Machine Learning on Big Data

Part Two

Agenda

- Big Data Approaches
- Sampling and Fitting in Memory
- A Tour of Model Families

Big Data Approaches

Hypothesis One

More examples means better machine learning.

- See the forest from the trees.
- Noisy data

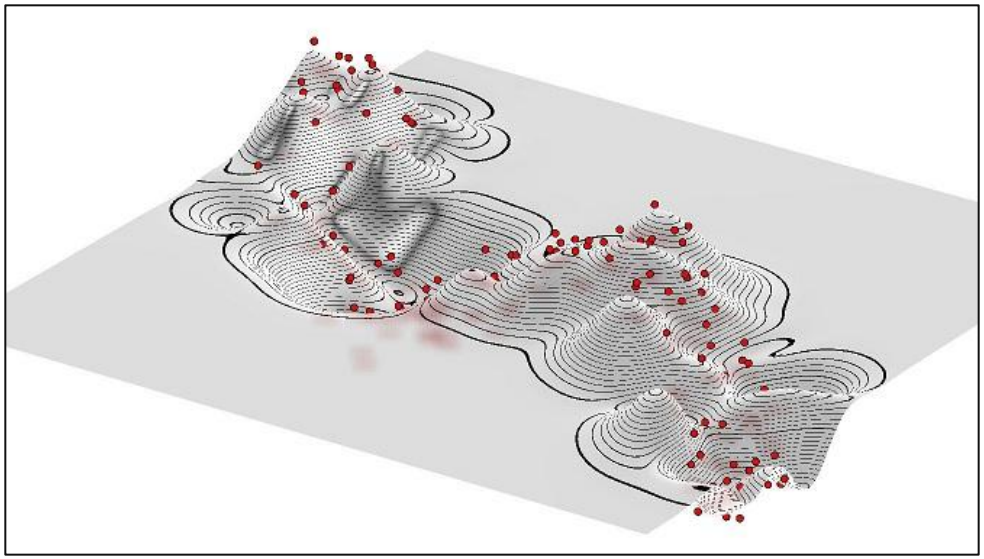
Attempt to capture a complete search space.

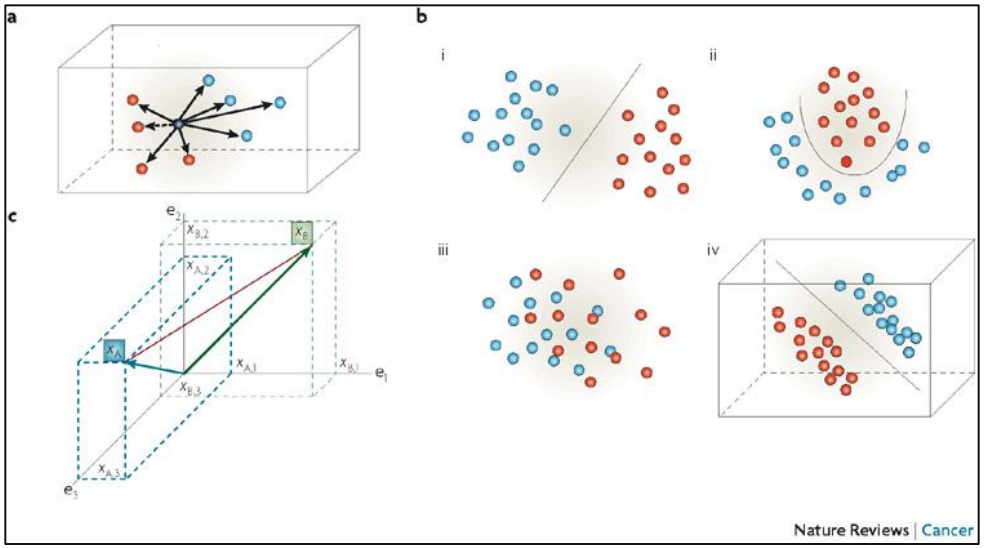
Hypothesis Two

Increasing dimensions or finding better properties improves pattern recognition.

- Semantic Embedding
- Separability

Attempt to divide a search space better.



