Classification

Ensemble Methods 2

Random forests

Given: N training samples, p variables.

Algorithm:

1. For $b = 1$ to B:
   a. Draw a bootstrap sample of size N from training data.
   b. Grow a random-forest tree T_b on the bootstrapped data, by recursively repeating the following steps for each terminal node, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable and split-point among the m.
      iii. Split the node into two child nodes.
2. Output the ensemble of B trees $\{T_b\}$.

Random forests

Given: N training samples, p variables.

Algorithm:

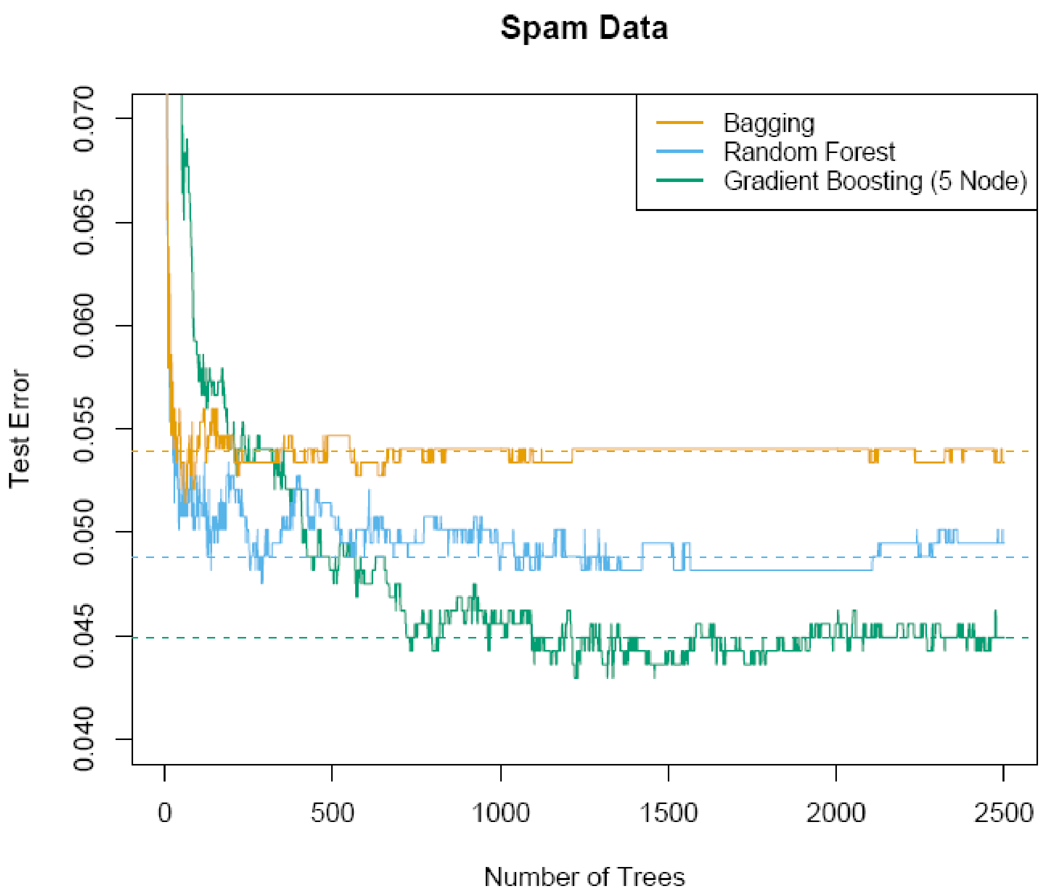
1. For $b = 1$ to B:
   a. Draw a bootstrap sample of size N from training data.
   b. Grow a random-forest tree T_b on the bootstrapped data, by recursively repeating the following steps for each terminal node, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable and split-point among the m.
      iii. Split the node into two child nodes.
2. Output the ensemble of B trees $\{T_b\}$.

Only difference from bagging with decision trees.

– m typically $\leq$ sqrt(p) (even as low as 1)

Random forests

Random forests routinely outperform bagged ensembles, and are often competitive with boosting.



Random forests

- Random forests provide even more reduction of variance than bagged decision trees.
  - But still do not impact bias.
- Benefit appears to be from *de-correlation* of individual trees.
  - Bootstrap samples still have significant correlation.
- Simpler to train and tune than boosting algorithms.

Random forests

- First implemented in FORTRAN by Leo Breiman and Adele Cutler, and the term trademarked by them.

  http://stat-www.berkeley.edu/users/breiman/RandomForests/cc_home.htm

- Commercial distribution licensed exclusively to Salford Systems.

- Lots of open-source implementations in various languages and machine learning packages.

- Available in MATLAB as class TreeBagger (Statistics Toolbox).

Classifier ensembles

- For improved prediction accuracy (vs. single model) often need 100's to 1000's of base classifiers in ensemble

BUT …

- Committee-type classifier ensembles are readily parallelized

Ensemble Cloud Army (ECA)

A Platform for Parallel Processing of Machine Learning Problems in the Amazon Cloud

J. Jeffry Howbert

Insilicos LLC

May 11, 2011

Insilicos LLC: background

- Started 2003
  - Founders: Erik Nilsson, Brian Pratt, Bryan Prazen
- 8 employees
- $4M in grant funding to date (mostly SBIR)
- Focus on mass spec proteomics
  - Software: analysis tools and pipeline
  - Cardiovascular biomarker discovery

