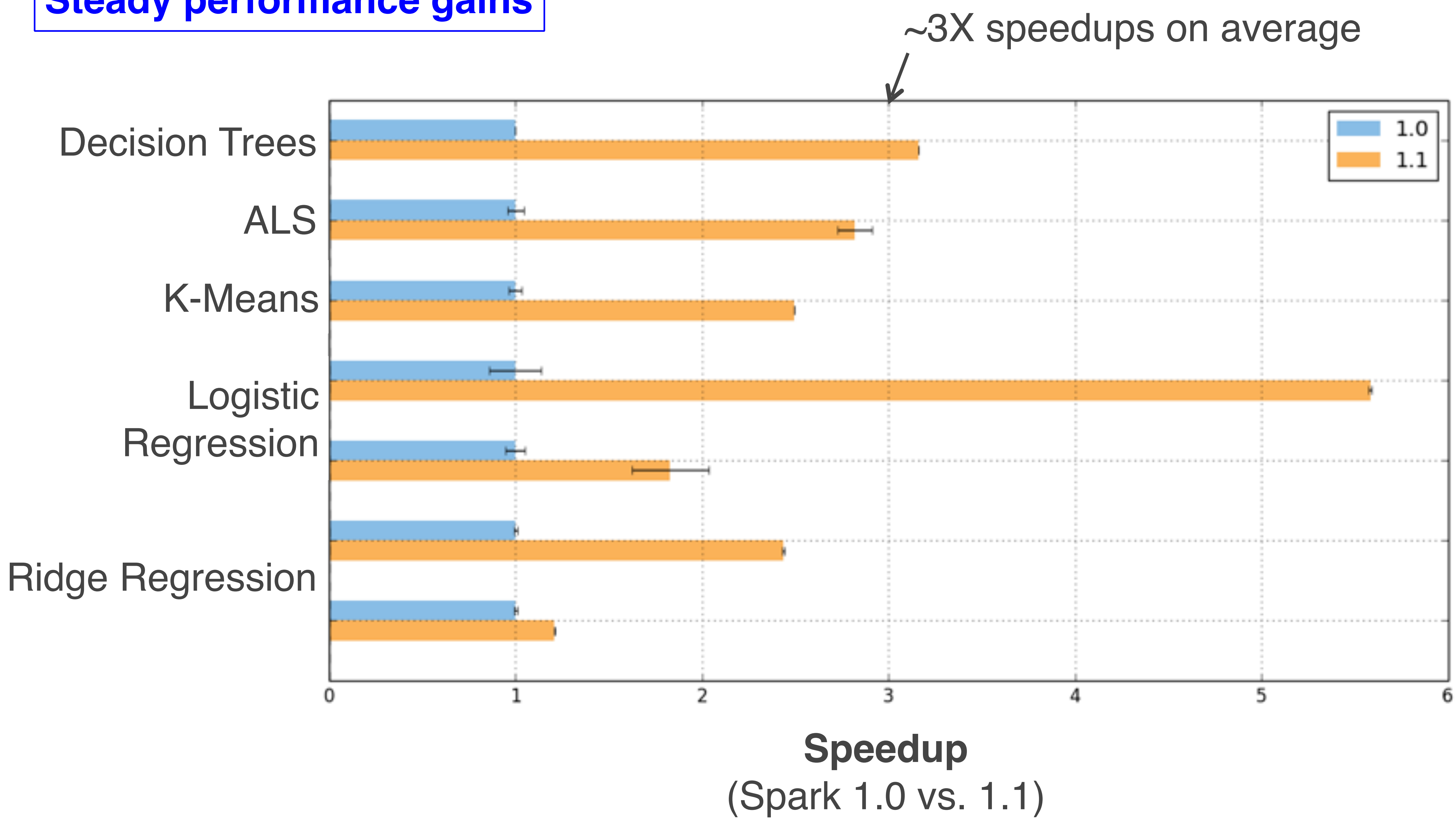
Spark: 10-100X faster than Hadoop & Mahout



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



ML Developer API (MLI)

- Shield ML Developers from low-details
  - Provide familiar mathematical operators in distributed setting
  - Standard APIs defining ML algorithms and feature extractors
- Tables
  - Flexibility when loading data
  - Common interface for feature extraction / algorithms

- Matrices
  - Linear algebra (on local partitions at first)
  - Sparse and Dense matrix support

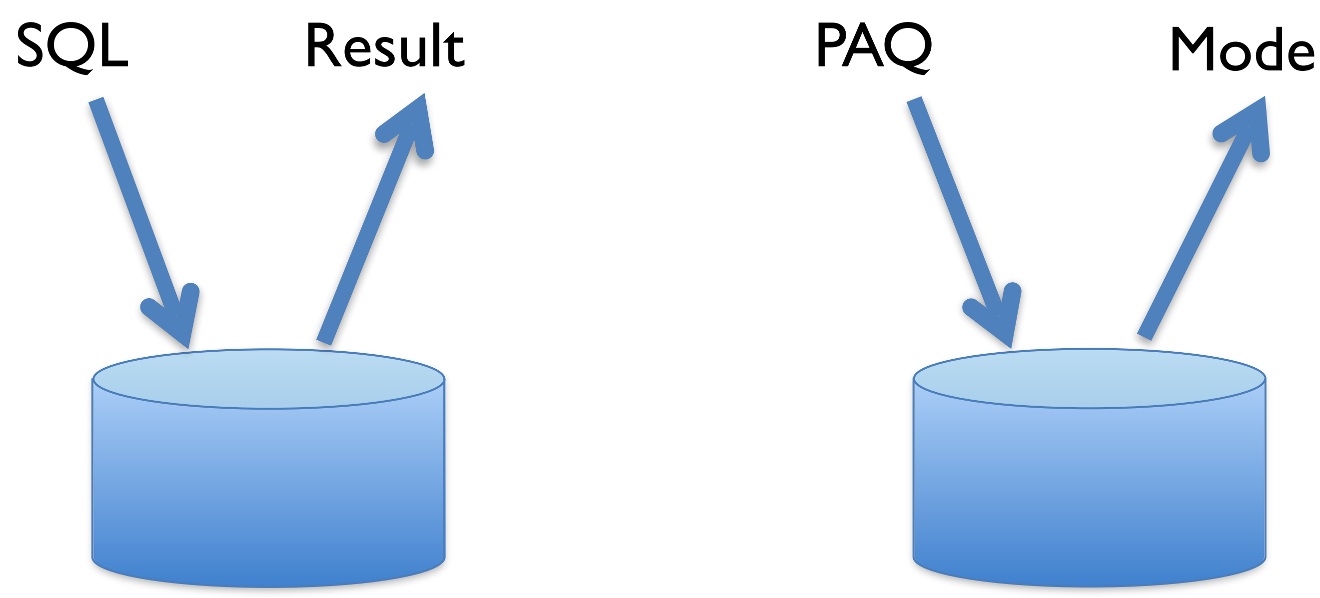
- Optimization Primitives
  - Distributed implementations of common patterns

MLI, MLlib and Roadmap

- MLlib incorporate ideas from MLI
  - Matrices and optimization primitives already in MLlib
  - Tables and ML API will be in next release
- Longer term for MLlib
  - Scalable implementations of standard ML methods and underlying optimization primitives
  - Further support for ML pipeline development (including hyper parameter tuning using ideas from MLOpt)

Feedback and Contributions Encouraged!