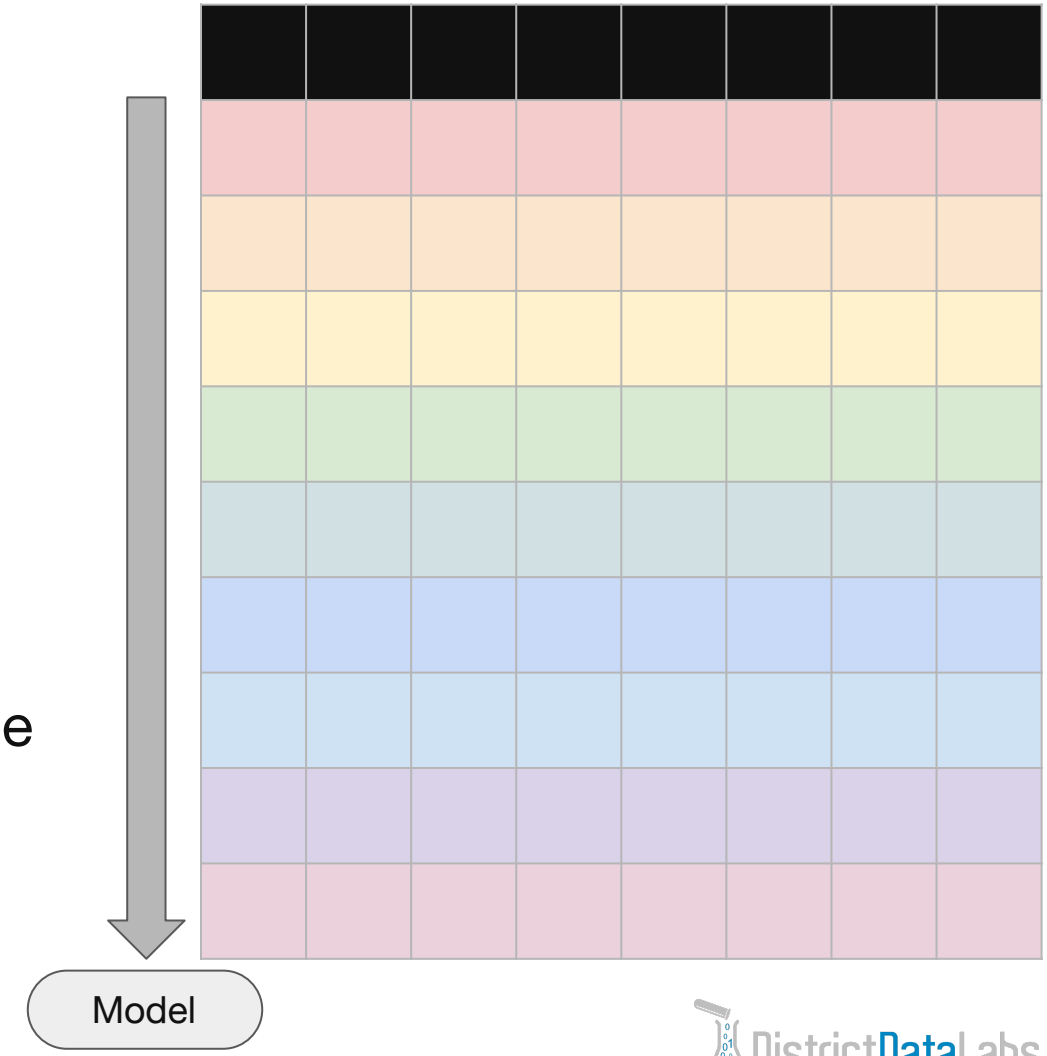


Sequential Machine Learning

Machine learning generally finds the best set of parameters for a model by optimizing some error function.

(This is most apparent in regression)