ECA project: concept

Machine learning ensembles, trained and used in parallel

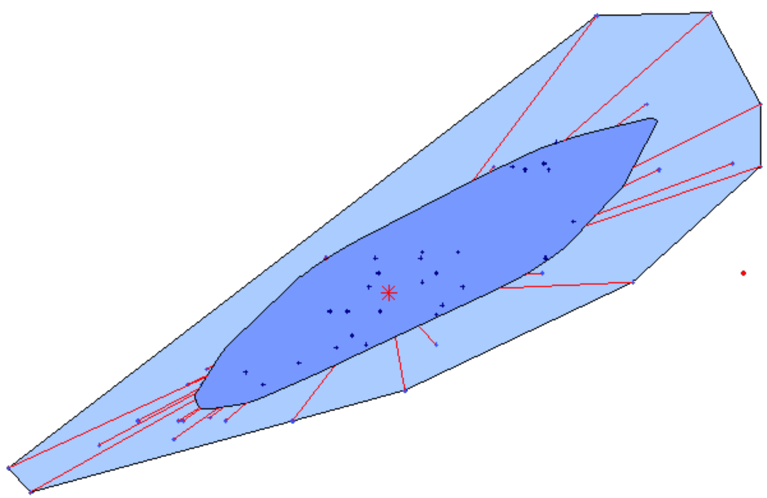
Two performance benefits:

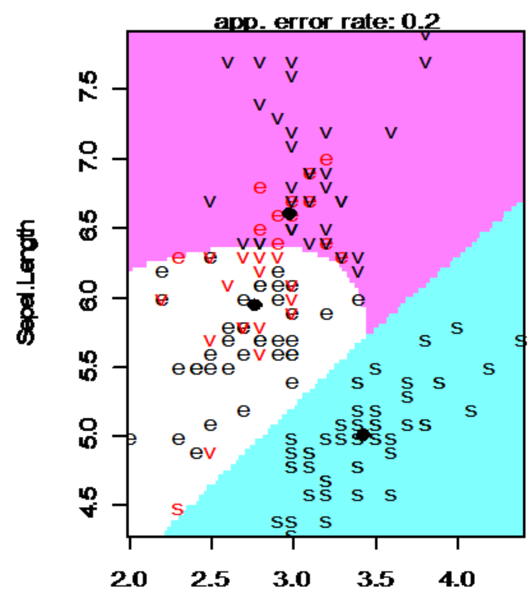
- *Ensemble* of models => *better prediction accuracy* than single model (usually)
- Ensembles are readily *parallelized* => *faster* computation

NOTE: Work to date all on *classifiers*, but is being extended to regression and clustering.

R programming language

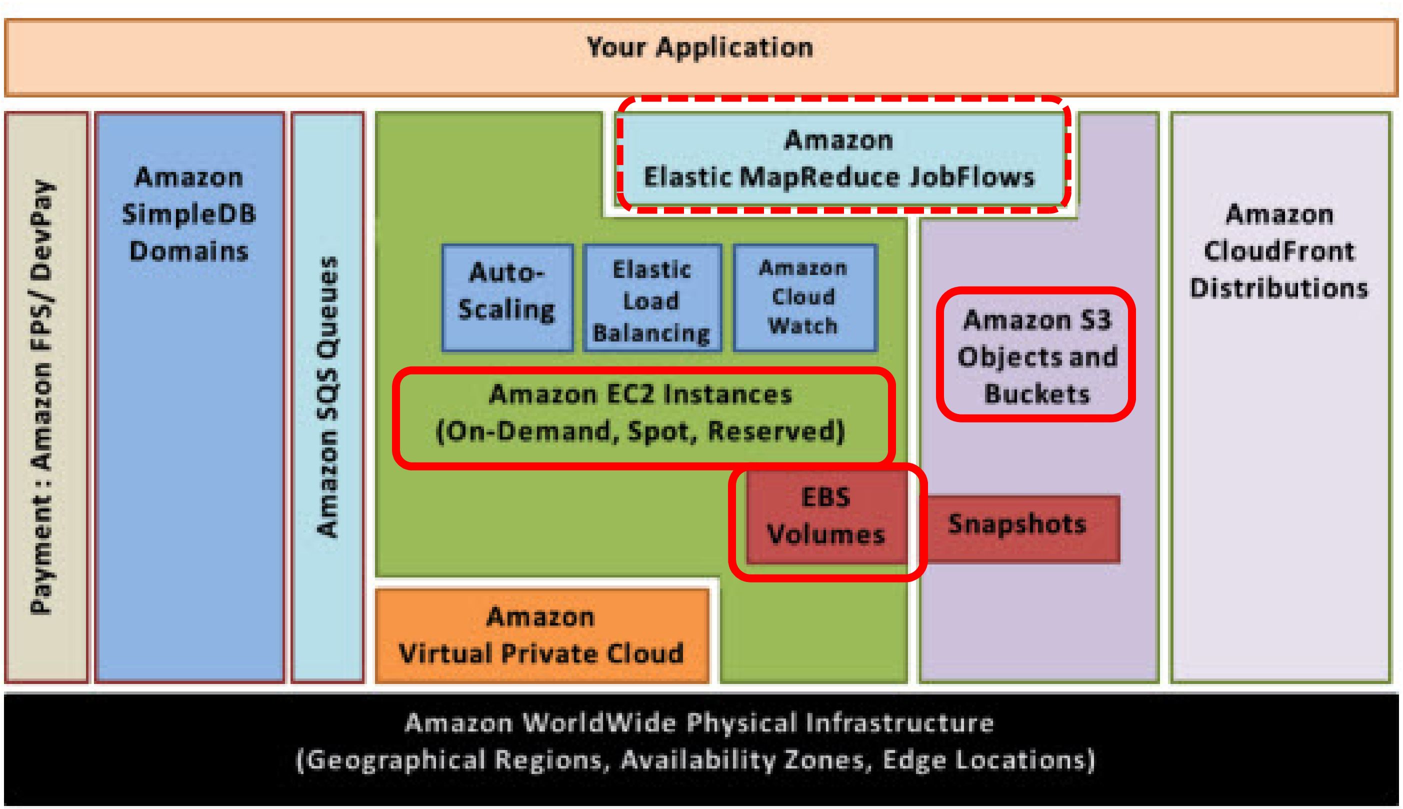
- Functional language for statistical computing and graphics
- *de facto* standard throughout statistics community
- Hundreds of supporting packages
- Open source





