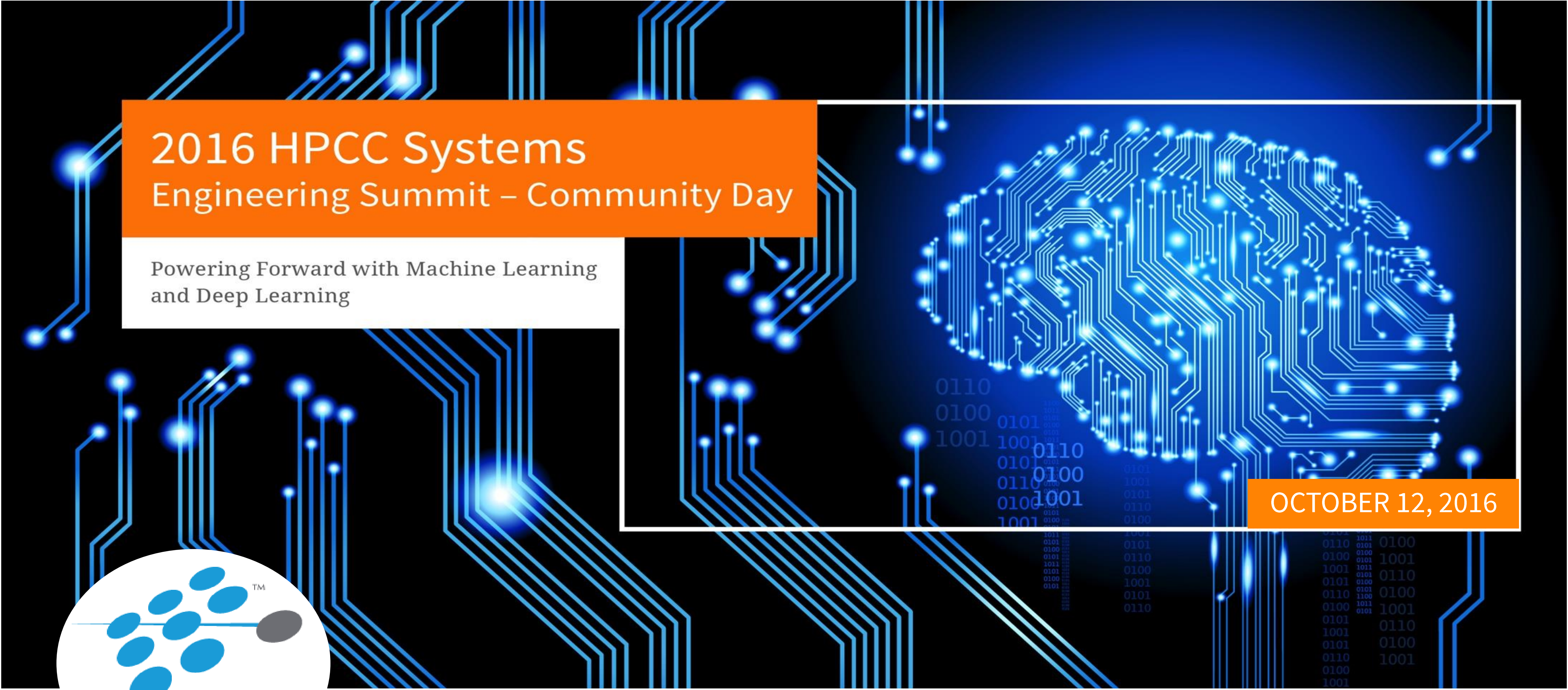


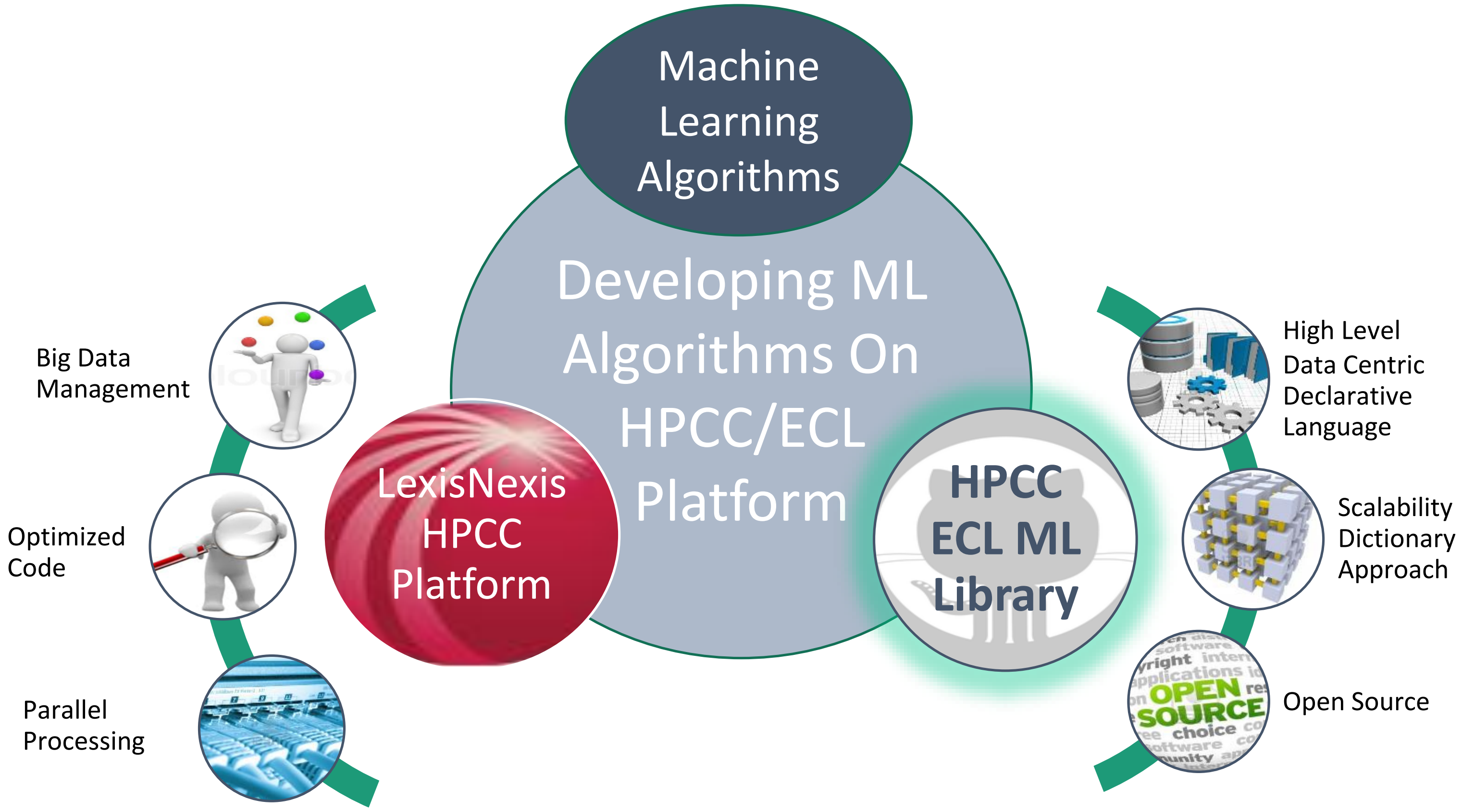

HPCC SYSTEMS®

# Optimizing Supervised Machine Learning Algorithms and Implementing Deep Learning in HPCC Systems

Victor Herrera
Maryam Najafabadi

FAU

# LexisNexis/Florida Atlantic University Cooperative Research



# Agenda

- Optimizing Supervised Methods
  Victor Herrera

- Toward Deep Learning
  Maryam Najafabadi

# Optimizing Supervised Methods

# Overview

ML-ECL Random Forest Optimization:

- Decreased significantly the time for Learning and Classification phases.
- Improved Classification performance.