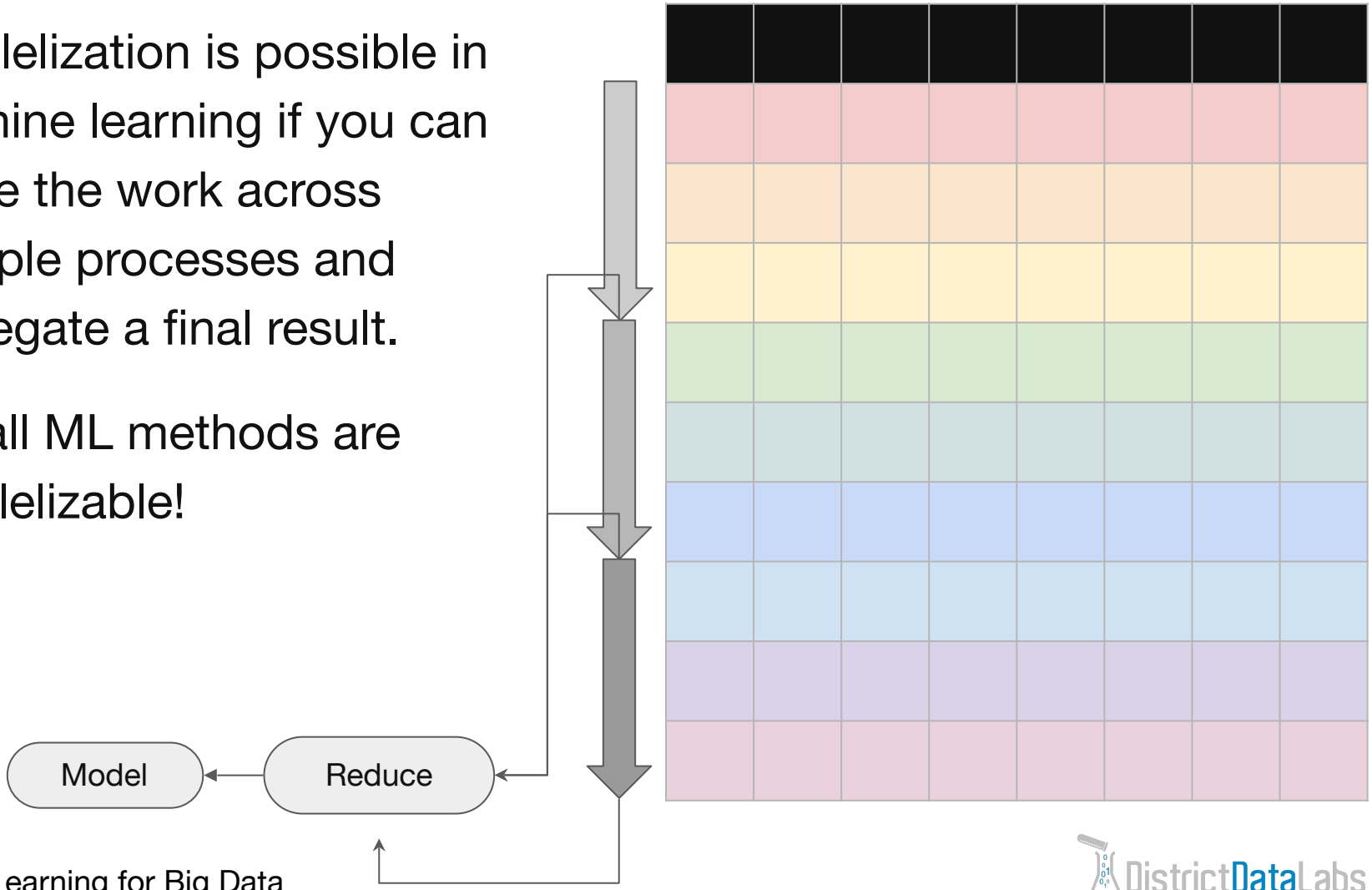
Multiple passes over multiple instances.