Vision
MLlib / MLI
MLOpt

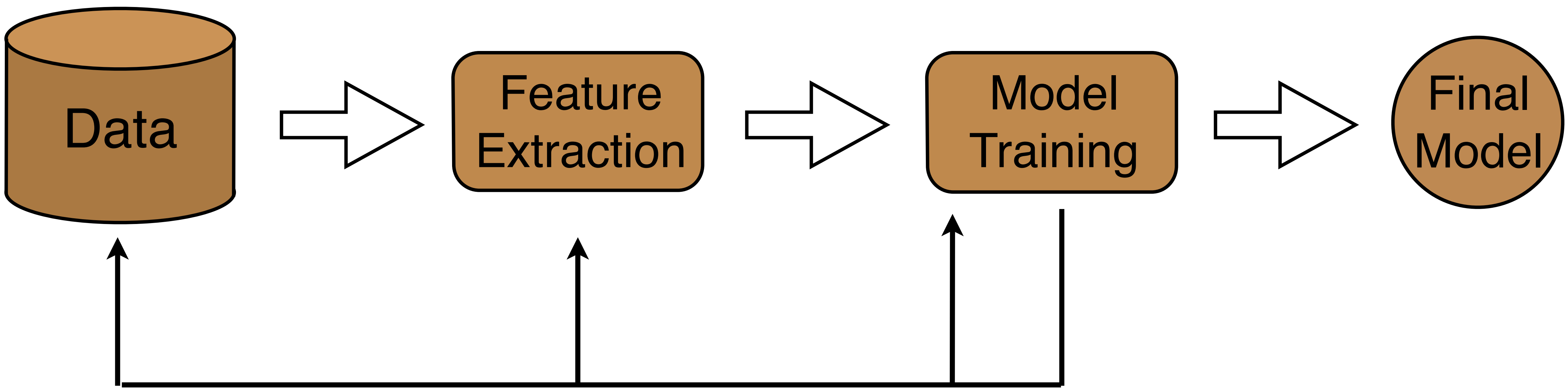


ML

- User declaratively specifies task
- PAQ = Predictive Analytic Query
- Search through MLlib to find the best model/pipeline

```
SELECT e.sender, e.subject, e.message
FROM Emails e
WHERE e.user = 'Bob'
AND PREDICT(e.spam, e.message) = false GIVEN
LabeledData
```

A Standard ML Pipeline

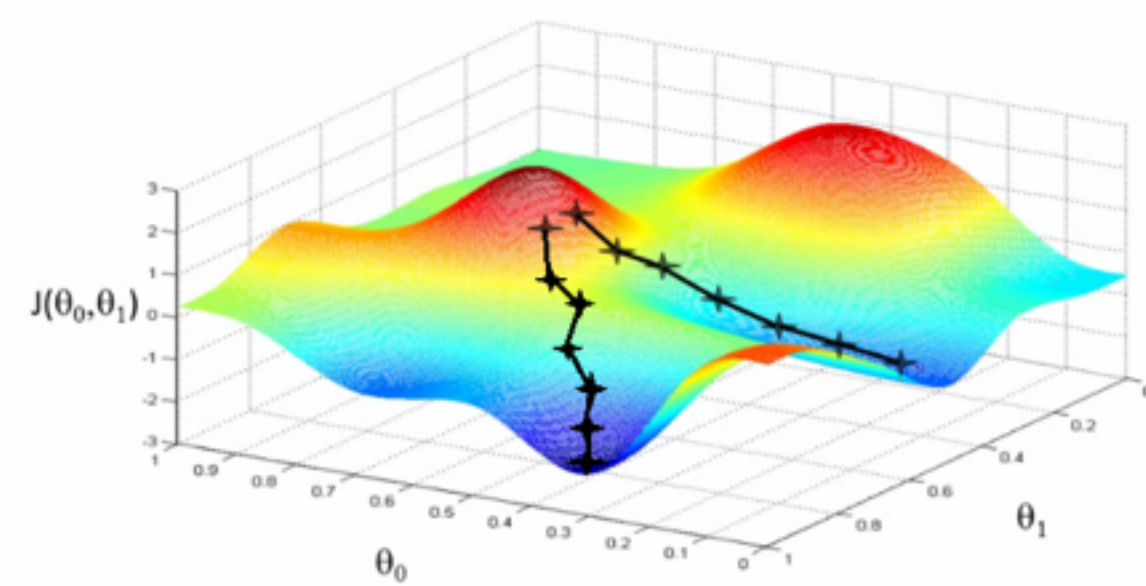


- In practice, model building is an iterative process of continuous refinement
- Our grand vision is to automate the construction of these pipelines

Training A Model

- Iteratively read through data
  - compute gradient
  - update model
  - repeat until converged
- Requires ***multiple passes***
- Common access pattern
  - ALS, Random Forests, etc.
- Minutes to train an SVM on 200GB of data on a 16-node cluster

