A Summary of Parallel Learning Efforts
DIMACS Workshop on Parallelism: A 2020 Vision

John Langford (Yahoo!)

What is Machine Learning?

The simple version:

Given data $(x, y)^*$ find a function $f(x)$ which predicts y.

$y \in \{0, 1\}$ is a "label"
$x \in R^n$ is "features"
$f(x) = \langle w \cdot x \rangle$ is a linear predictor.

What is Machine Learning?

The simple version:

Given data $(x, y)^*$ find a function $f(x)$ which predicts y.

$y \in \{0, 1\}$ is a "label"
$x \in R^n$ is "features"
$f(x) = \langle w \cdot x \rangle$ is a linear predictor.

y might be more complex and structured. Or nonexistent...
x might be a sparse vector or a string.
f can come from many more complex functional spaces.
In general: the discipline of data-driven prediction.

Where is Machine Learning?

At YAHOO! LABS :

1. Is the email spam or not?
2. Which news article is most interesting to a user?
3. Which ad is most interesting to a user?
4. Which result should come back from a search?

Where is Machine Learning?

At 

:

1. Is the email spam or not?
2. Which news article is most interesting to a user?
3. Which ad is most interesting to a user?
4. Which result should come back from a search?

In the rest of the world.

1. "statistical arbitrage"
2. Machine Translation
3. Watson
4. Face detectors in cameras
5. ... constantly growing.

How does it work?

A common approach = gradient descent.
Suppose we want to choose w for $f(x) = \langle w \cdot x \rangle$.
Start with $w = 0$.
Compute a "loss" according to $l_f(x, y) = (f(x) - y)^2$
Alter the weights according to $w \leftarrow w - \eta \frac{\partial l_f}{\partial w}$.

How does it work?

A common approach = gradient descent.
Suppose we want to choose w for $f(x) = \langle w \cdot x \rangle$.
Start with $w = 0$.
Compute a "loss" according to $l_f(x, y) = (f(x) - y)^2$
Alter the weights according to $w \leftarrow w - \eta \frac{\partial l_f}{\partial w}$.

There are many variations and many other approaches.
All efficient methods have some form of greedy optimization core.
But it's not just optimization:

1. We must predict the y correctly for new x.
2. There are popular nonoptimization methods as well.

Demonstration

Learning to classify news articles (RCV1 dataset)

Demonstration

Learning to classify news articles (RCV1 dataset)

An Outline of What's Next

Ron Bekkerman, Misha Bilenko and I are editing a book on "Scaling up Machine Learning". Overview Next.