Amazon Web Services (AWS)

Basic resources



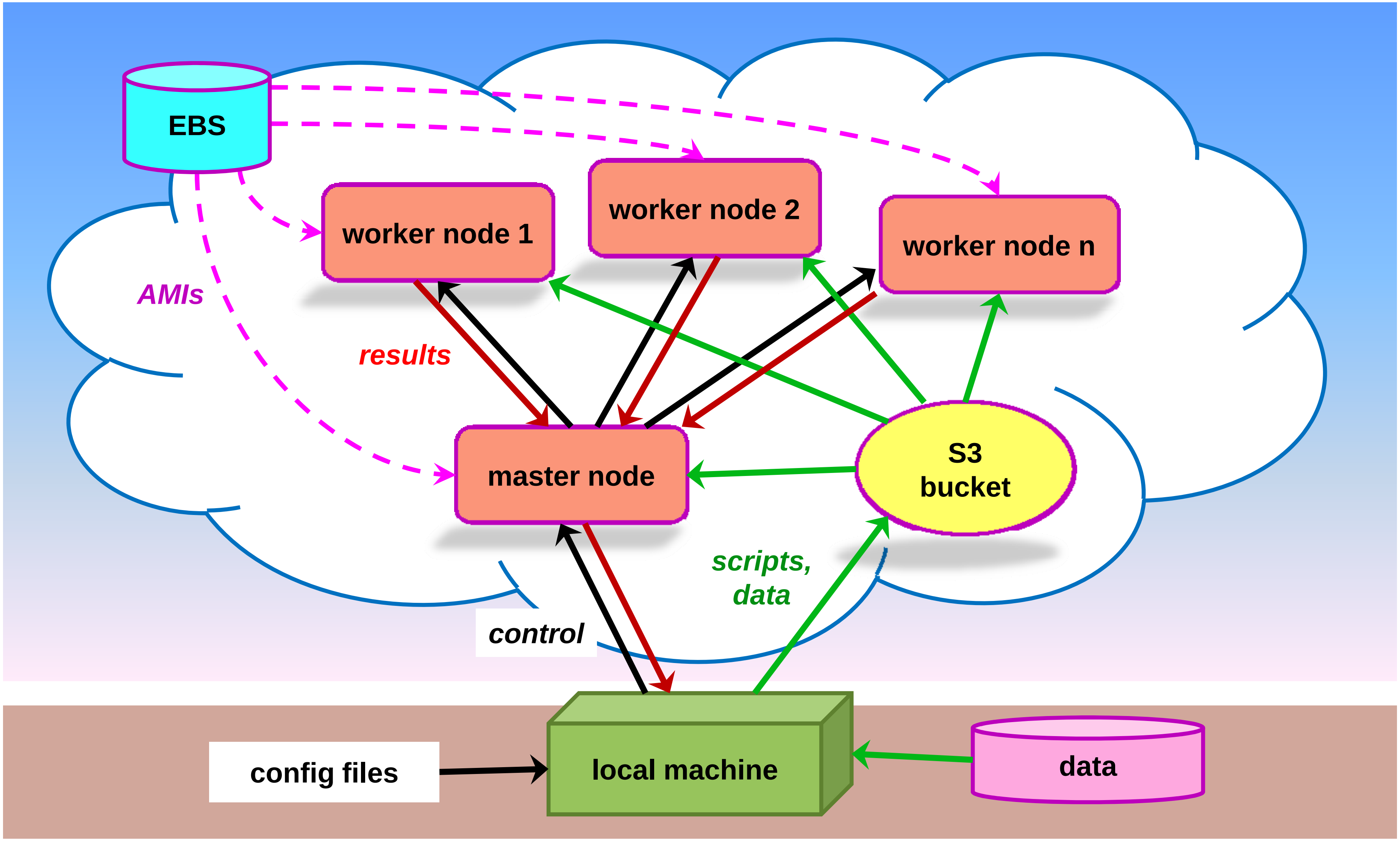
AWS basic resources

- **EC2**: Elastic Compute Cloud
  - Configurable compute nodes
  - Virtual machines in a variety of "sizes"
  - <u>On-demand</u>, reserved, or spot instances
- **S3**: Simple Storage Service
  - Store in named S3 "bucket"
  - Holds unlimited number of objects
  - Any type of object, 1 B to 5 TB in size
  - No file system; put and get using name of object

AWS basic resources

- **EBS**: Elastic Block Store
  - Block level storage volumes from 1 GB to 1 TB
  - Can be attached to any running EC2 instance
  - Persist independently of instances
- **AMI**: Amazon Machine Image
  - Pre-configured virtual machine: OS + apps + tools
  - Loads onto EC2 node at launch
  - Thousands available
  - Can customize own AMIs and save for later use

ECA architecture



ECA hardware components

CLOUD

- EC2 nodes
  - Mostly “small” size
    - 32-bit Intel processor, 1.7 GB RAM, 160 GB hard drive
    - $0.085 / hr
  - Limited use of “large” size (64-bit, faster, more memory, etc.)
- S3 buckets for off-node data storage
- EBS volume to store AMIs