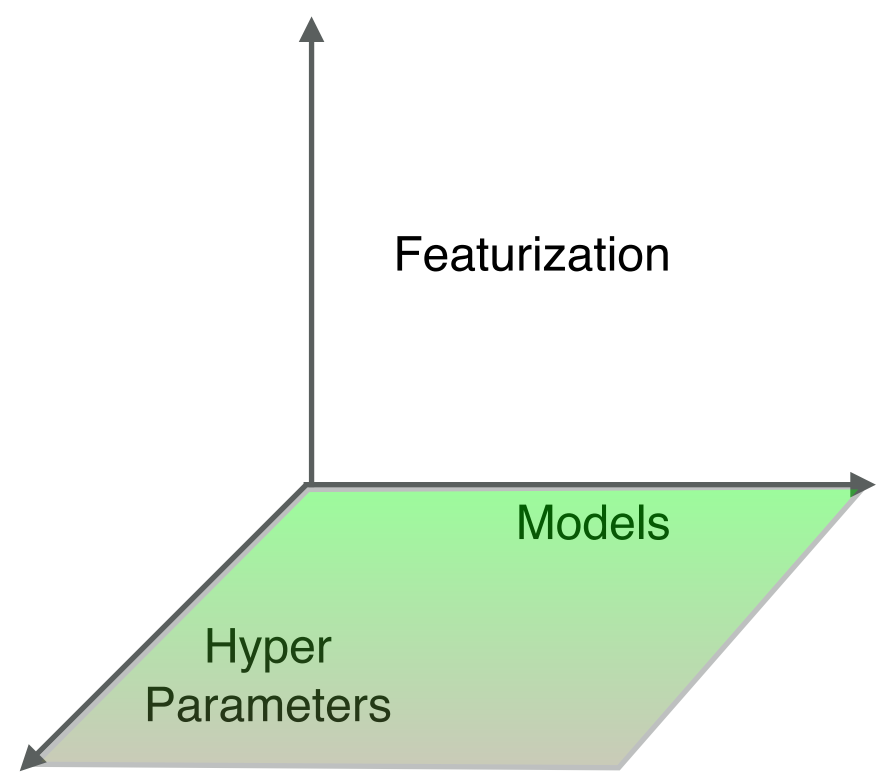
$$w := w - \alpha \nabla Q(w) = w - \alpha \sum_{i=1}^{n} \nabla Q_i(w),$$



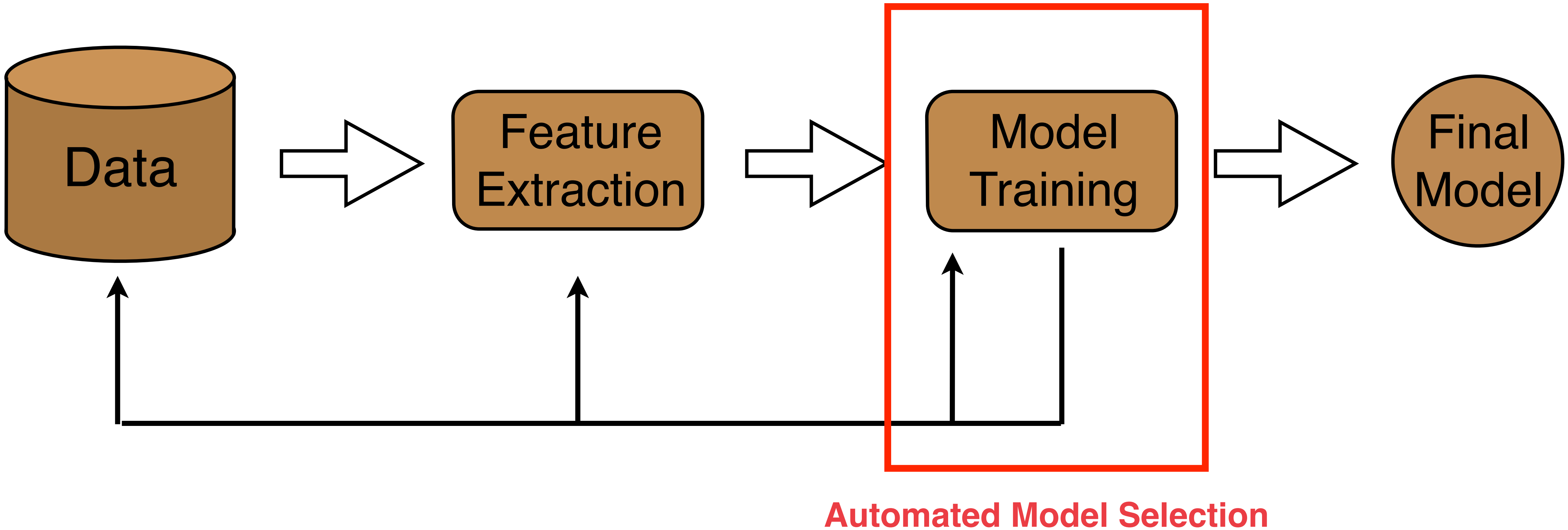
The Tricky Part

- Model
  - Logistic Regression, SVM, Tree-based, etc.
- Model hyper-parameters
  - Learning Rate, Regularization, etc.

- Featurization
  - Text: n-grams, TF-IDF
  - Images: Gabor filters, random convolutions
  - Random projection? Scaling?