Working with Sparse Data:

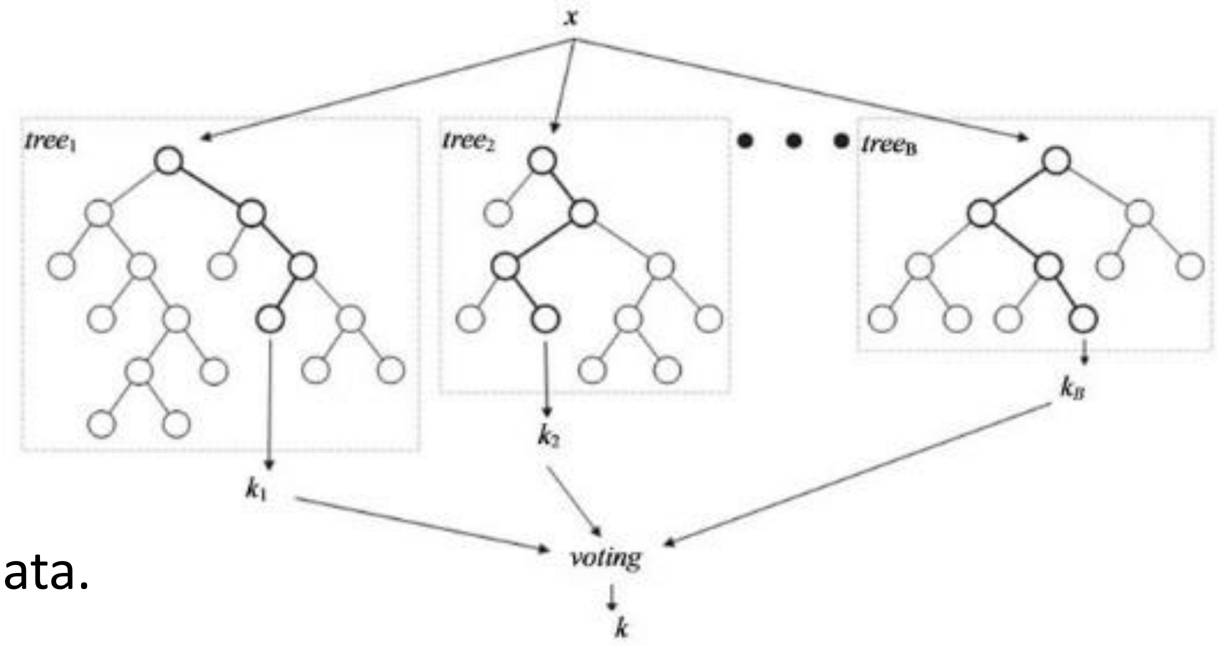
- Sparse ARFF reduced dataset representation.
- Speed Up Naïve Bayes algorithm learning and classification time on highly sparse datasets.

# Random Forest

Random Forest (Breiman, Leo. 2001)
Ensemble supervised learning algorithm for classification and regression.
Operate by constructing a multitude of decision trees.

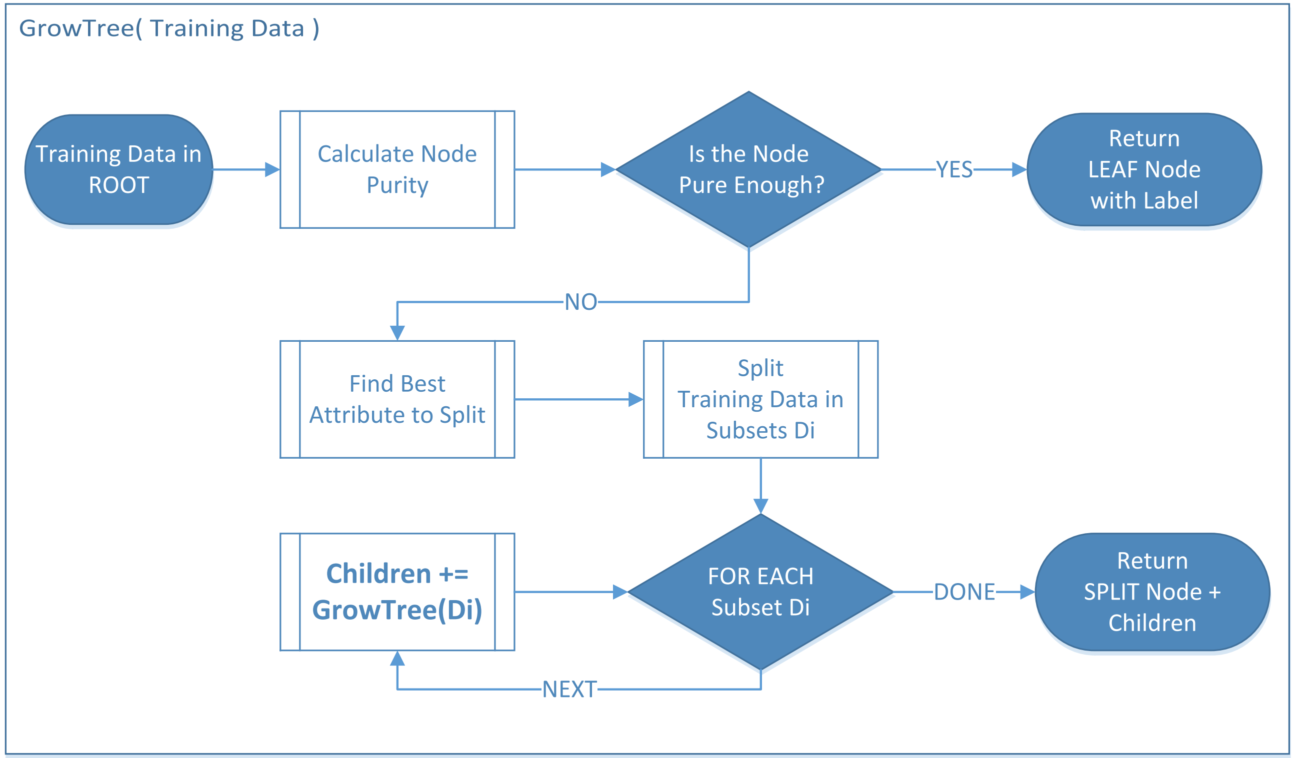
Main Idea:
Most of the trees are good for most of the data and make mistakes in different places

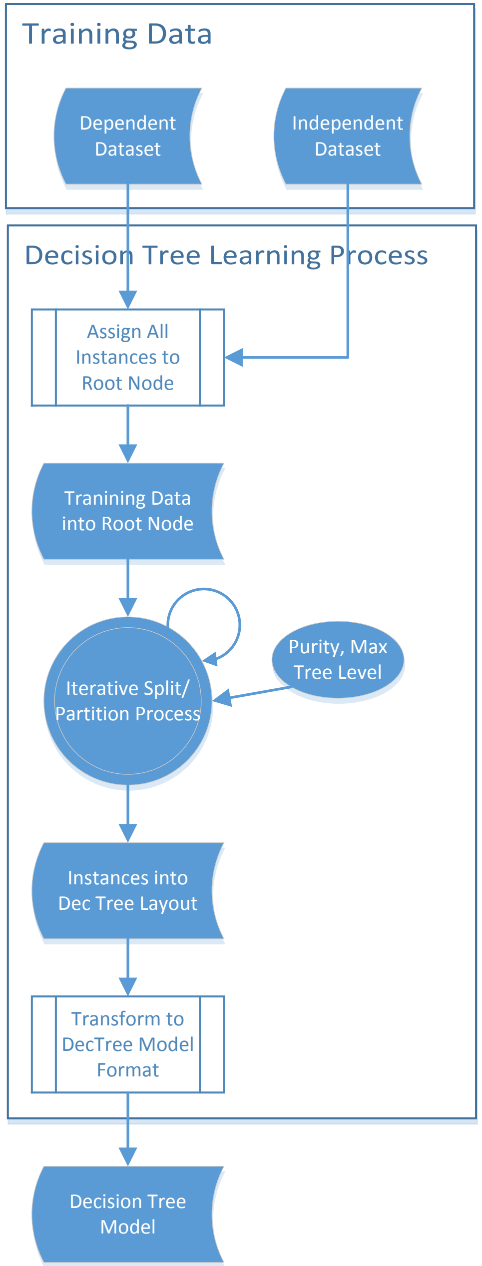
How:
DT Bagging - Rnd samples with replace
Splits over Rnd Selection of Features
Majority Voting



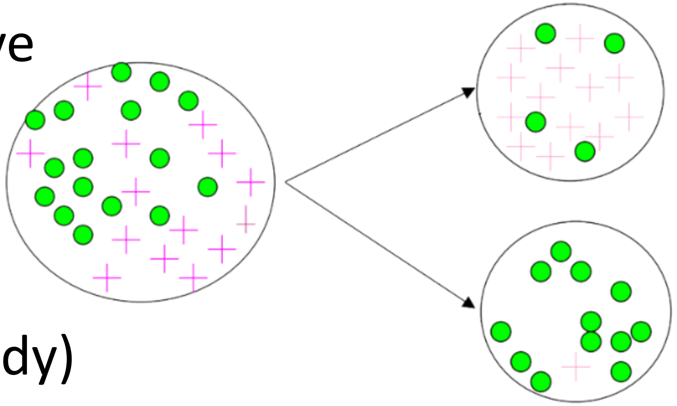

Why RF:
Overcomes overfitting problem
Handles wide, unbalanced class, and noisy data.
Generally outperforms single algorithms.
Good for parallelization.