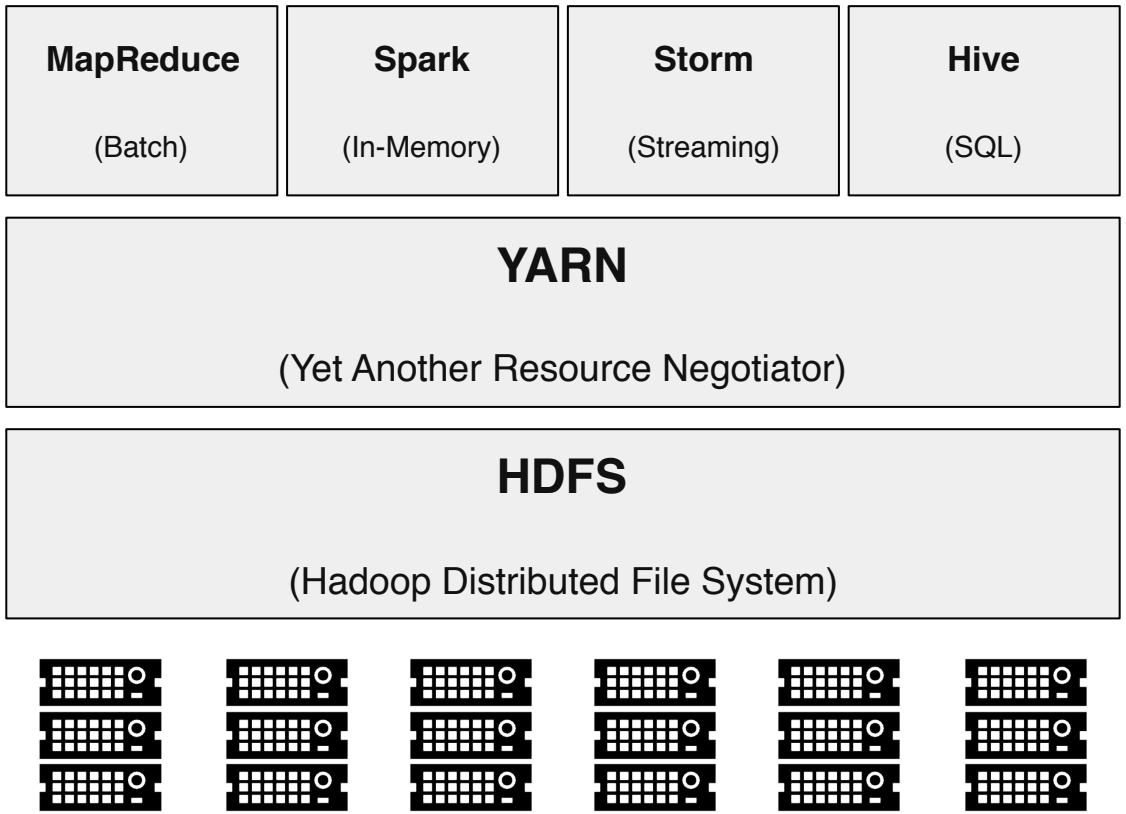
Parallel Machine Learning

Parallelization is possible in machine learning if you can divide the work across multiple processes and aggregate a final result.

Not all ML methods are parallelizable!



Hadoop Stack

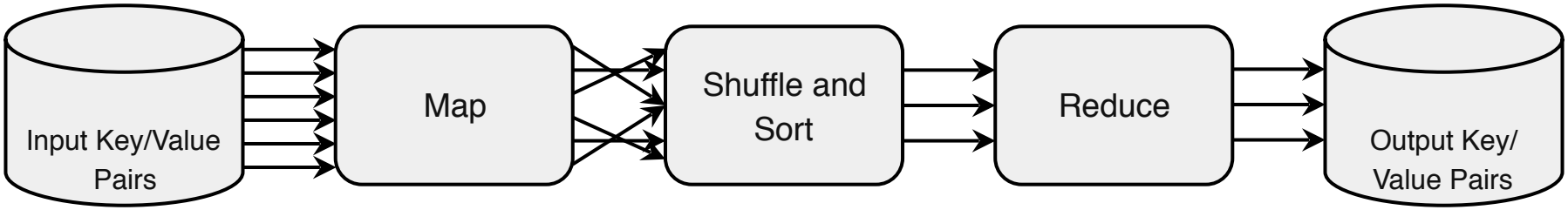


Cluster of economy disks and processors

Single Map and Reduce

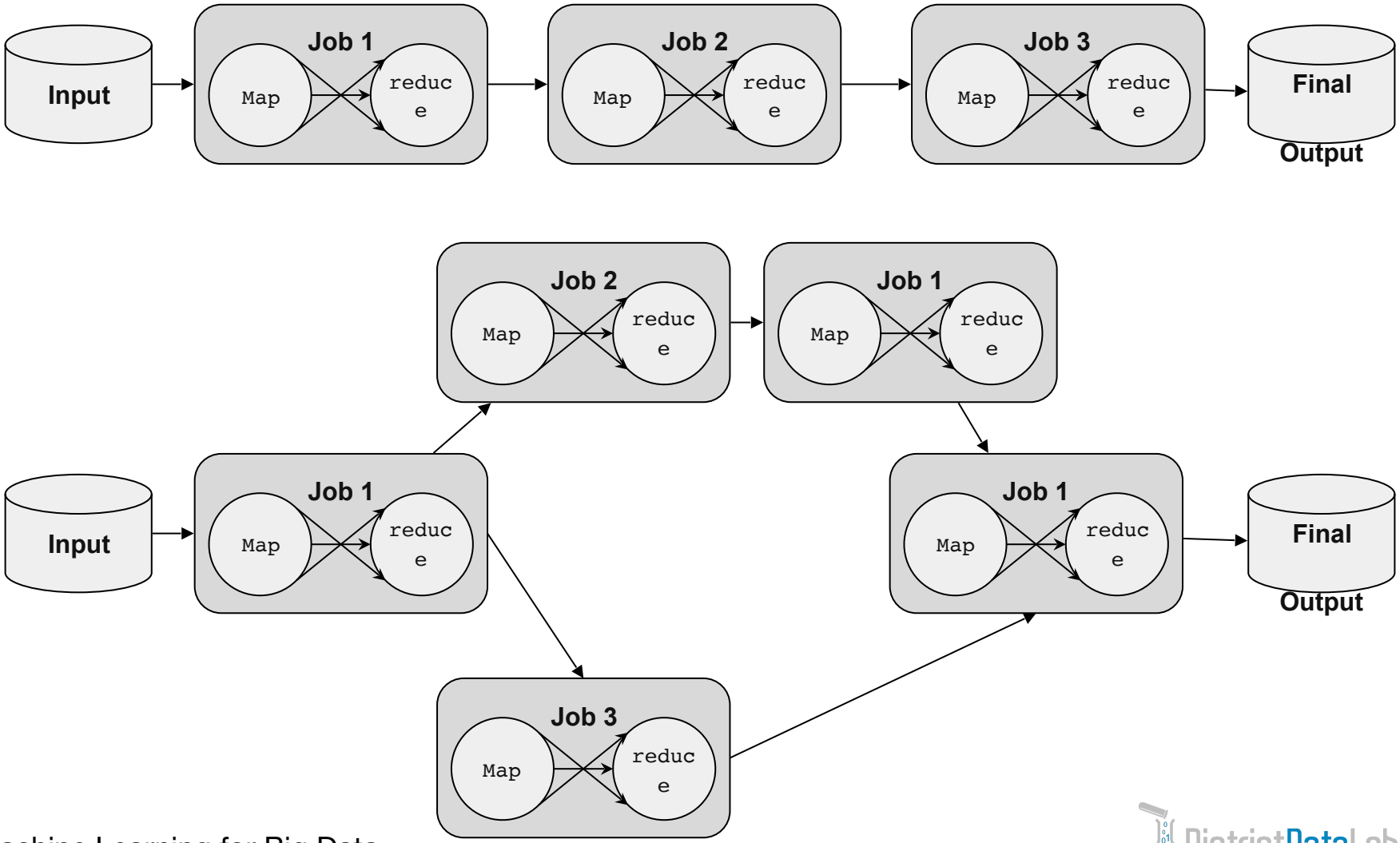
Data is already split into blocks and passed to mapping computations.