LOCAL MACHINE

- Personal computer (Windows)

ECA software components

- Used only open source components

CLOUD: Amazon Machine Image

- Ubuntu Linux OS
- MPI (message passing interface) – MPICH2
- Python
- R statistical language
- R package Rmpi
  - Allows parallel distribution of calculations to a cluster
  - Communicates via underlying MPI

LOCAL MACHINE: Python

- boto – Python wrapper for AWS API; allows calls to cloud resources
- simplejson – Python parser for JSON-formatted config files

ECA system launch (1)

1) *CLOUD*: pre-existing resources
   – S3 bucket
   – AMI stored in EBS

2) *LOCAL MACHINE*: Python script initiates launch
   – Reads config files (JSON format)
   – Uploads data and R scripts to S3
   – Makes request to AWS for one master node
   – Passes control to master node and waits for results

…. < *job runs autonomously in cloud* > ….

ECA system launch (2)

3) *CLOUD*: Python and bash scripts

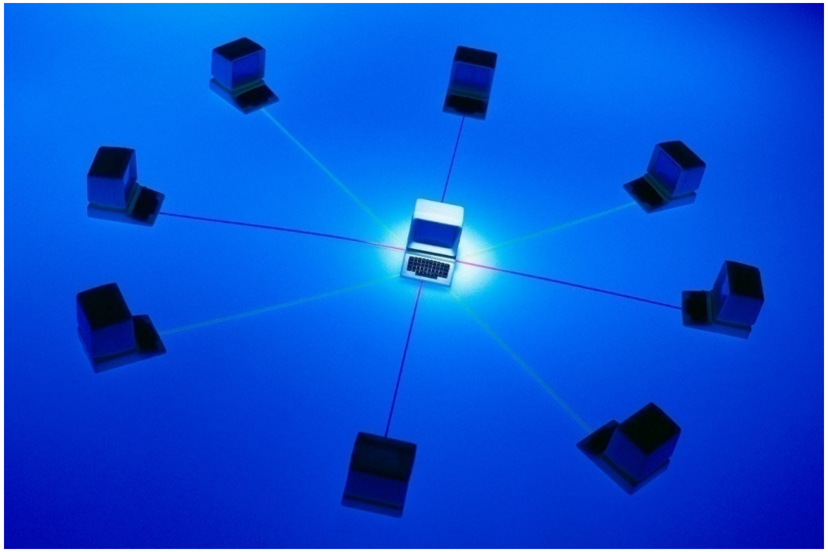
a) Head node:

- Requests desired number of worker nodes from AWS
- Verifies all worker nodes have booted
- Verifies SSH communication with all worker nodes
- Boots MPI demon on all nodes, verifies communication around MPI ring
- Transfers R scripts from S3 bucket to local disk

b) All nodes: transfer data from S3 bucket to local disk

c) Head node: passes control to ensemble R script

Ensemble program flow (1)



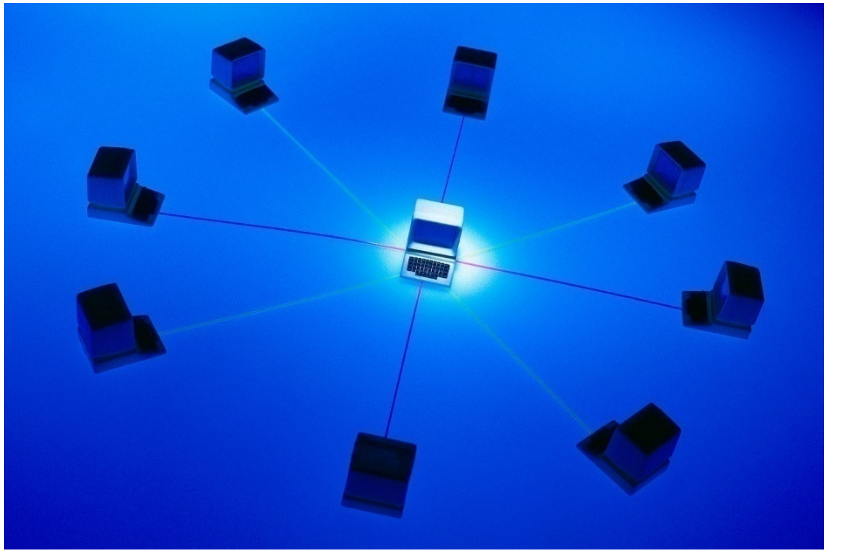
SETUP

- One master node
- Multiple worker nodes
- Master is hub for all communication
  - Bidirectional communication via MPI between master and each worker
  - No worker-worker

R script with all commands for training, testing, etc. on master

Full copy of training and test data on each worker

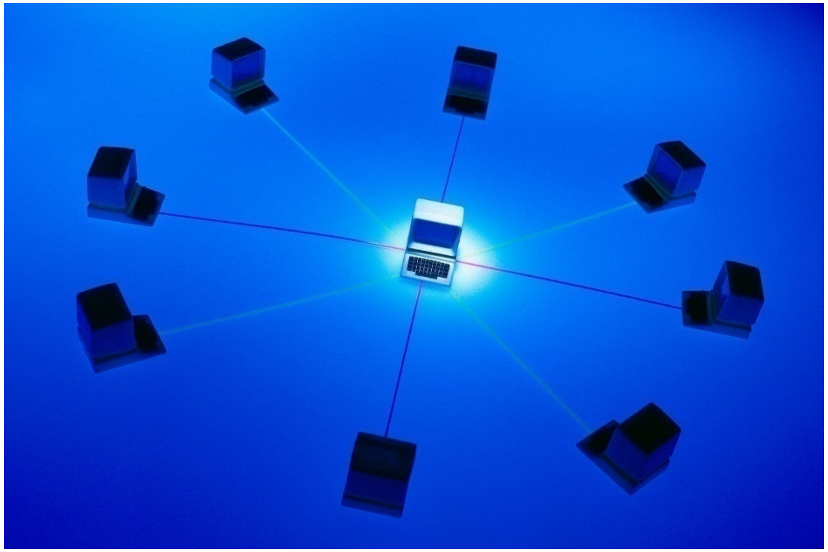
Ensemble program flow (2)