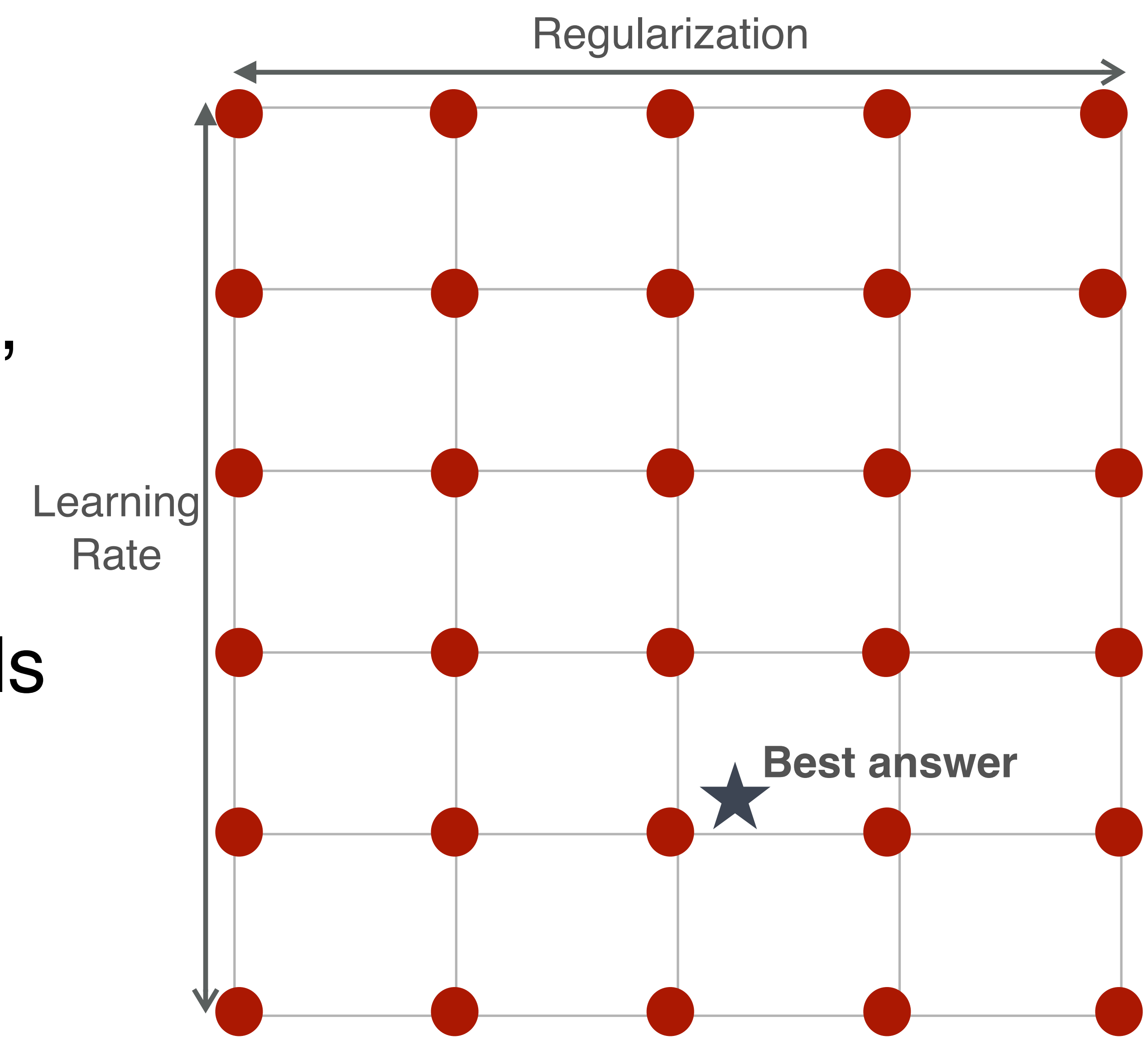
A Standard ML Pipeline



- In practice, model building is an iterative process of continuous refinement
- Our grand vision is to automate the construction of these pipelines
- Start with one aspect of the pipeline - model selection

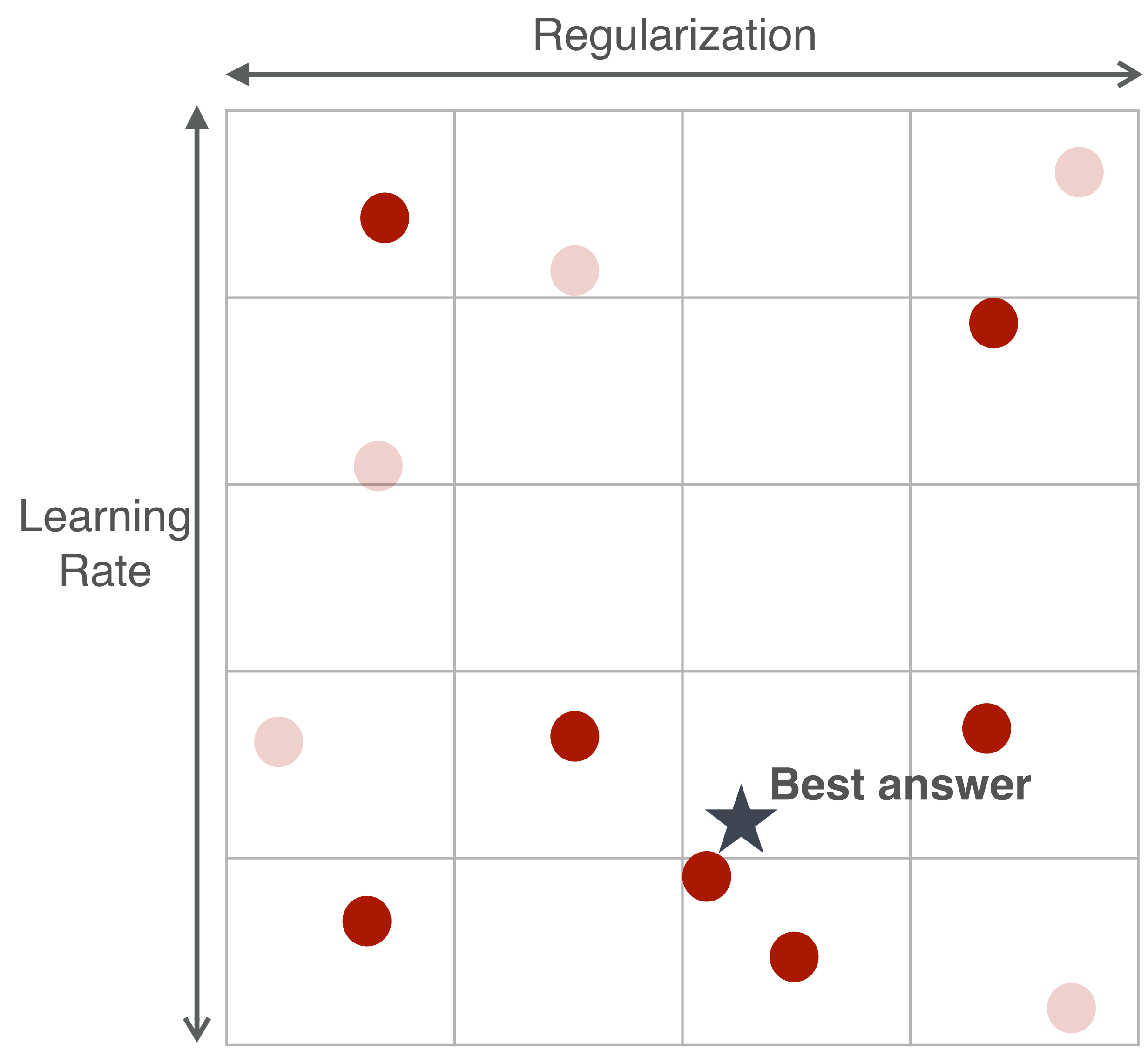
One Approach

- Sequential Grid Search
  - Search over all hyperparameters, algorithms, features, etc.
- Drawbacks
  - Expensive to compute models
  - Hyperparameter space is large
- Common in practice!