# Recursive Partitioning as Iterative in ECL

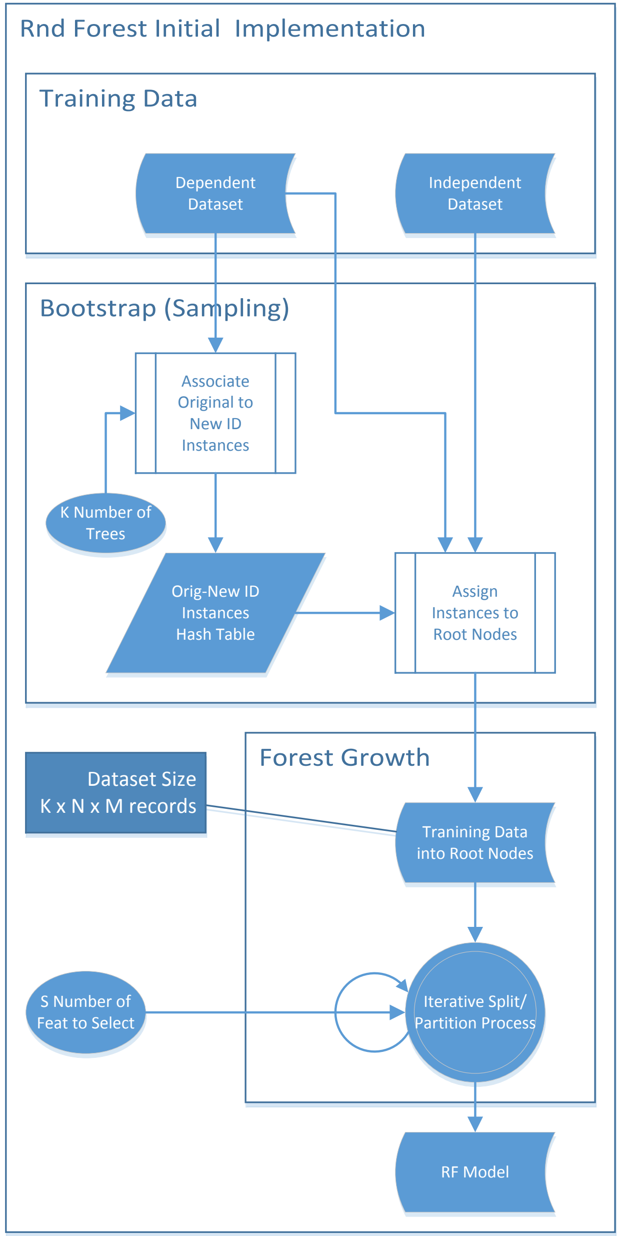



- Random Forest Learning is based on Recursive Partitioning as in Decision Trees.
- Forward References not allowed in ECL
- DecTree Learning implemented in ECL as an Iterative Process via LOOP(dataset, …,loopbody)



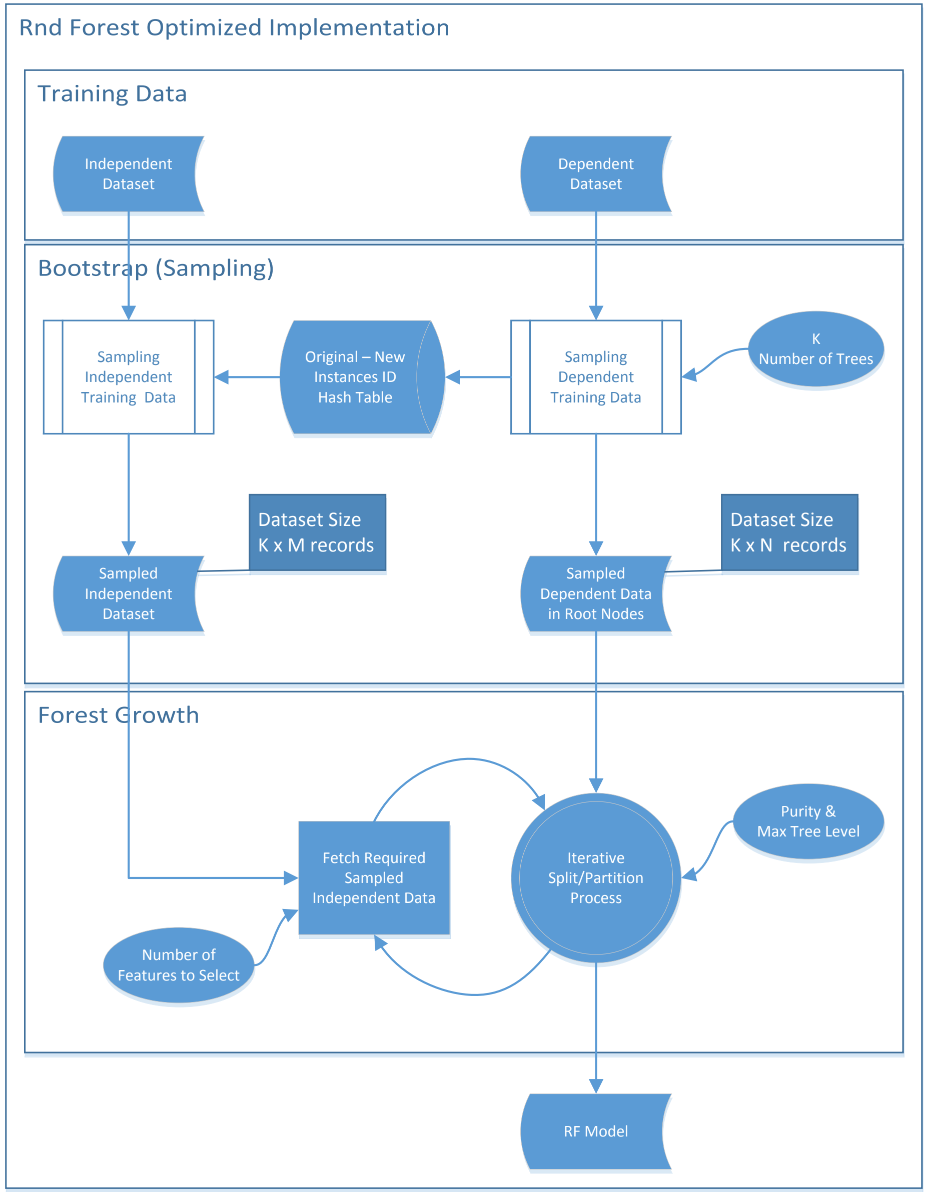
# Random Forest Learning Optimization




Initial Implementation Flaws:

- For every single iteration of Iterative Split/Partition LOOP at least K x N x M records are sent to the loopbody function:
  - For each LOOP iteration every Node-Instance record pass to loopbody function regardless of whether its processing was completed or not.
  - Wasting resources by including Independent data as part of loopbody function's INPUT:
    - Node Purity based only upon Dependent data
    - Finding Best Split per Node only needs subsets of Independent data (Feature Selection)
- Implementation was not fully parallelized.