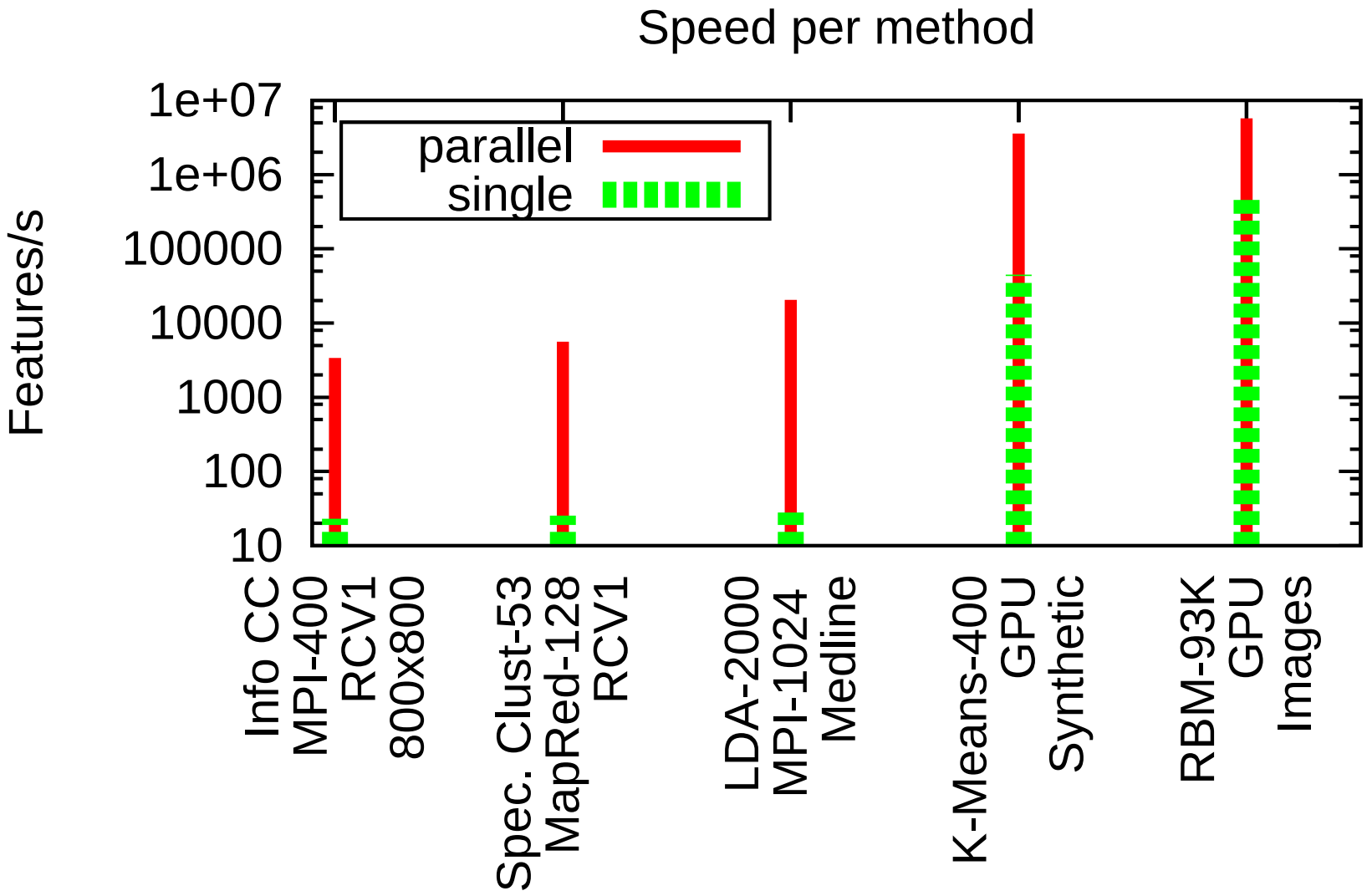
What's in the book?

Parallel Unsupervised Learning Methods

1. Information-Theoretic Co-Clustering with MPI
2. Spectral Clustering using MapReduce as a subroutine
3. K-Means with GPU
4. Latent Dirichlet Analysis with MPI

It's *very* hard to compare different results.

... But let's try


Speed per method
Features/s
1e+07
1e+06
100000
10000
1000
100
10
parallel
single
Info CC
MPI-400
RCV1
800x800
Spec. Clust-53
MapRed-128
RCV1
LDA-2000
MPI-1024
Medline
K-Means-400
GPU
Synthetic
RBM-93K
GPU
Images

Ground Rules

Ginormous caveat: Prediction performance varies wildly depending on the problem–method pair.

Ground Rules

Ginormous caveat: Prediction performance varies wildly depending on the problem–method pair.

The standard: Input complexity/time.

Ground Rules

Ginormous caveat: Prediction performance varies wildly depending on the problem–method pair.

The standard: Input complexity/time.

$\Rightarrow$ No credit for creating complexity then reducing it. (Ouch!)