MAIN CYCLE

1. Master sends command to all workers to perform these tasks in parallel:
   a. Create unique partition of training data, using unique random seed
   b. Train a base classifier on partition
   c. Generate class predictions for test data, using trained classifier
2. Workers automatically return predictions to master
3. Master stores predictions
4. Repeats …

Ensemble program flow (3)



END PROCESSING

All by master:

1. Aggregates predictions from all workers over all cycles
2. Computes most commonly predicted class for each instance in test set, outputs that as ensemble prediction

ECA benchmarks

Datasets

Name	Source	Domain	Instances	Features	Feature type(s)	Classes
satimage	UCI	soil types from satellite images	4435 train, 2000 test	36	numeric (0-255)	6
covertype	UCI	forest cover types from cartographic variables	581012	54	10 numeric, 44 binary qualitative	7
jones	Ref. 3	protein secondary structure	209529 train, 17731 test	315	numeric	3

ECA benchmarks

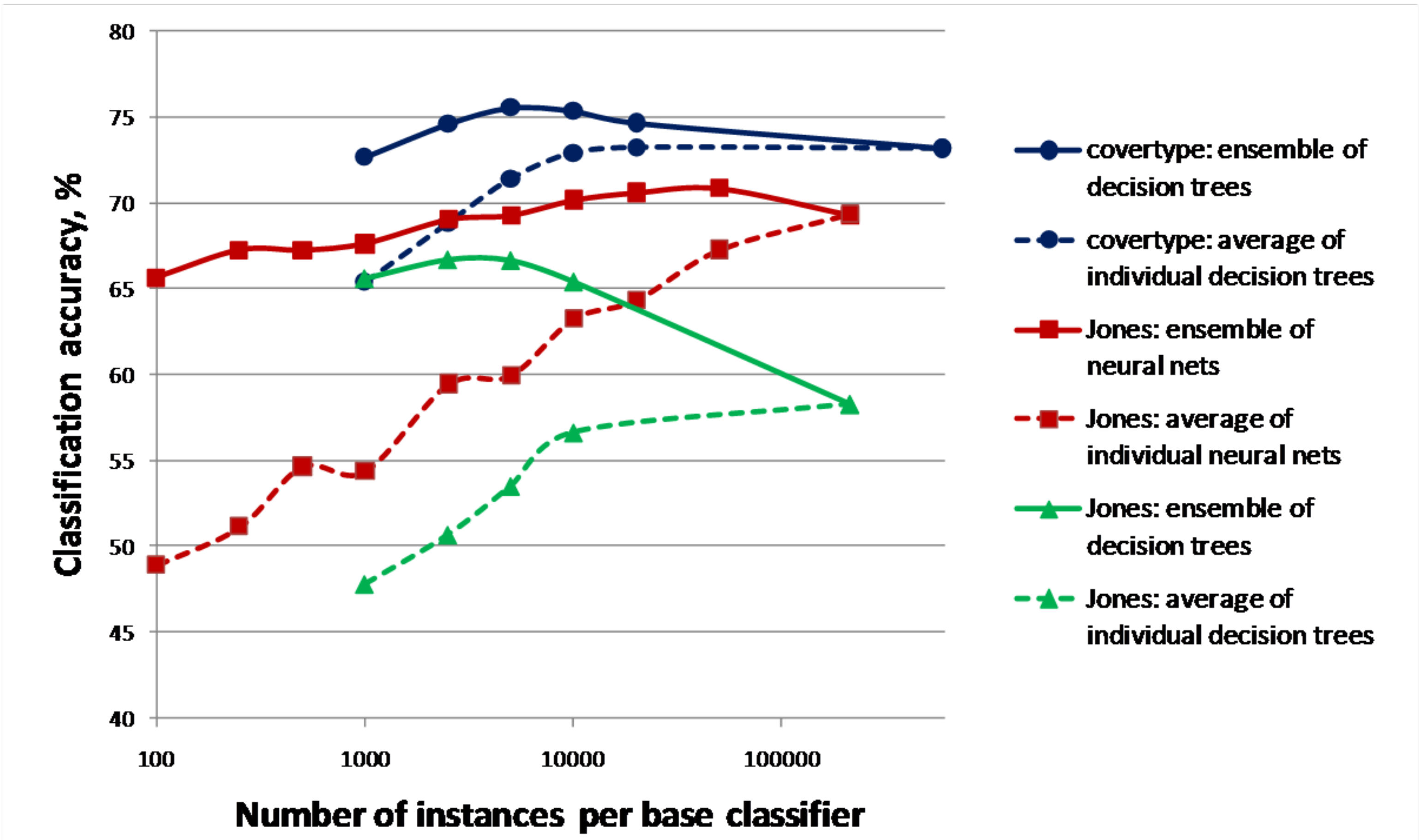
- For ensembles, training subsets must deliver *diversity*, *accuracy*, and *fast computation*.
- For large datasets used with ECA, bootstrap samples are too large for practical computation.
- Instead, much smaller subsets of records are generated by random sampling without replacement.