# Random Forest Learning Optimization

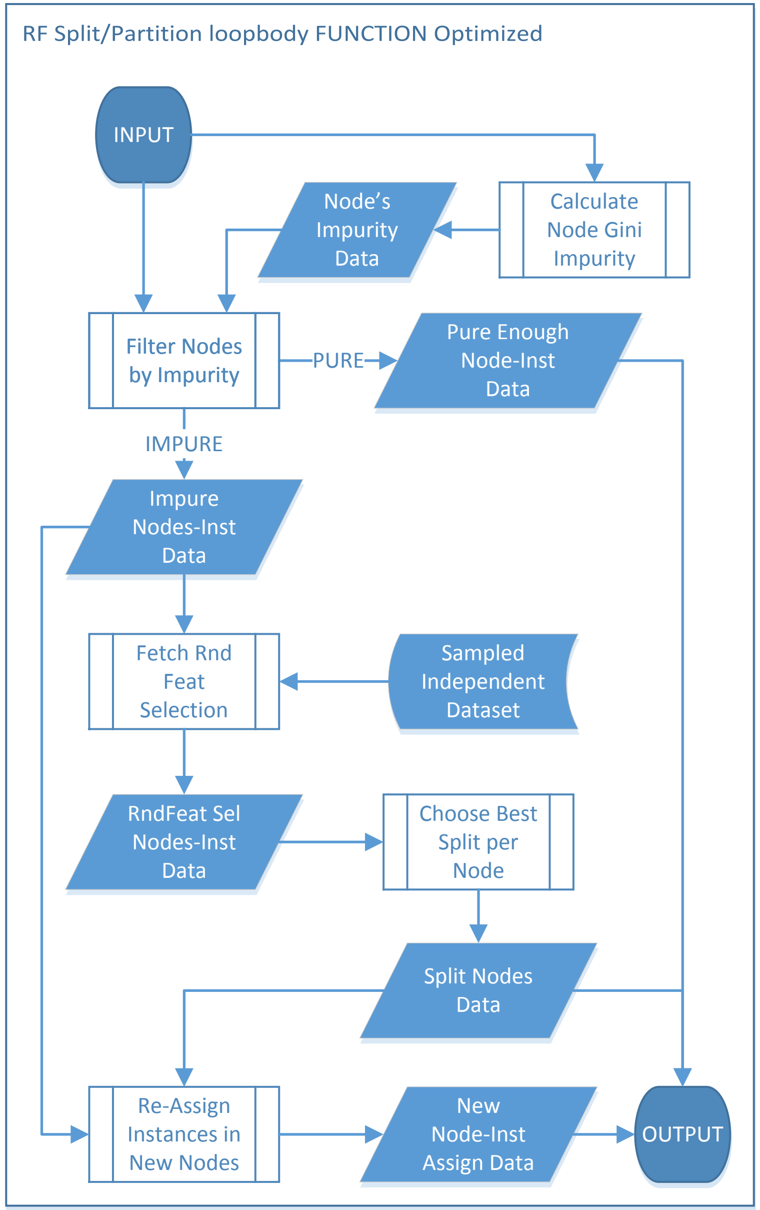



Review of initial implementation helped to re-organize the process and data flows.

We improved our initial approach in order to:

- Filter records not requiring further processing (LOOP - rowfilter).
- Pass only one RECORD per instance (dependent value) into loopbody function.
- Fetch only Required Independent data from within the function at each iteration.
- Take full advantage of distributed data storage and parallel processing capabilities of the HPCC Systems Platform.

# Random Forest Learning Optimization



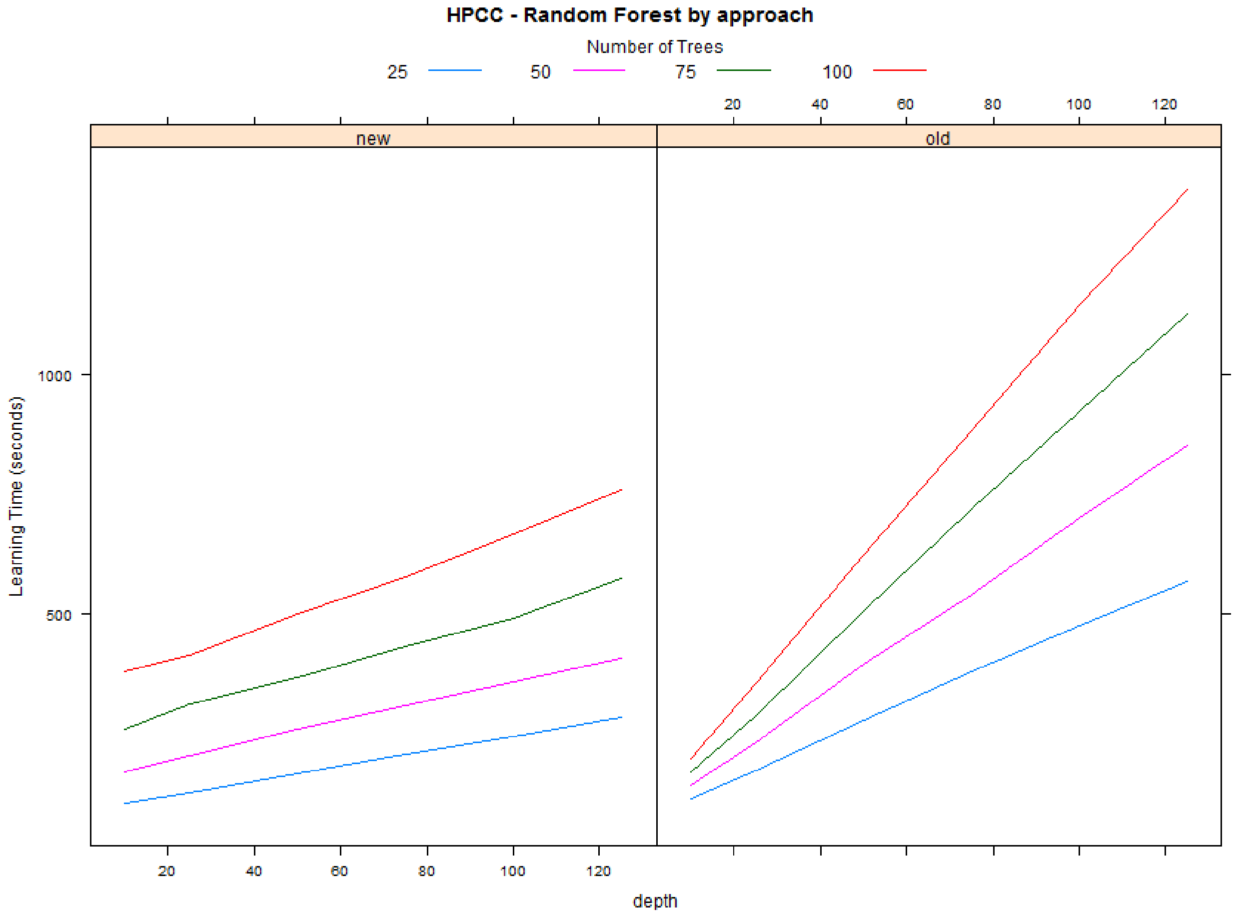

Loopbody function fully parallelized:

- Receives and returns one RECORD per Instance.
- Node Impurity and Best Split per Node calculations done LOCAL-ly:
  - Node-Instance Data DISTRIBUTED by Node_id.
- Fetching Rnd Feat Selection using JOIN-LOCAL:
  - Sampled Independent data generated and DISTRIBUTED by inst. Id at BOOTSTRAP.
  - Instances-Features Selected combinations dataset (RETRIEVER) DISTRIBUTED by inst. Id.
- Inst. Relocation to New Nodes done LOCAL-ly:
  - Impure Node-Instance Data still DISTRIBUTED by Node_id.
  - JOIN-LOOKUP with Split Nodes data.

# Random Forest Learning Optimization – Preliminary Results

Preliminary Comparison of Learning Time between Initial version (old) and Optimized Beta version (new):

- Adult Dataset:
  - Discrete dataset
  - 16281 instances * 13 feat + class
  - Balanced
- 6 Features Selected (HALF total)
- Number of Trees: 25 , 50, 75 and 100
- Depth: 10, 25, 50, 75, 100 and 125
- 10 runs for each case




Preliminary Results gave us green light to complete the final optimized implementation:

- Fully parallelized Learning Process
- New Optimized Classification Process

# Working with Sparse Data – NaïveBayes

```
0 0 1 0 0 posclass
2 0 0 0 1 posclass
0 0 0 0 0 negclass
```

Sparse matrix is a matrix in which most elements are zero. One way to reduce its dataset representation is using Sparse ARFF file format.

```
//ARFF file                | //Sparse ARFF file        | //Sparse Types.DiscreteField DS
                           | attr index starts in 0    | attr index starts in 1
@data                      | @data                     | //defValue:= 0, posclass:=1 ,
negclass:= 0
0, 0, 1, 0, 0, posclass    | {2 1, 5 posclass}         | [{1, 3, 1}, {1, 6, 1},
2, 0, 0, 0, 1, posclass    | {0 2, 4 1, 5 posclass}    |  {2, 1, 2}, {2, 5, 1}, {2, 6, 1},
0, 0, 0, 0, 2, negclass    | {4 2, 5 negclass}         |  {3, 5, 2}, {3, 6, 0}]
```

**Instead of using 15 records to represent the data only 7 were enough.**

NaiveBayes using Sparse ARFF:

- Highly sparse datasets, as in Text Mining Bag of Words, are represented with a few records in ECL.
- Save Disk/Memory space.
- Extend the default value “0” to any value defined by DefValue:= value;
- Accelerate Calculations based on DefValue’s frequency pre computations. Speed up both Learning and Classification time.