A Better Approach

- Better resource utilization
  - through batching
- Early Stopping
- Improved Search



A Tale Of 3 Optimizations

Better Resource Utilization

Early Stopping

Improved Search

Better Resource Utilization

- Typical model update requires 2-4 flops/double
- But modern memory much slower than processors
  - We can do 25 flops / double read!
  - This equates to 6-8 model updates per double we read, assuming models fit in cache
- **Train multiple models simultaneously**




What Do We See In Spark?

- 2x and 5x increase in models trained/sec with batching

Batch Size \ D	100	500	1000	10000
1	1.00	1.00	1.00	1.00
2	1.91	1.50	1.51	1.38
5	4.05	2.53	1.93	1.36
10	5.31	3.40	2.37	1.14

What Do We See In Spark?

- These numbers are with vector-matrix multiplies

Batch Size \ D	100	500	1000	10000
1	1.00	1.00	1.00	1.00
2	1.91	1.50	1.51	1.38
5	4.05	2.53	1.93	1.36
10	5.31	3.40	2.37	1.14

What Do We See In Spark?

- These numbers are with vector-matrix multiplies

Batch Size \ D	100	500	1000	10000
1	1.00	1.00	1.00	1.00
2	1.91	1.50	1.51	1.38
5	4.05	2.53	1.93	1.36
10	5.31	3.40	2.37	1.14

- Can do better when rewriting in terms of matrix-matrix multiplies

Batch Size \ D	100	500	1000	10000
1	1.00	1.00	1.00	1.00
2	0.97	1.51	0.92	1.28
5	2.67	2.95	2.26	3.18
10	4.54	7.36	6.39	5.40