Ground Rules

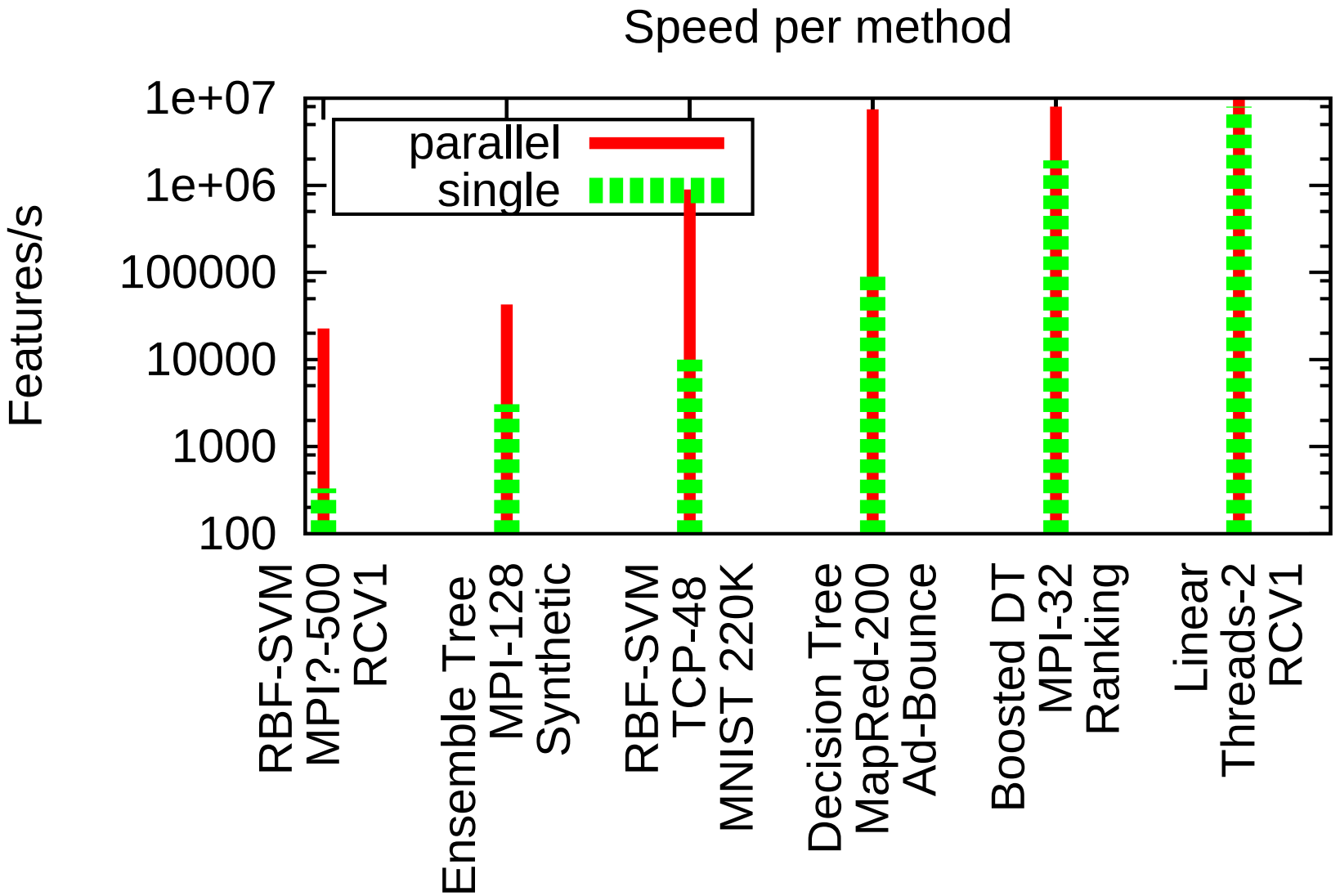
Ginormous caveat: Prediction performance varies wildly depending on the problem–method pair.

The standard: Input complexity/time.