Results of mapping have to be aggregated into a final computation.



Data has to be moved from the mapping tasks to the reducing ones.

Job Chaining for Iteration: Directed Acyclic Graphs



Challenges to Distributed ML

Rewriting Algorithms

How do we restructure algorithms to take advantage of parallelism?

E.g. use gradient descent over ordinary least squares.

Iterative Analysis

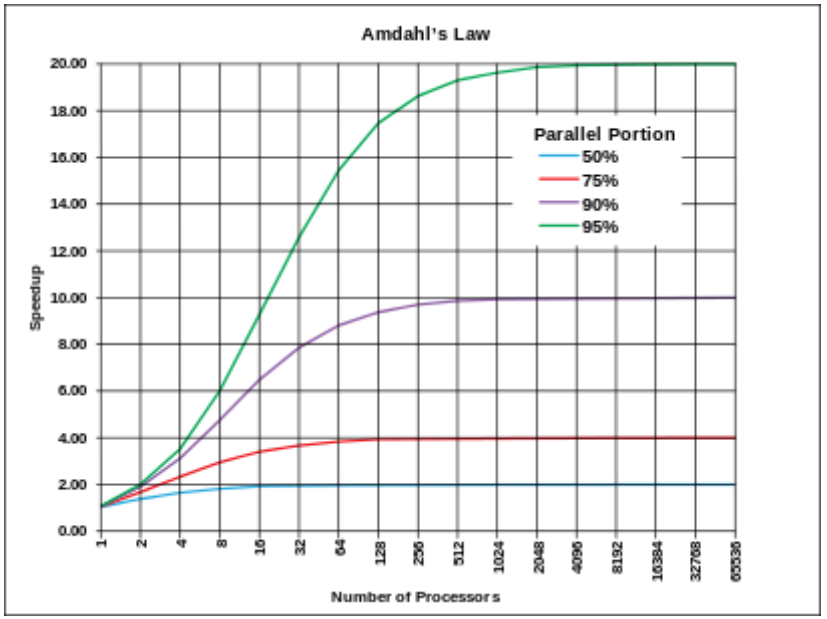
MapReduce writes to disk between Map and Reduce phase.

MapReduce also fixes the number of jobs that can be run.

Amdahl's Law

The amount of speed-up of a parallel system by adding more processors is limited by the percent of the program that is parallelizable.

Parallel processes have setup overhead (sequential portions).



Two Approaches to Big Data ML

Decompose to Memory

Use decomposition methods to reduce input domain into something that can fit into memory (filtering, sampling, summarization, indexing).

Compute model in memory on a single machine (128 GB).

Evaluate model on cluster.

Boost Weaker Models

Use distributed implementations for models that are in Mahout or Spark to perform computation.

Boost or bag the models together to produce a stronger model.

Performs better with some models, e.g. Random Forest.

Two Approaches to Big Data ML

Decompose to Memory

Use decomposition methods to reduce input domain into something that can fit into memory (filtering, sampling, summarization, indexing).