# Sparse Naïve Bayes – Results

Sentiment dataset (Bag of Words):

- 1.6 millions instances x 109,735 feat + class
- 175.57 billions of DiscreteField records

Original NaiveBayes classification using sub-samples of Sentiment Dataset:

- 5% , job completed, in avg. a little more than 1 hour
- 20% , job completed, in avg. around 4.5 hours
- 50% , job completed, in avg. around 12.5 hours

SparseARFF format Sentiment dataset:

- 1.6 e+6 lines, Between 1 to 30 Non Default values per line – Very High Sparsity
- Assuming 15 values in avg: 1.6e+6 x 15 = 24 millions DiscreteField records
- Default value "0"

SparseNaïveBayes using Sentiment equivalent SparseARFF dataset:

- Classification Test done in just 70 seconds.
- 10-Fold Cross Validation run takes only 6 minutes to finish

# Summary

ML-ECL Random Forest Speed Up:

- Learning Processing time reduction:
  - Reduction of R/W operations – Data passing simplification
  - Parallelization of loopbody function:
    - Reorganization of data distribution and aggregations
    - Fetching Required Independent Data only
- Classification Processing time reduction:
  - Implemented as iteration and fully parallelized.
- Classification performance improvement:
  - Feature selection randomization upgraded to node level

# Summary

Working with Sparse Data:

- Functionality to work with Sparse ARFF format files in HPCC
  - Sparse-ARFF to DiscreteField function implementation
- Implement Sparse Naïve Bayes Discrete classifier
  - Learning and classification phases fully operative
  - Highly Sparse Big Datasets processed in seconds

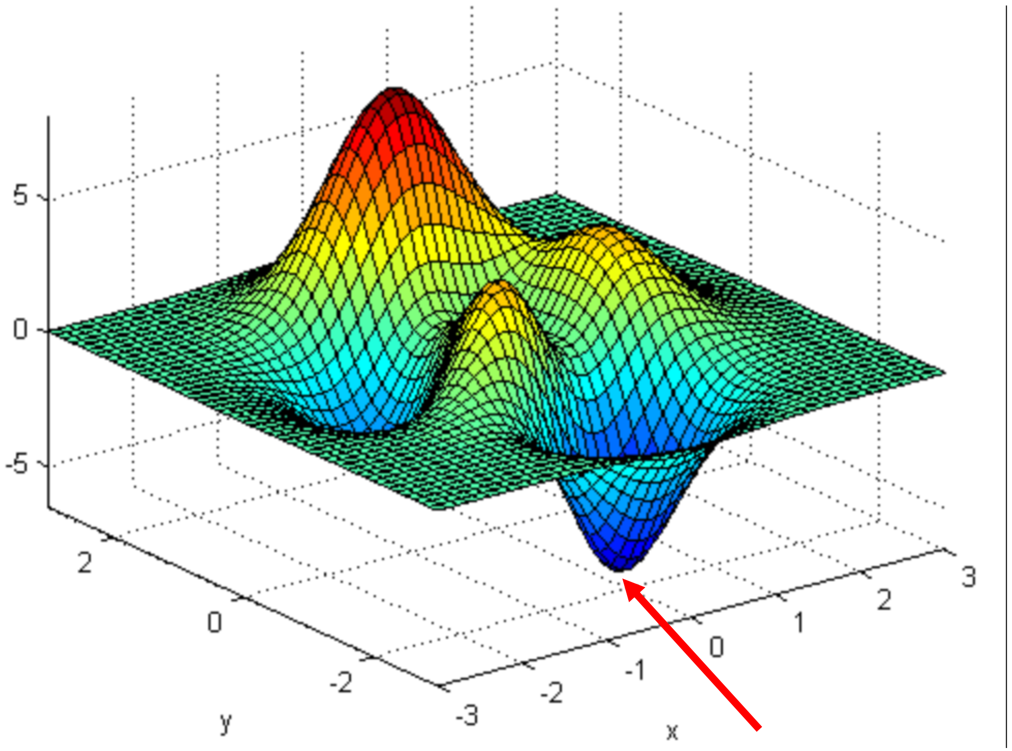
# Toward Deep Learning

# Overview

- Optimization algorithms on HPCC Systems
- Implementations based on the optimization algorithm

# Mathematical optimization

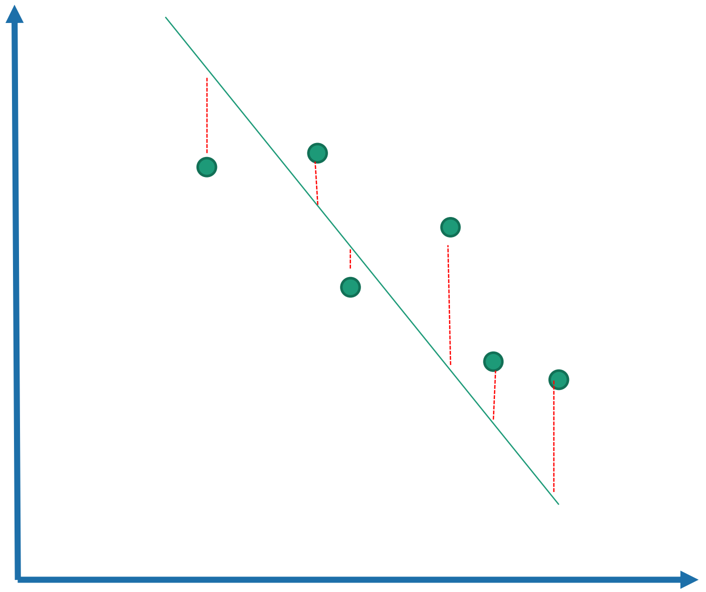
- Minimizing/Maximizing a function




Minimum

# Optimization Algorithms in Machine Learning

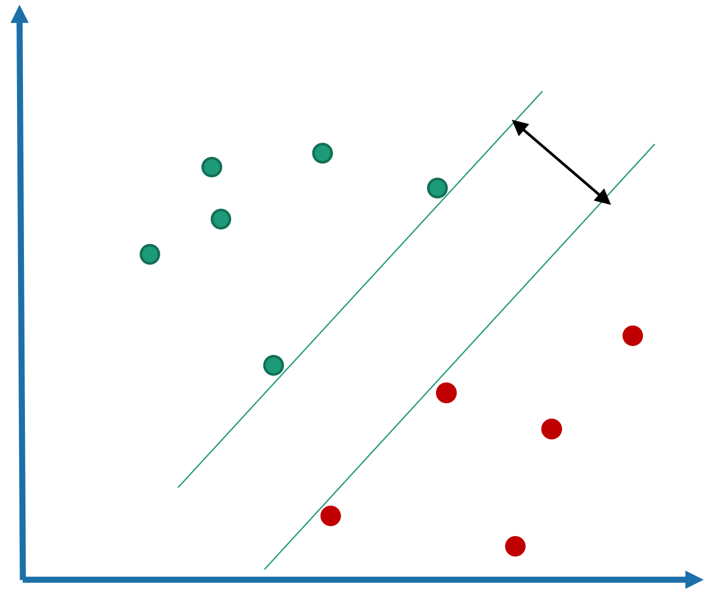
- The heart of many (most practical?) machine learning algorithms:
  - Linear regression