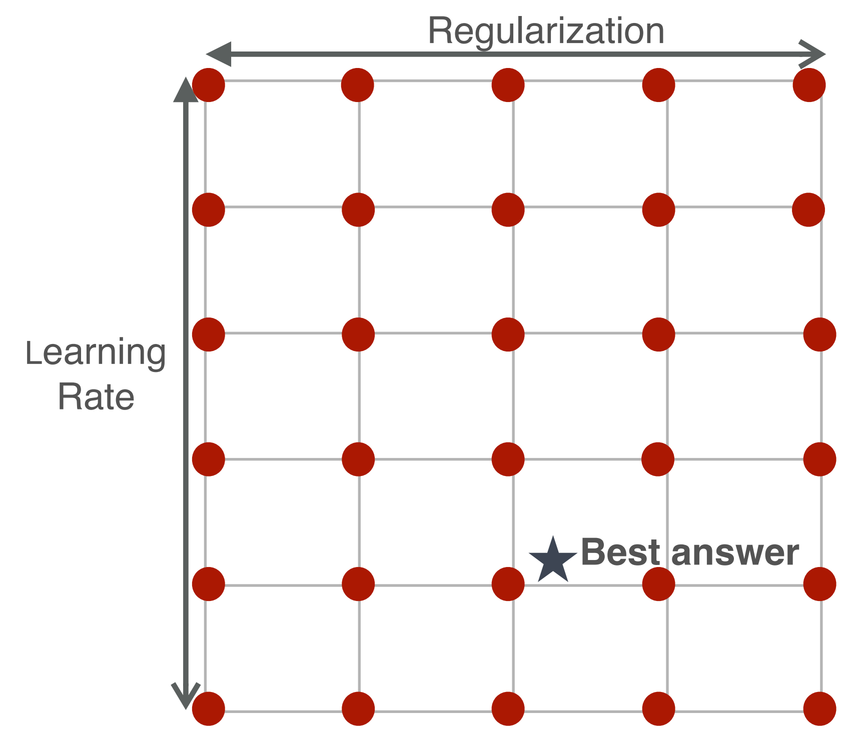
A Tale Of 3 Optimizations

Better Resource Utilization

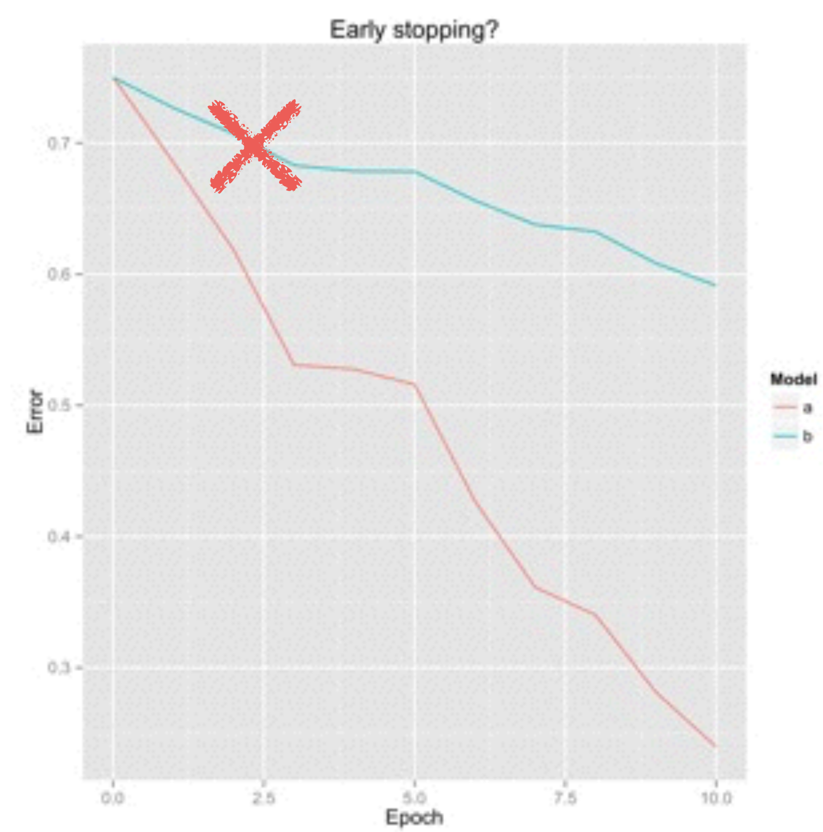
Early Stopping

Improved Search

Early Stopping

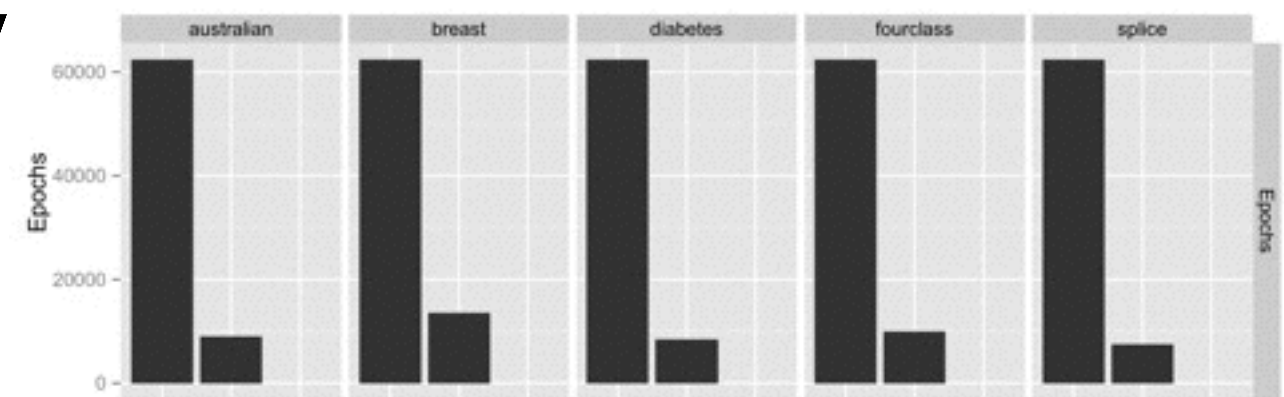


- Each point is a trained model
- Some models look bad early
  - So we give up early!
- So far a heuristic…
  - …but can be framed as a multi-armed bandit problem



Early Stopping

- Each point is a trained model
- Some models look bad early
  - So we give up early!
- So far a heuristic…
  - …but can be framed as a multi-armed bandit problem



A Tale Of 3 Optimizations

Better Resource Utilization

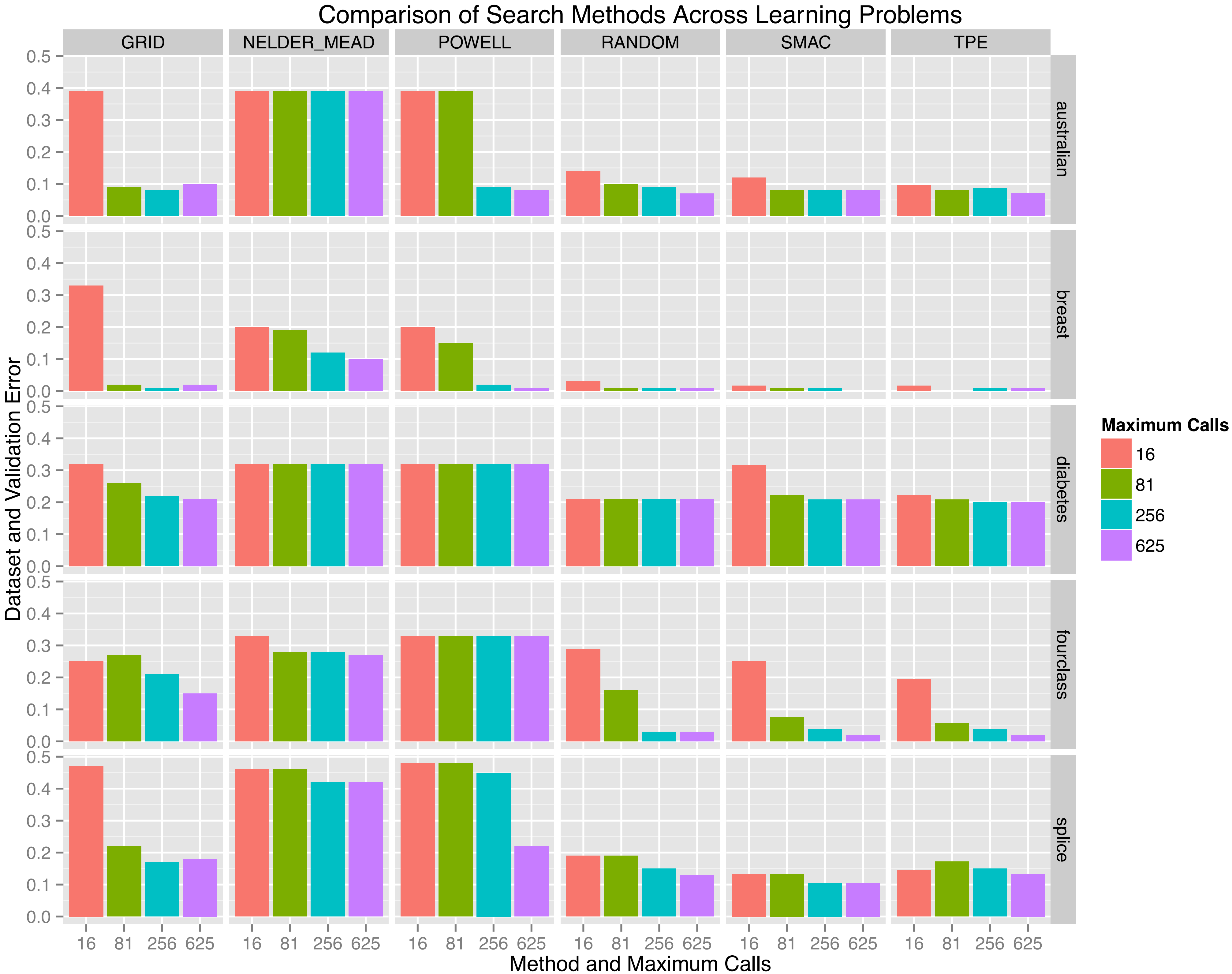
Algorithmic Speedups

Improved Search

What Search Method?