Compute model in memory on a single machine (scikit-learn).

Evaluate model on cluster.

Boost Weaker Models

Use distributed implementations for models that are in Mahout or Spark to perform computation.

Boost or bag the models together to produce a stronger model.

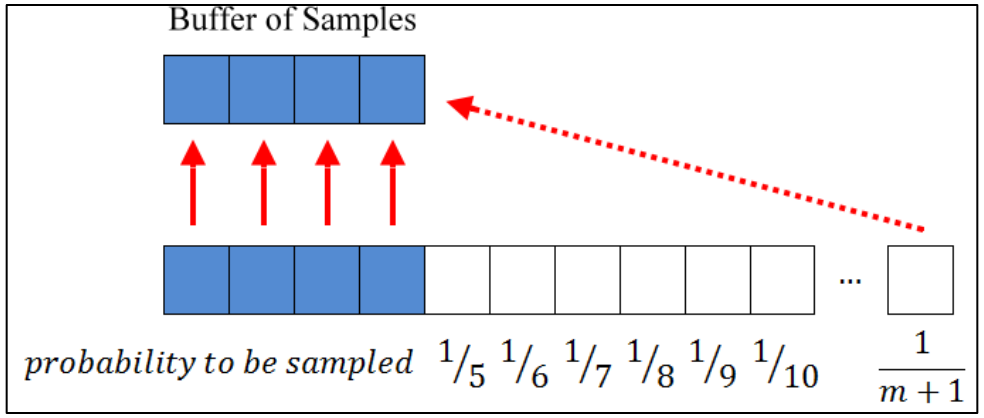
Performs better with some models, e.g. Random Forest.