Minimize Errors

# Optimization Algorithms in Machine Learning

- SVM



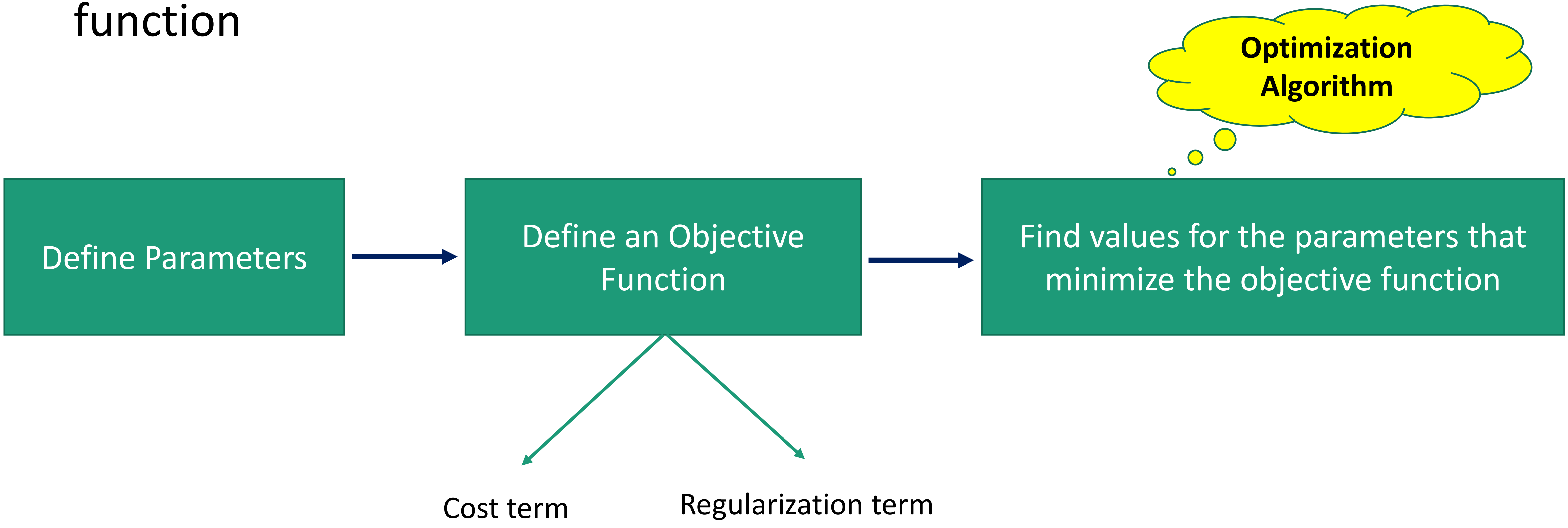
Maximize Margin

# Optimization Algorithms in Machine Learning

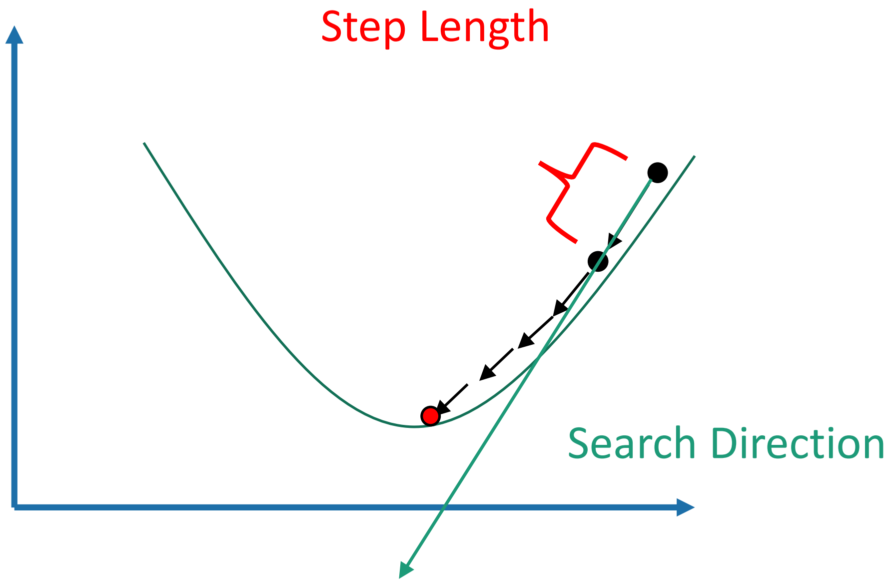
- Collaborative filtering
- K-means
- Maximum likelihood estimation
- Graphical models
- Neural networks
- Deep Learning

# Formulate Training as an Optimization Problem

- Training model: finding parameters that minimize some objective function



# How they work



# Gradient Descent

- Step length
  - Constant value

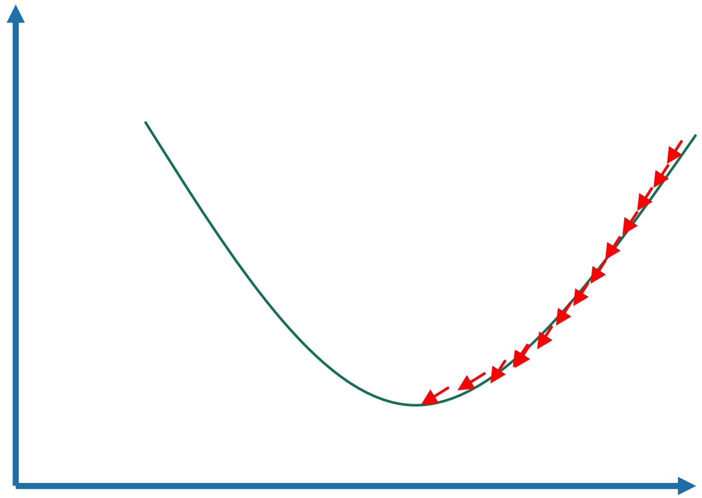
- Search direction
  - Negative gradient

# Gradient Descent

- Step length
  - Constant value

- Search direction
  - Negative gradient

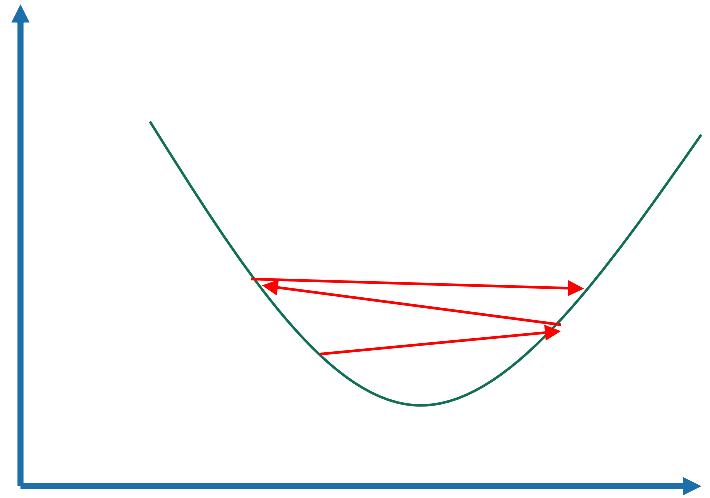
Small Step Length



# Gradient Descent

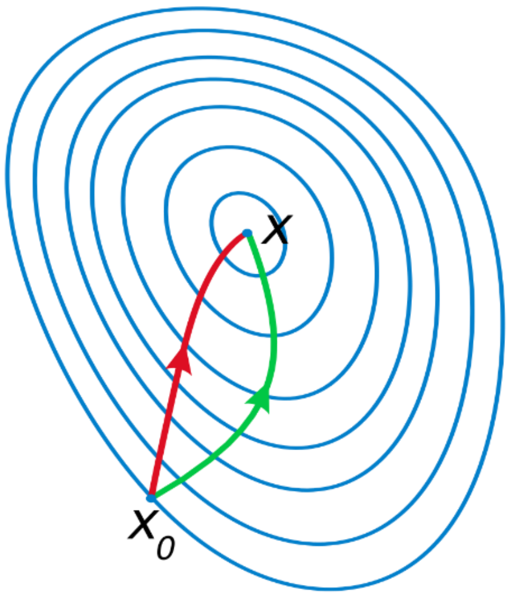
- Step length
  - Constant value
- Search direction
  - Negative gradient

Large Step Length



# Gradient Descent

- Step length
  - Constant value
- Search direction
  - Negative gradient



# L-BFGS

- Step length
  - Wolfe line search

- Search direction
  - Simple & Compact representation of Hessian Matrix

# L-BFGS

- Limited-memory -> only a few vectors of length n (instead of n by n )
- Useful for solving large problems (large n)
- More stable learning
- Uses curvature information to take a more direct route -> faster convergence

# How to use

- Define a function that calculates Objective value and Gradient

ObjectiveFunc (x, ObjectiveFunc_params, TrainData , TrainLabel)

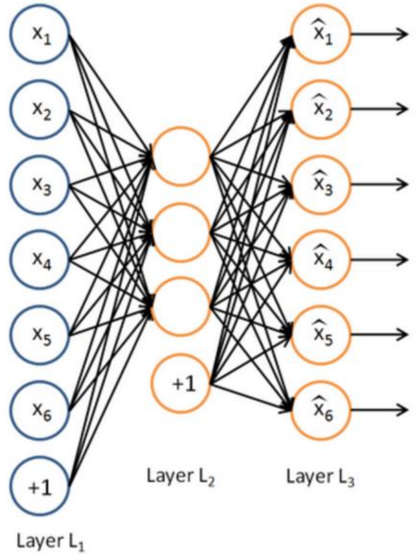
# L-BFGS based Implementations on HPCC Systems

- Sparse Autoencoder
- Softmax

# Sparse Autoencoder

- Autoencoder
  - Output is the same as the input
- Sparsity
  - constraint the hidden neurons to be inactive most of the time

- Stacking them up makes a Deep Network

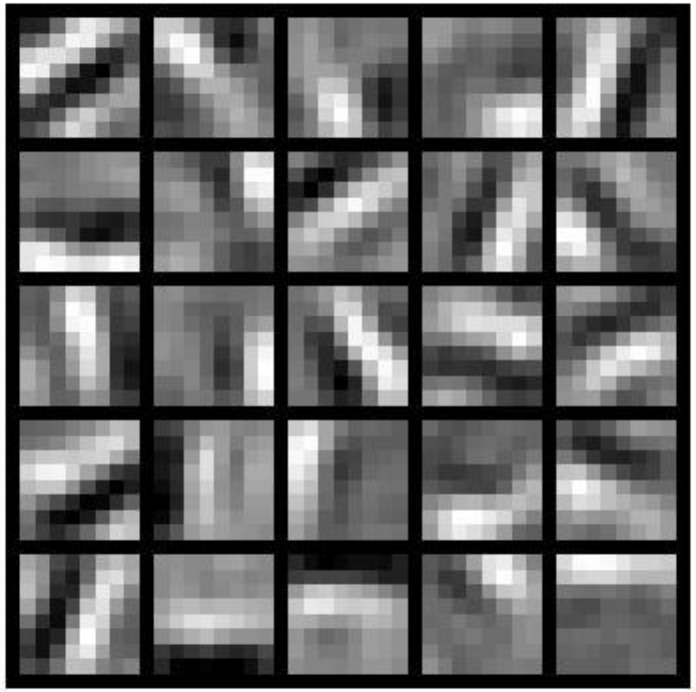


# Formulate to an optimization problem

- Parameters
  - Weight and bias values
- Objective function
  - Difference between output and expected output
  - Penalty term to impose sparsity
- Define a function to calculate objective value and Gradient at a give point