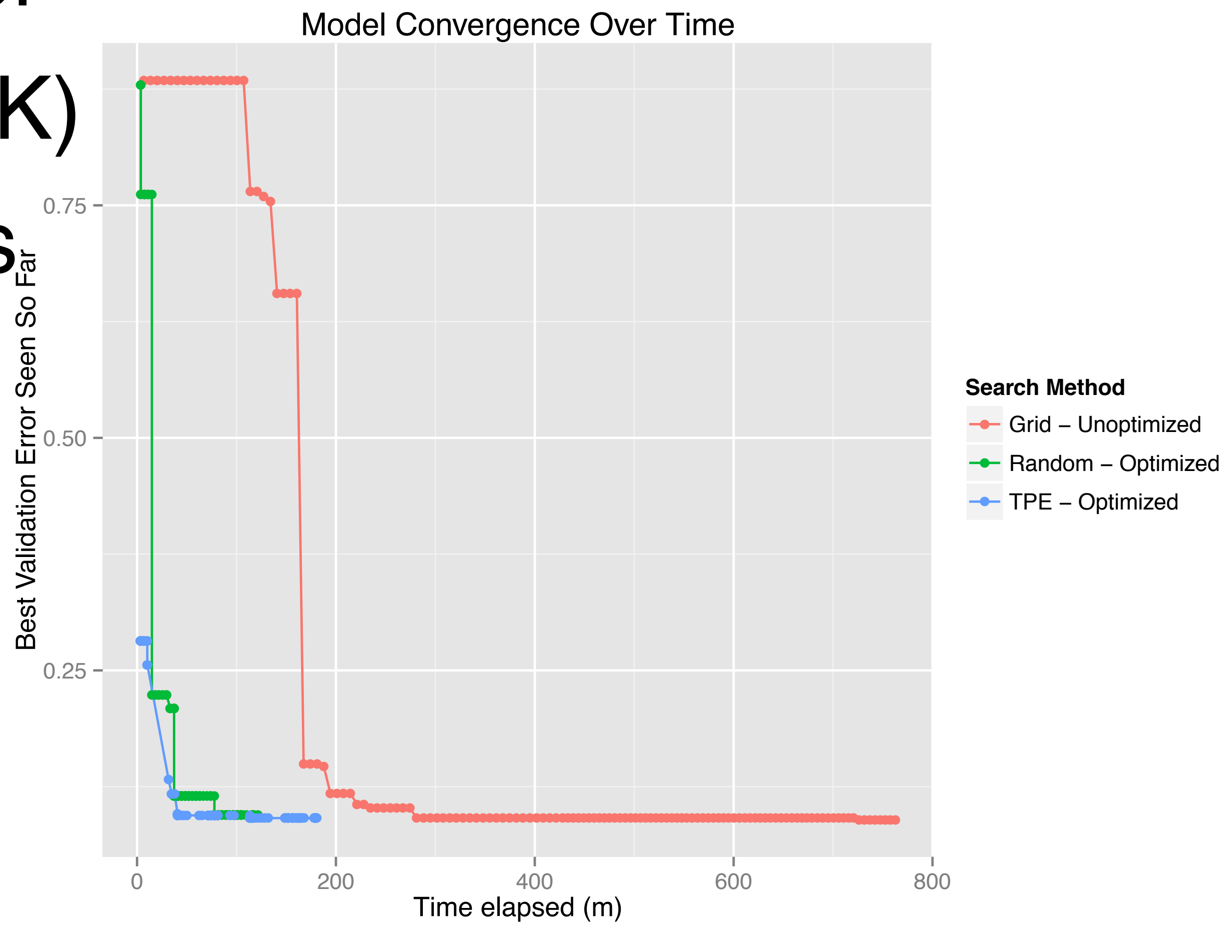
- Various derivative-free optimization techniques
  - Simple ones (Grid, Random)
  - Classic Derivative-Free (Nelder-Mead, Powell's method)
  - Bayesian (e.g., SMAC, TPE)
- What should we do?

What Search Method?



Putting It All Together

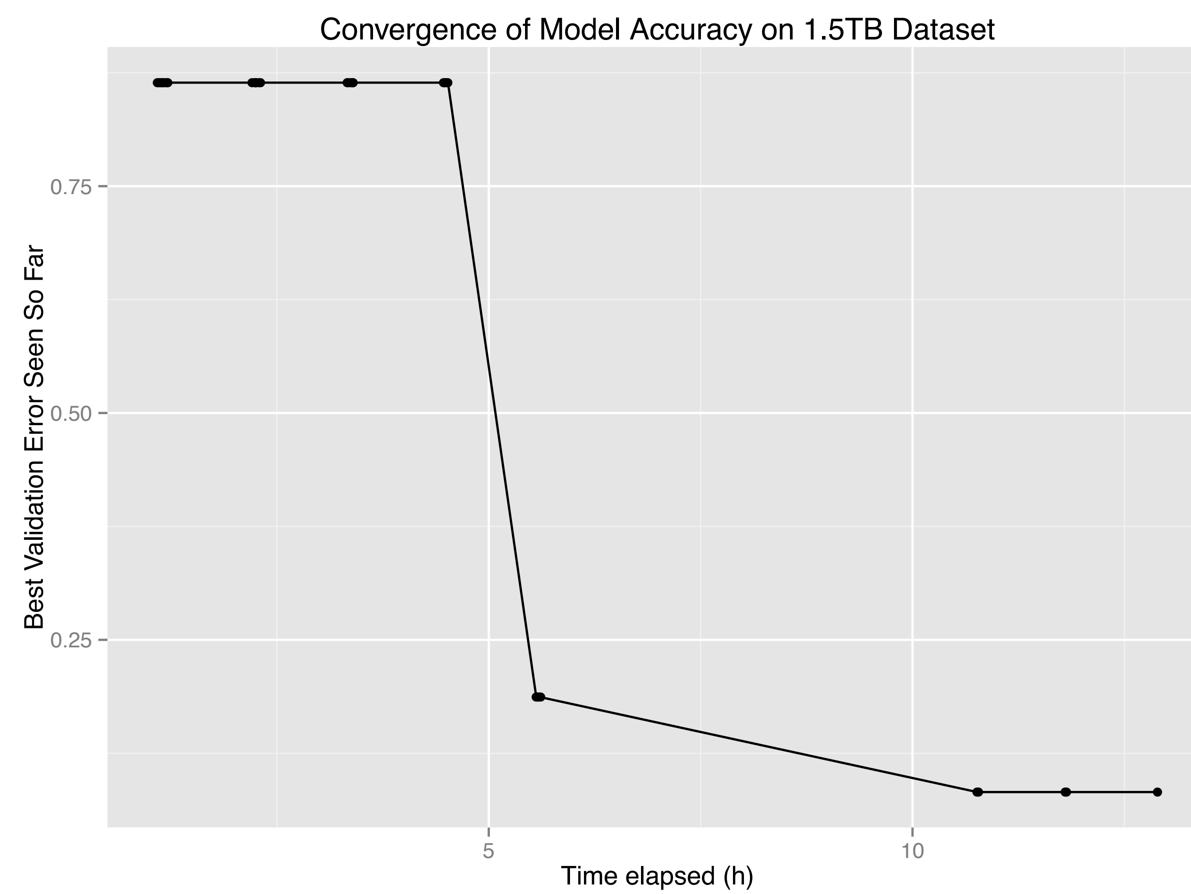
- First version of MLbase optimizer
- 30GB dense images (240K x 16K)
- 2 model families, 5 hyperparams
- Baseline: grid search
- Our method: combination of
  - Batching
  - Early stopping
  - Random or TPE