- From Lecture 3:

  "The key principle for effective sampling is the following:
  - Using a sample will work almost as well as using the entire data set, provided the sample is *representative*.
  - A sample is representative if it has approximately the same distribution of properties (of interest) as the original set of data"

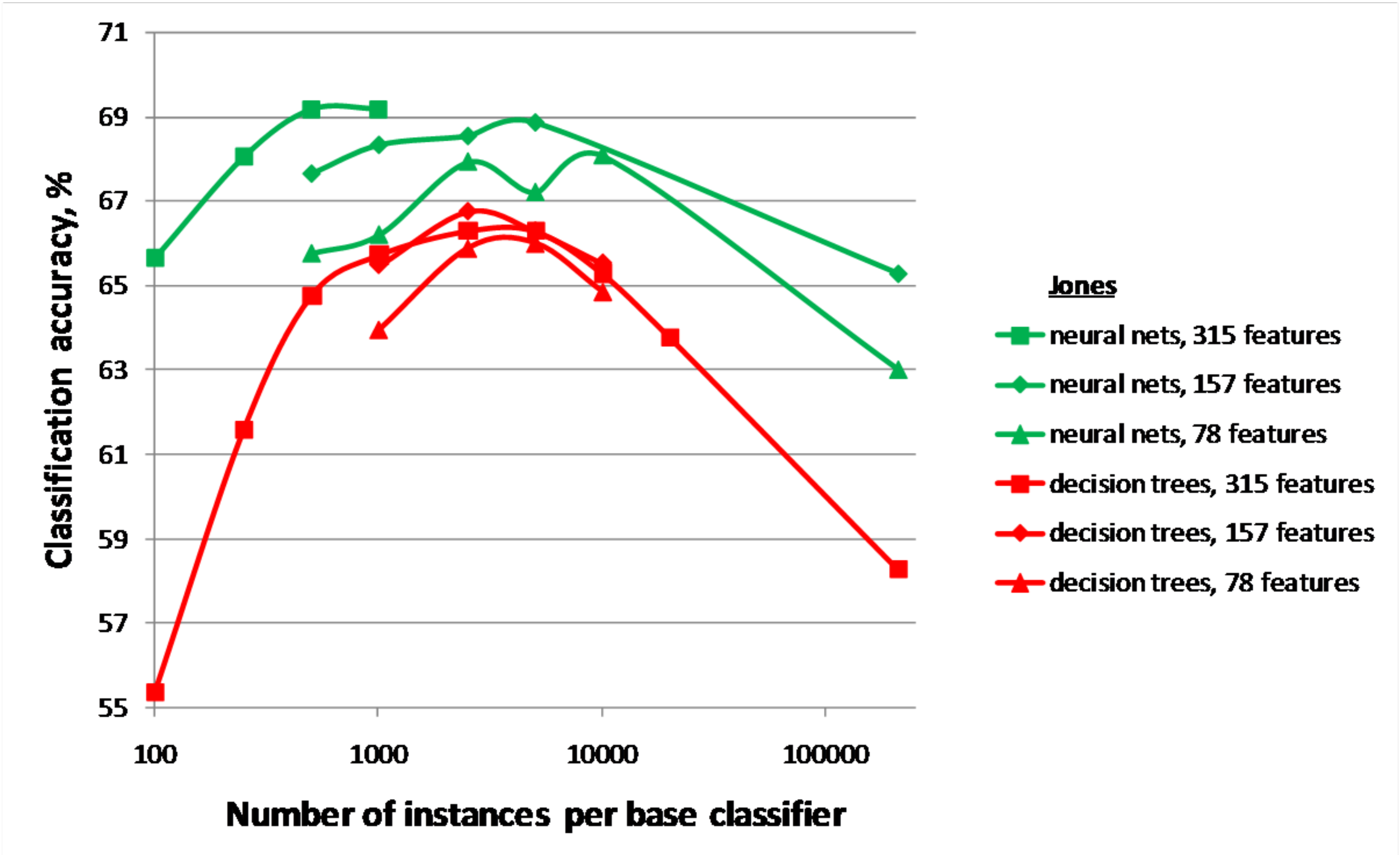
ECA benchmarks

Ensembles have better accuracy than individual component classifiers



ECA benchmarks