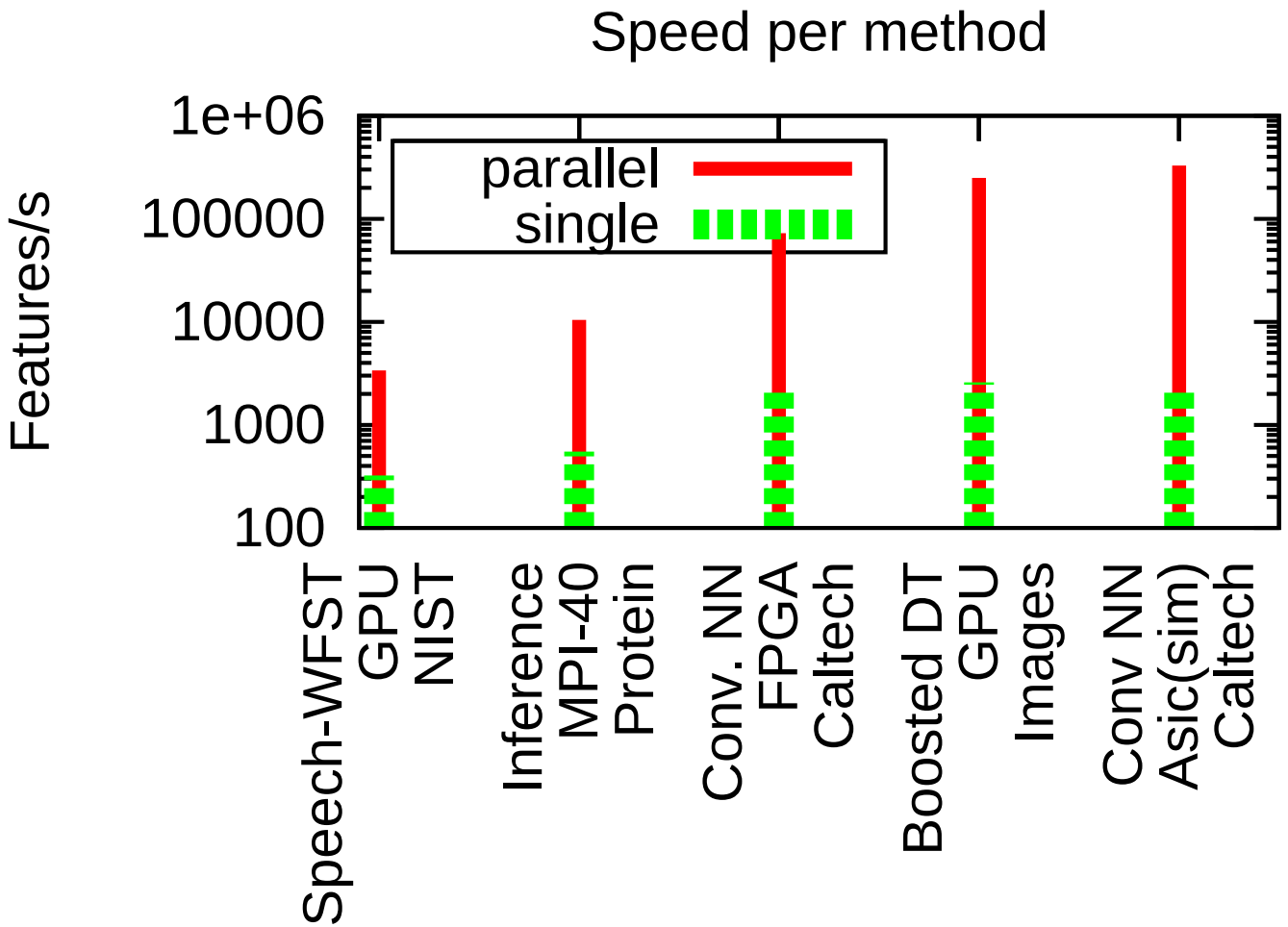
$\Rightarrow$ No credit for creating complexity then reducing it. (Ouch!)

Most interesting results reported. Some cases require creative best-effort summary.

Supervised Training



Supervised Testing (but not training)



My Flow Chart for Learning Optimization

1. Choose an efficient effective algorithm
2. Use compact binary representations.
3. If (Computationally Constrained)
4. then GPU
5. else
    1. If few learning steps
    2. then Map-Reduce
    3. else Research Problem.