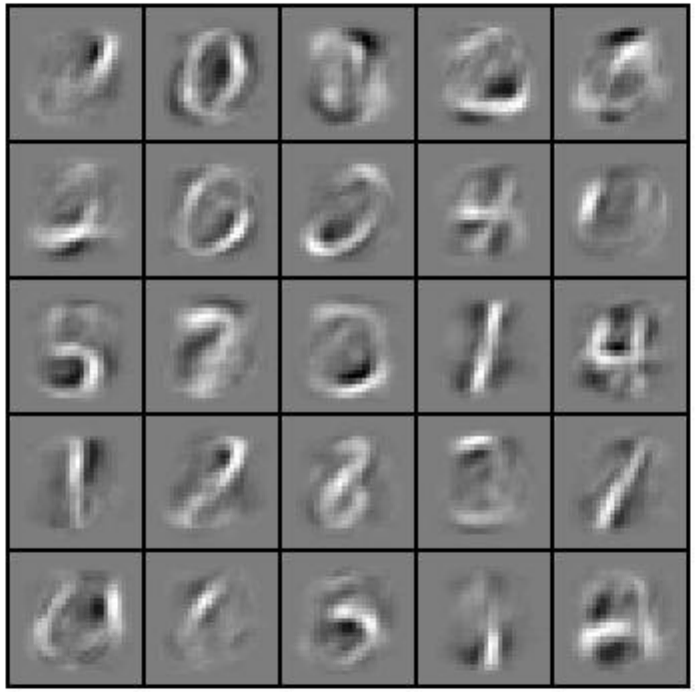
# Sparse Autoencoder results

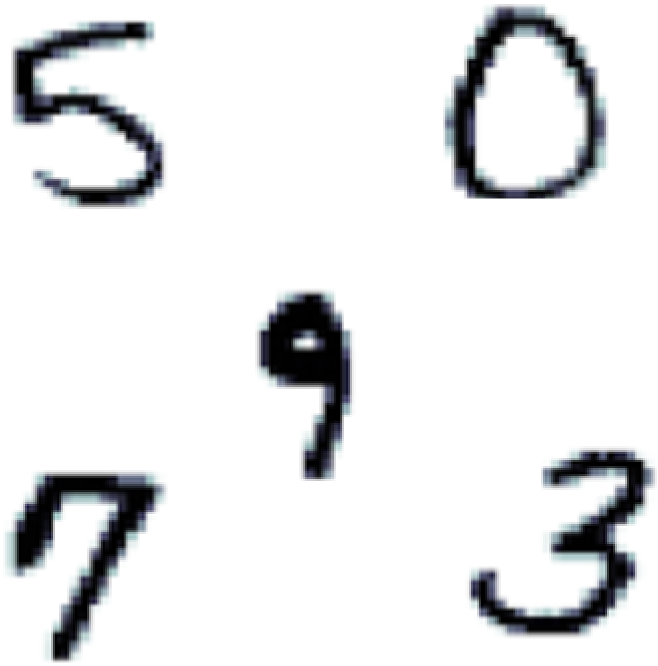
- 10’000 samples of randomly 8*8 selected patches



# Sparse Autoencoder results

- MNIST dataset





# SoftMax Regression

- Generalizes logistic regression
- More than two classes
- MNIST -> 10 different classes

# Formulate to an optimization problem

- Parameters
  - K by n variables
- Objective function
  - Generalize logistic regression objective function
- Define a function to calculate objective value and Gradient at a give point

# SoftMax Results

- Test on MNIST data
- Using features extracted by Sparse Autoencoder
- 96% accuracy

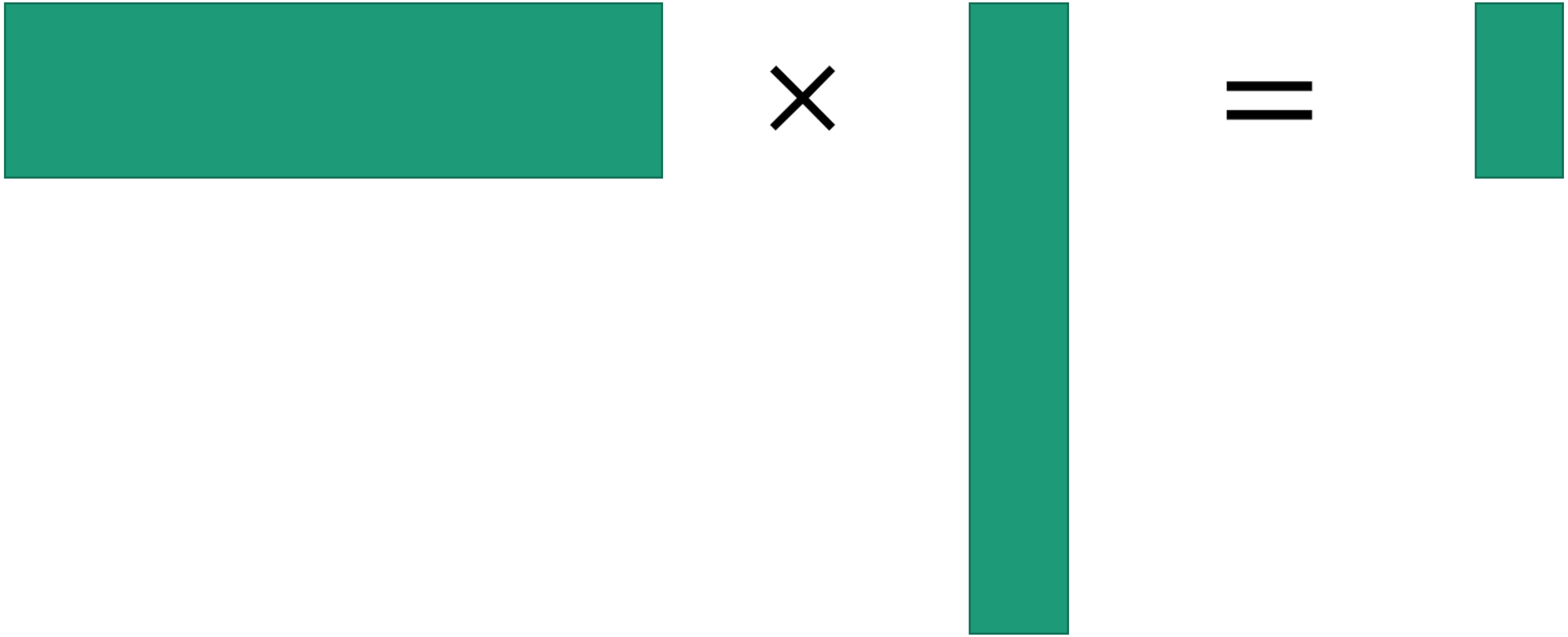
# Toward Deep Learning

- Provide learned features from one layer to another sparse autoencoder
- …. Stack up to build a deep network
- Fine tuning
  - Using forward propagation to calculate cost value and back propagation to calculate gradients
  - Use L-BFGS to fine tune