Accuracy remains high despite large reduction in features



Amdahl's Law

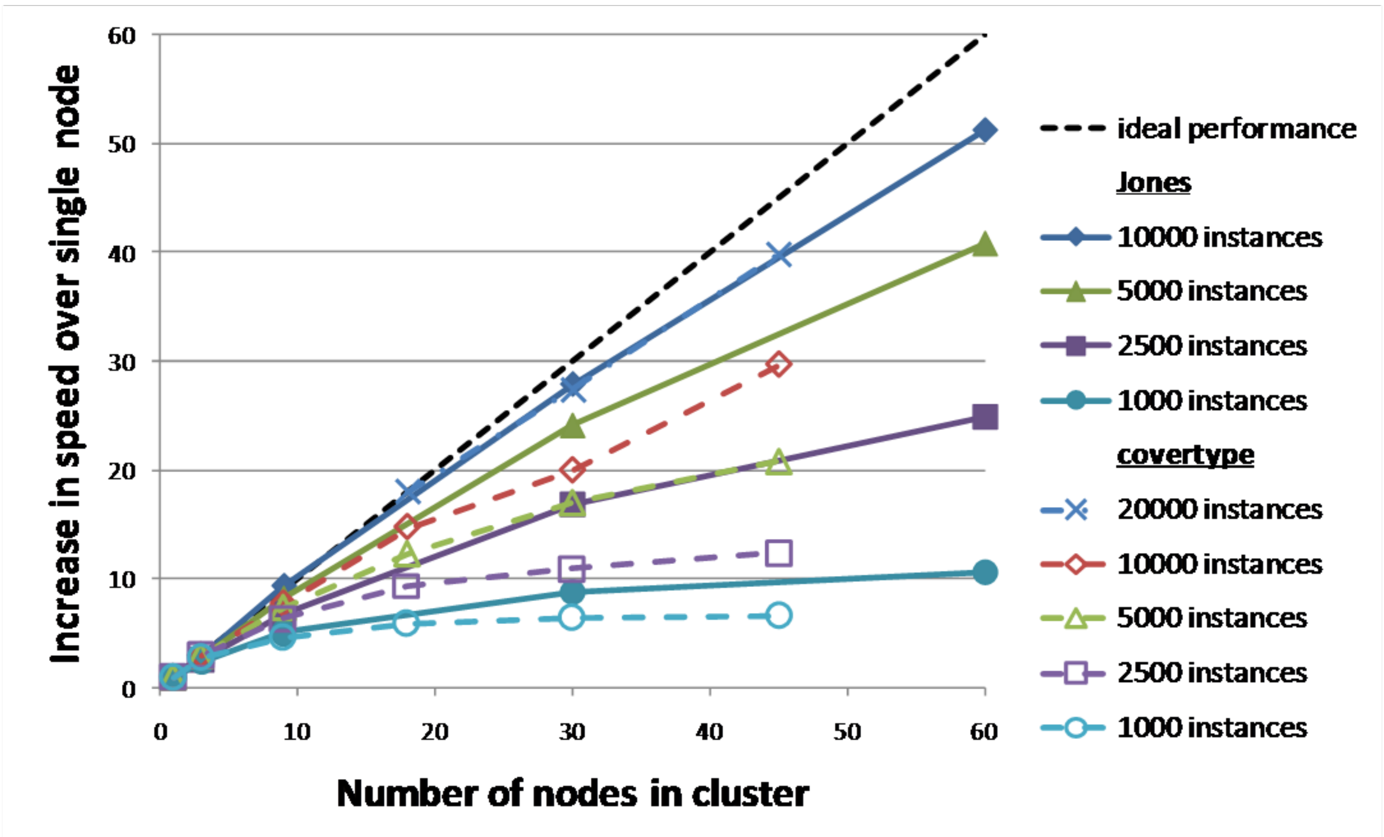
- The potential speedup from parallelization is strictly limited by the portion of the computation that cannot be parallelized.
- Assume proportion *P* of computation can be parallelized, and proportion (1 – *P*) is necessarily sequential. The speedup from parallelizing on *N* processors is:

$$\frac{1}{(1-P)+\frac{P}{N}}$$

- For example, if *P* = 0.9, maximum possible speedup is 10, no matter how large *N* is.