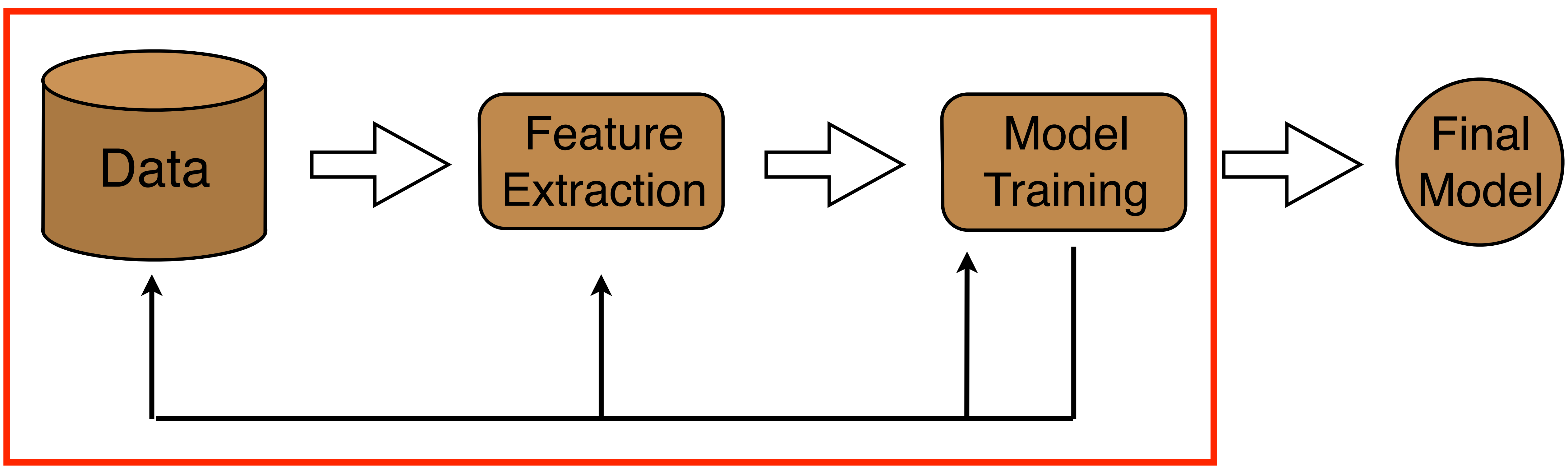
20x speedup compared to grid search
15 minutes vs 5 hours!

Does It Scale?

- 1.5TB dataset (1.2M x 160K)
- 128 nodes, thousands of passes over data
- Tried 32 models in 15 hours
  - Good answer after 11 hours



Future Work



Automated ML Pipeline Construction

Other Future Work

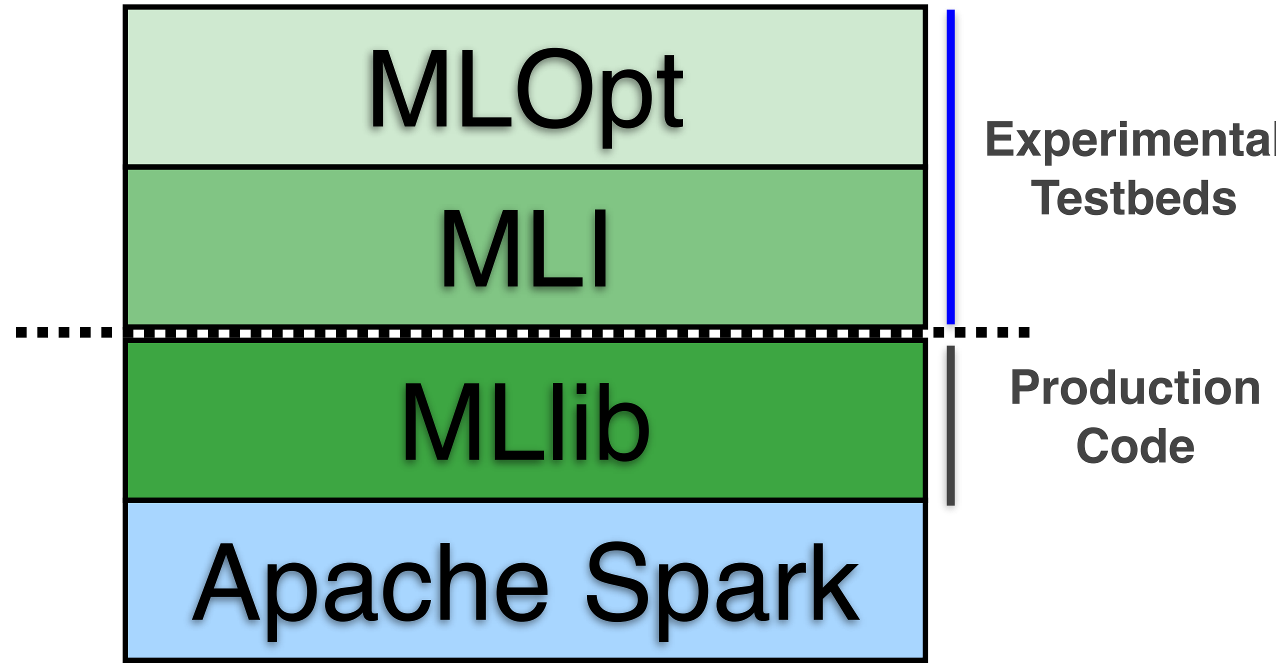
- Ensembling
- Leverage sampling
- Better parallelism for smaller datasets
- Multiple hypothesis testing issues

MLOpt: Declarative layer to automate hyperparameter tuning

MLI: API to simplify ML development

MLlib: Spark's core ML library

Spark: Cluster computing system designed for iterative computation



MLbase website

www.mlbase.org

MLlib Programming Guide

spark.apache.org/docs/latest/mllib-guide.html

Scalable Machine Learning

www.edx.org/course/scalable-machine-learning-uc-berkeleyx-cs190-1x

Spark user lists

spark.apache.org/community.html