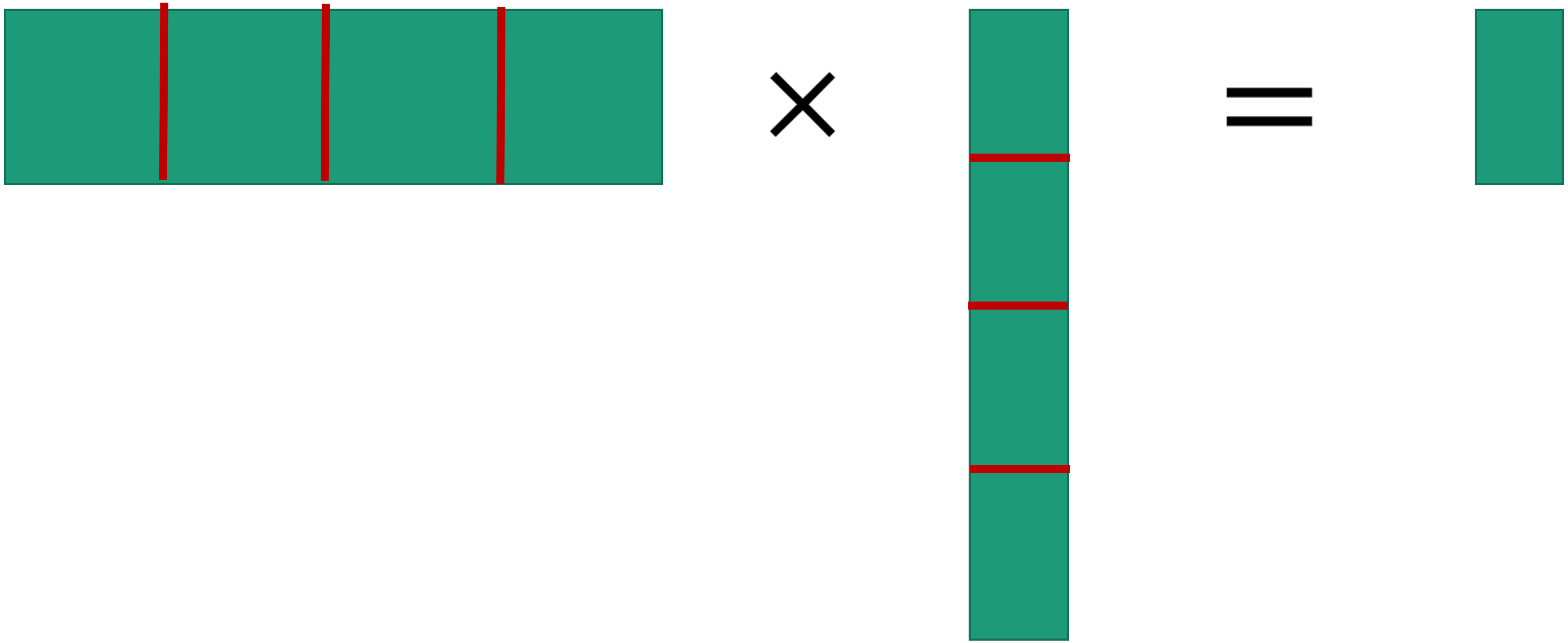
# Take Advantages of HPCC Systems

- PBblas
- Graphs

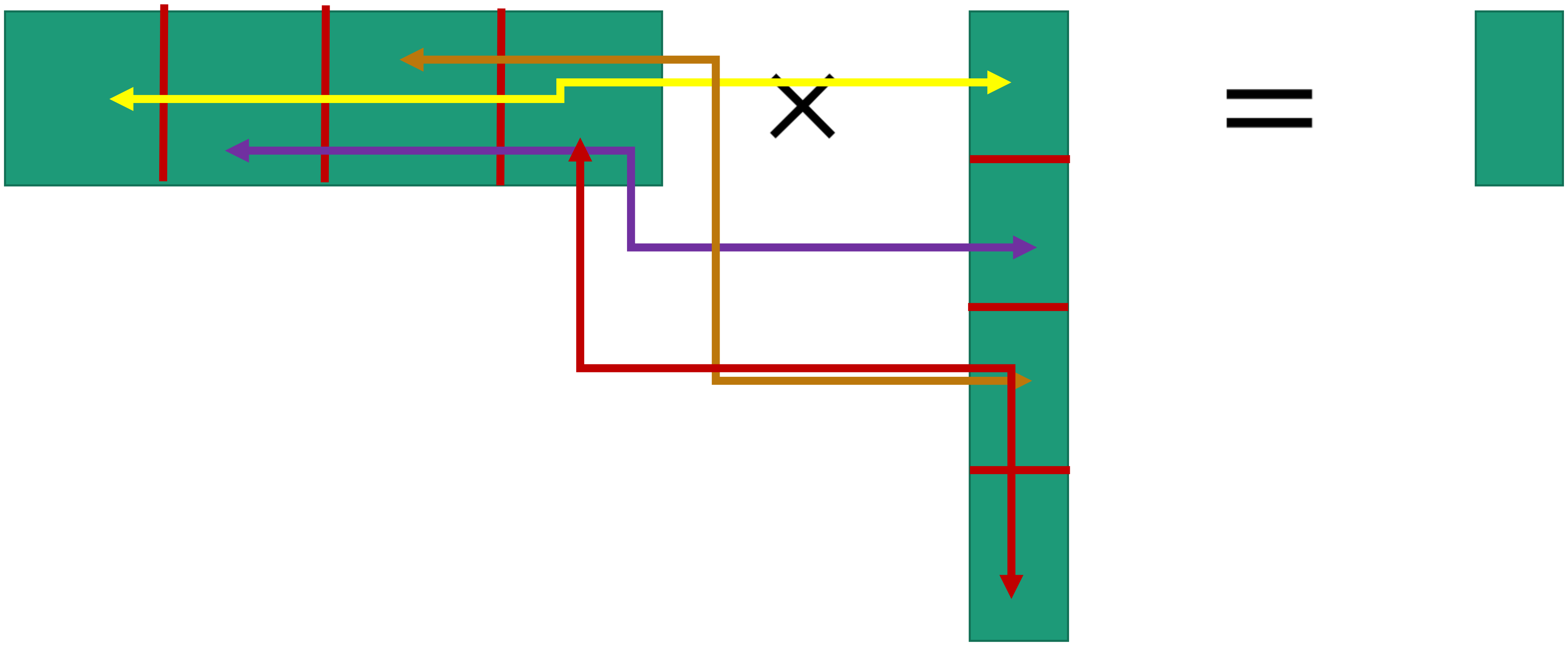
# Example

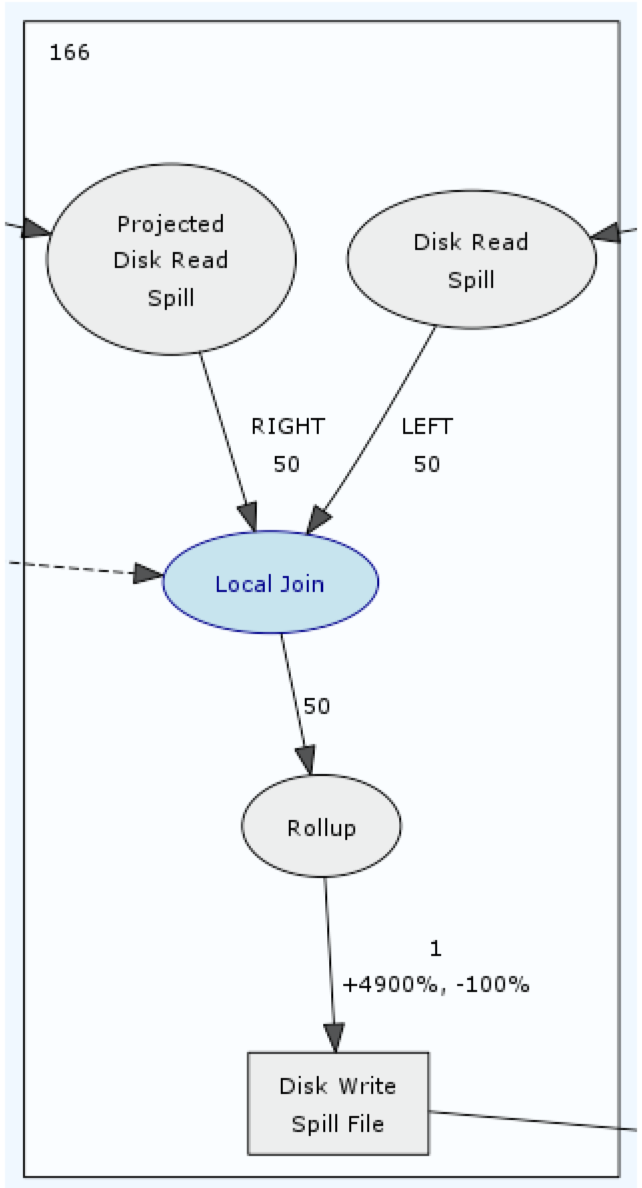


# Example



# Example




166
Projected
Disk Read
Spill
Disk Read
Spill
RIGHT
50
LEFT
50
Local Join
50
Rollup
1
+4900%, -100%
Disk Write
Spill File

# SUMMARY

- Optimization Algorithms an important aspect for advanced machine learning problems

- L-BFGS implemented on HPCC Systems
  - SoftMax
  - Sparse Autoencoder
- Implement other algorithms by calculating objective value and gradient
- Toward deep learning

# Questions?

Thank You