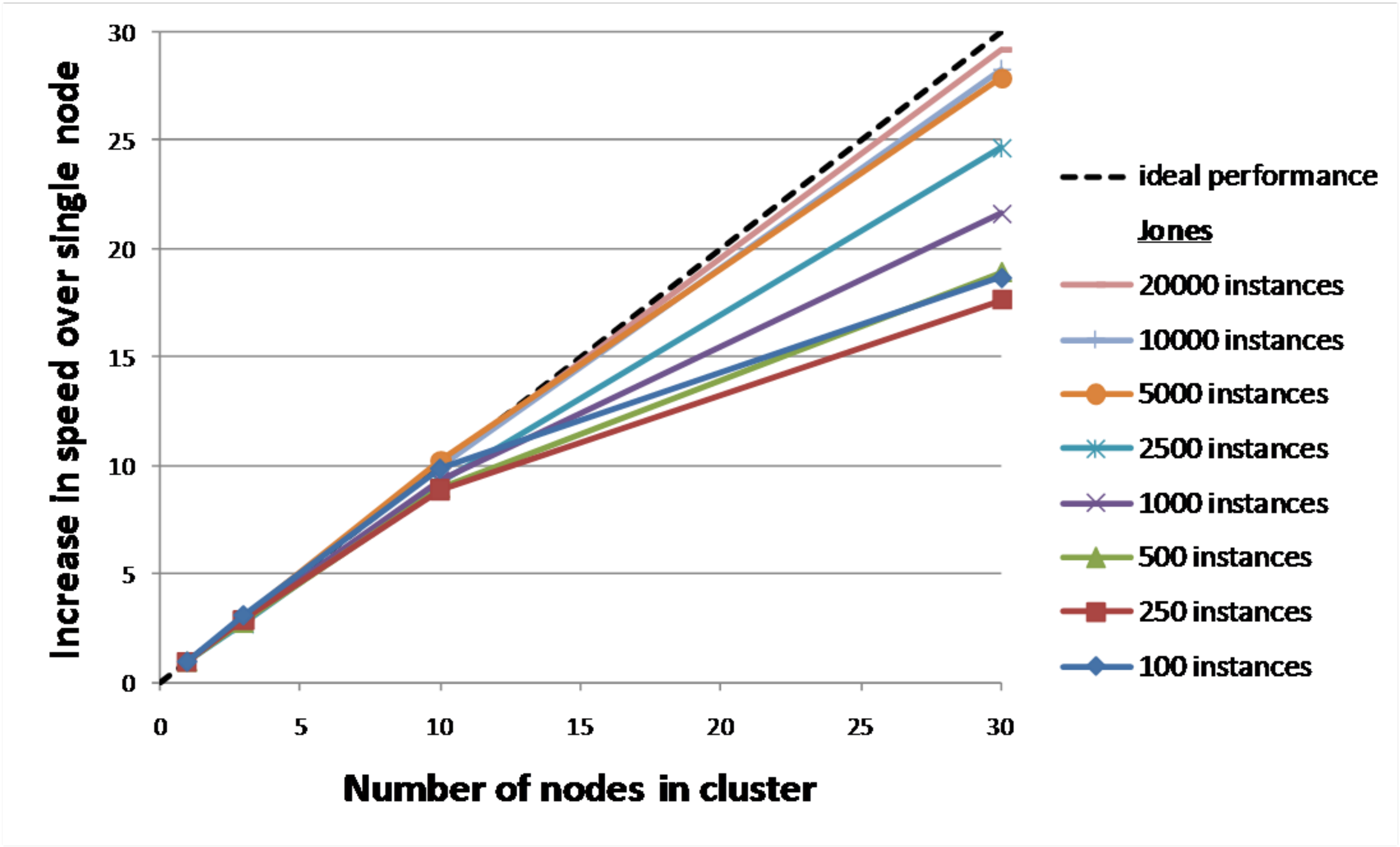
ECA benchmarks

Computational performance: ensembles of decision trees



ECA benchmarks

Computational performance:
ensembles of neural networks



Important lessons (1)

- Large data handling not as critical as expected
  - Best ensemble accuracy associated with smaller partitions (< 5,000 instances)

- Ensembles with small partitions run much faster than those with larger partitions

Important lessons (2)

- Ensembles with small partitions run much faster than single classifier trained on all of data, *and* are more accurate

Number of trees	Instances per tree	Processing mode	Number of nodes	Node type	Runtime	Accuracy, %
1	209529	serial	1	64-bit	2:01:34	58.30
100	2500	serial	1	64-bit	29:54	66.30
180	2500	parallel	60	32-bit	5:44	66.66

Jones dataset, ensemble of decision trees

ECA is open source

RMPI version released on SourceForge

ica.sf.net

Important lessons (3)

- As time went on, AWS's growing popularity led to higher utilization loads, longer latencies for inter-node communication.
- MPI became less and less reliable. Eventually MPI-based clusters on generic EC2 nodes were useless.
- Solutions:
  - Use Hadoop instead of MPI.
  - Rent high-end EC2 nodes designed for communication-intensive clusters.

Occam's Razor

- Given two models with similar generalization errors, one should prefer the simpler model over the more complex model.
- For complex models, there is a greater chance it was fitted accidentally by errors in data.
- Model complexity should therefore be considered when evaluating a model.

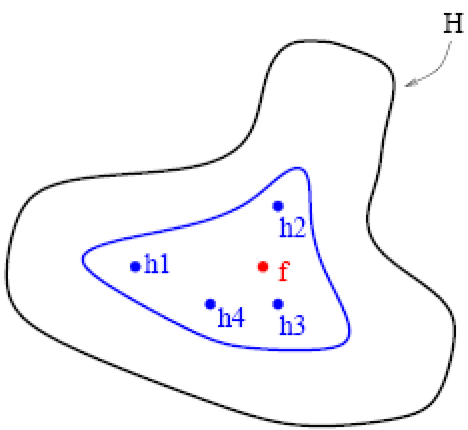
Generalization paradox of ensembles

Ensemble models—built by methods such as *bagging*, *boosting*, and *Bayesian model averaging*—appear dauntingly complex, yet tend to strongly outperform their component models on new data. Doesn't this violate "Occam's razor"—the widespread belief that "the simpler of competing alternatives is preferred"? We argue no: if complexity is measured by function rather than form—for example, according to generalized degrees of freedom (GDF)—the razor's role is restored. On a two-dimensional decision tree problem, bagging several trees is shown to actually have less GDF complexity than a single component tree, removing the generalization paradox of ensembles.

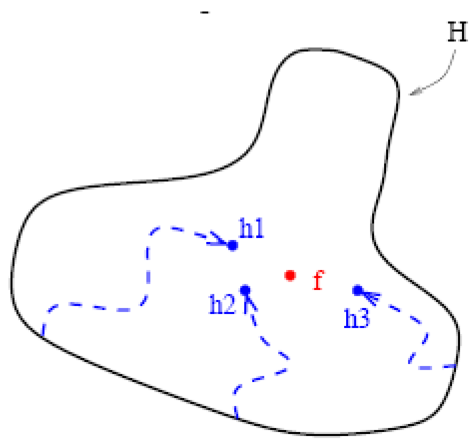
http://www.datamininglab.com/pubs/Paradox_JCGS.pdf

Ensemble methods

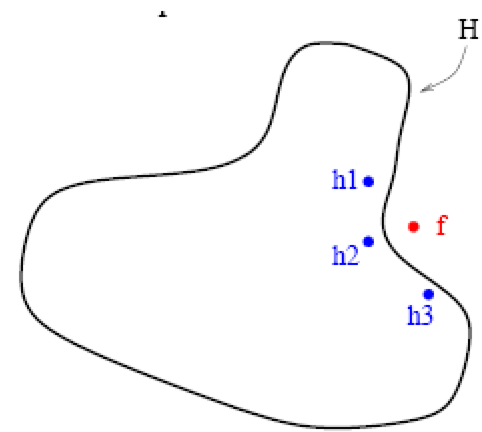
Three fundamental reasons an ensemble may work better than a single classifier



statistical



computational



representational

Tom Dietterich, “Ensemble Methods in Machine Learning” (2000)