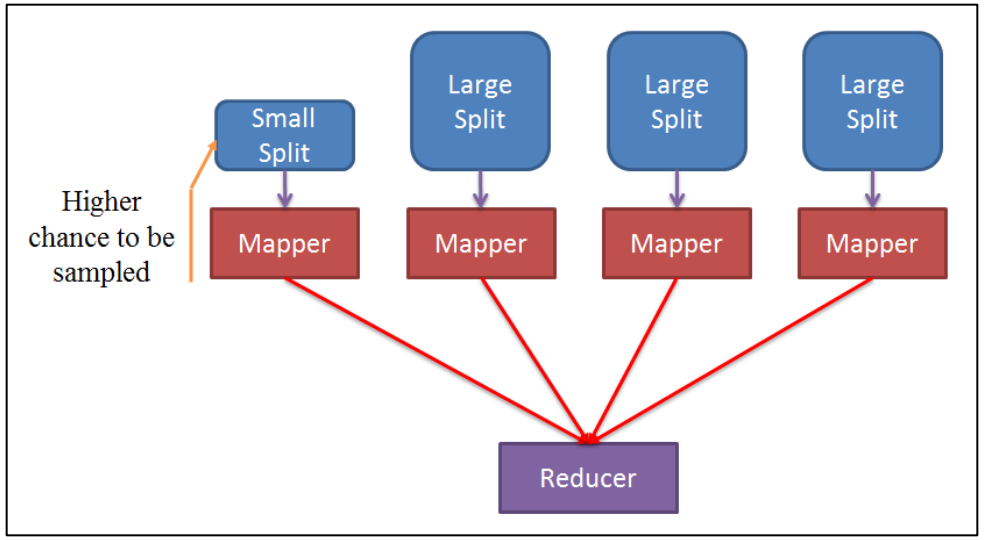
Sampling and Fitting in Memory

Sampling Approach 1: % of Data Set

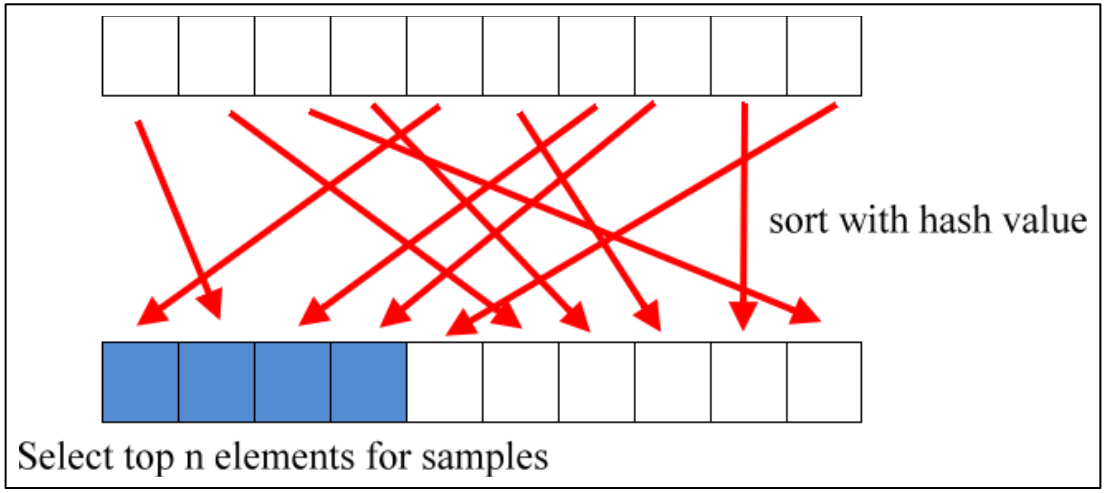
```
import random

class PercentSampleMapper(Mapper):

    def __init__(self, *args, **kwargs):
        self.percentage = kwargs.pop("percentage")
        super(PercentSampleMapper, self).__init__(*args,
**kwargs)

    def map(self):
        for _, val in self:
            if random.random() < self.percentage:
                self.emit(None, val)

if __name__ == '__main__':
    mapper = PercentSampleMapper(sys.stdin, percentage=0.20)
    mapper.map()
```

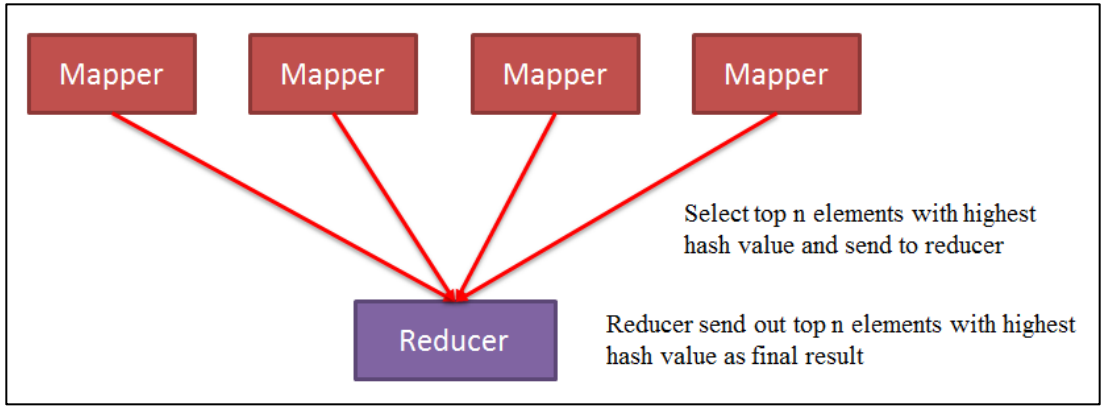
Reservoir Sampling: Sequential and Distributed





Solution: Top N of Top N Random Numbers





Sampling Approach 2: Distributed Reservoir

```
import random, heapq

class SampleMapper(Mapper):

    def __init__(self, n, *args, **kwargs):
        self.n = n
        super(SampleMapper, self).__init__(*args, **kwargs)

    def map(self):
        # initialize our heap as a list with n zeros
        heap = [0 for x in xrange(self.n)]

        for value in self:
            # maintain a heap of only n largest values
            heapq.heappushpop(heap, (random.random(), value))

        for item in heap:
            # emit the sampled values
            self.emit(None, item)
```

Sampling Approach 2: Distributed Reservoir

```
class SampleReducer(Mapper):

    def __init__(self, n, *args, **kwargs):
        self.n = n
        super(SampleReducer, self).__init__(*args, **kwargs)

    def reduce(self):
        # initialize our heap as a list with n zeros
        heap = [0 for x in xrange(self.n)]

        for _, values in self:
            for value in values:
                heapq.heappushpop(heap, make_tuple(value))

        for item in heap:
            # emit the sampled values
            self.emit(None, item[1])
```

Scikit-Learn

Standard API for machine learning:

Estimator

Allows the use of multiple models with little code change.

```
class Estimator(object):

    def fit(self, X, y=None):
        """Fits estimator to data. """
        # set state of ``self``
        return self

    def predict(self, X):
        """Predict response of ``X``.
"""
        # compute predictions ``pred``
        return pred
```

Scikit-Learn

Standard API for machine learning:

Estimator

Allows the use of multiple models with little code change.

```
from sklearn import svm

estimator = svm.SVC(gamma=0.001)
estimator.fit(X, y)
estimator.predict(x)
```

How do you serialize a linear model?

$$\hat{Y} = \varepsilon + \beta_0 x_0 + \beta_1 x_1 + \ldots + \beta_n x_n$$

Writing Scikit-Learn Models to Disk

```
import pickle

# Dump a Scikit-Learn Fitted Estimator to
disk
with open(path, 'wb') as f:
    pickle.dump(model, f)

# Load a Scikit-Learn Fitted Estimator from
disk
with open(path, 'rb') as f:
    model = pickle.load(f)
```

Computing the MSE using Spark

```
def cost(row, clf=None):
    """Computes the square error given the row."""
    return (row[0] - clf.predict(row[1:])) ** 2

def main(sc):
    # Load the model from the pickle file
    with open('clf.pickle', 'rb') as f:
        clf = sc.broadcast(model.load(f))

    # Create an accumulator to sum the squared error
    sum_square_error = sc.accumulator(0)

    # Load and parse the blog data
    blogs = sc.textFile("blogData").map(float)

    # Map the cost function and accumulate the sum.
    error = blogs.map(partial(cost, clf=clf))
    error.foreach(lambda cost: sum_square_error.add(cost))

    # Print and compute the mean.
    print sum_square_error.value / error.count()
```

Key Concepts in Spark ML

Spark ML standardizes APIs for machine learning algorithms to make it easier to combine multiple algorithms into a single pipeline, or workflow.

DataFrame: Spark ML operates on a DataFrame loaded from SparkSQL

Transformer: Modifies one DataFrame into another DataFrame. Used for feature extraction or preprocessing, standardization, normalization.

Estimator: A learning algorithm is an Estimator which trains on a DataFrame and produces a model.

Key Concepts in Spark ML

Pipeline: A Pipeline chains multiple Transformers and Estimators together to specify an ML workflow.

Parameter: All Transformers and Estimators now share a common API for specifying parameters.

A Tour of Model Families

Models: Instance Methods

Compare instances in data set with a similarity measure to find best matches.

- Suffers from curse of dimensionality.
- Focus on feature representation and similarity metrics between instances

k-Nearest Neighbors (kNN)
Self-Organizing Maps (SOM)
Learning Vector Quantization (LVQ)

Models: Regression

Model relationship of independent variables, X to dependent variable Y by iteratively optimizing error made in predictions.

Ordinary Least Squares
Logistic Regression
Stepwise Regression
Multivariate Adaptive Regression Splines (MARS)
Locally Estimated Scatterplot Smoothing (LOESS)

Models: Regularization Methods

Extend another method (usually regression), penalizing complexity (minimize overfit)

- simple, popular, powerful
- better at generalization

Ridge Regression
LASSO (Least Absolute Shrinkage & Selection Operator)
Elastic Net

Models: Decision Trees

Model of decisions based on data attributes. Predictions are made by following forks in a tree structure until a decision is made. Used for classification & regression.

Classification and Regression Tree (CART)
Decision Stump
Random Forest
Multivariate Adaptive Regression Splines (MARS)
Gradient Boosting Machines (GBM)

Models: Bayesian

Explicitly apply Bayes' Theorem for classification and regression tasks. Usually by fitting a probability function constructed via the chain rule and a naive simplification of Bayes.

Naive Bayes
Averaged One-Dependence Estimators (AODE)
Bayesian Belief Network (BBN)

Models: Kernel Methods

Map input data into higher dimensional vector space where the problem is easier to model. Named after the “kernel trick” which computes the inner product of images of pairs of data.

Support Vector Machines (SVM)
Radial Basis Function (RBF)
Linear Discriminant Analysis (LDA)

Models: Clustering Methods

Organize data into into groups whose members share maximum similarity (defined usually by a distance metric). Two main approaches: centroids and hierarchical clustering.

k-Means
Affinity Propegation
OPTICS (Ordering Points to Identify Cluster Structure)
Agglomerative Clustering

Models: Artificial Neural Networks

Inspired by biological neural networks, ANNs are nonlinear function approximators that estimate functions with a large number of inputs.

- System of interconnected neurons that activate
- Deep learning extends simple networks recursively

Perceptron
Back-Propagation
Hopfield Network
Restricted Boltzmann Machine (RBM)
Deep Belief Networks (DBN)

Models: Ensembles

Models composed of multiple weak models that are trained independently and whose outputs are combined to make an overall prediction.

Boosting
Bootstrapped Aggregation (Bagging)
AdaBoost
Stacked Generalization (blending)
Gradient Boosting Machines (GBM)
Random Forest

Models: Other

The list before was not comprehensive, other algorithm and model classes include:

Conditional Random Fields (CRF)
Markovian Models (HMMs)
Dimensionality Reduction (PCA, PLS)
Rule Learning (Apriori, Brill)
More ...

Hands-On Lab

Tasks