

QUICK DISCUSSION ON PARALLELISM

-APPLIED ANALYTICS-

Lecturer: Darren Homrighausen, PhD

INTRODUCTION

```

train = agaricus.train
test = agaricus.test
nround = 10000
bst = xgboost(data = train$data, label = train$label, max.depth = 2,
              eta = 1, nthread = 1, nround = nround, objective =
"binary:logistic",
              print.every.n = nround/10)
pred = predict(bst, test$data)

##
## GBM
##
require(gbm)
data(agaricus.train, package='xgboost')
data(agaricus.test, package='xgboost')
Xtrain = data.frame(as.matrix(agaricus.train$data))
Ytrain = agaricus.train$label
test = agaricus.test
nround = 10000
gbm1 <-
gbm(Ytrain ~ ., data=Xtrain,
     distribution="bernoulli",
     n.trees=200,
     shrinkage=0.1,
     interaction.depth=3,
     bag.fraction = 0.5,
     train.fraction = .9,
     n.minobsinnode = 10,
     cv.folds = 3,
     keep.data=TRUE,
     verbose=TRUE,
     n.cores=2)

```

formula
see the help for other choices
number of trees
shrinkage or learning rate,
0.001 to 0.1 usually work
1: additive model, 2: two-way interactions,
etc.
bag.fraction = 0.5,
train.fraction = .9,
first train.fraction*N used for training
minimum total weight needed in each node
do 3-fold cross-validation
keep a copy of the dataset with the object
don't print out progress
use only a single core (detecting #cores is
error-prone, so avoided here)

```

# 0.001 to 0.1 usually
# 1: additive model, 2:
# subsampling fraction,
# fraction of data for
# first train.fraction*N
# minimum total
# do 3-fold cross
# keep a copy of
# don't print out
# use only a sin

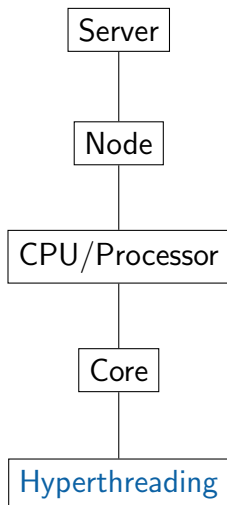
```

inDeviance	ValidDeviance	StepSize	Improve
1.2169	nan	0.1000	0.0837
1.0823	nan	0.1000	0.0670
0.9669	nan	0.1000	0.0578
0.8699	nan	0.1000	0.0486
0.7772	nan	0.1000	0.0465
0.7062	nan	0.1000	0.0354
0.6434	nan	0.1000	0.0311

a good

formula = formula(data), distribution = "bernoulli", data = list, weights, xval.monotone = NULL, n.trees = 100, interaction.depth = 1, n.minobsinnode = 10, shrinkage = 0.1, bag.fraction = 0.5, train.fraction = 0.9, cv.folds = 0, keep.data = TRUE, verbose = "FY", class.strategy = NULL, n.cores = NULL)

DISTRIBUTED COMPUTING HIERARCHY



EXAMPLE: A server might have

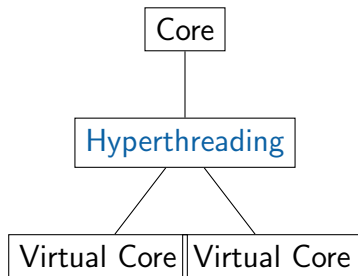
- 64 nodes
- 2 processors per node
- 16 cores per processor
- **hyper threading**

The goal is to somehow allocate a **job** so that these resources are used efficiently

Jobs are composed of **threads**, which are specific computations

HYPERTHREADING

Developed by Intel, Hyperthreading allows for each core to pretend to be two cores



This works by trading off computation and read-time for each core

EMBARRASSINGLY PARALLEL

Suppose we want to make a large number of computations, but it can be broken up into independent chunks

EXAMPLES:

- K-fold CV: Compute the performance estimate for each fold and then combine them together
- Bootstrap draws: We can recompute the statistic for each bootstrap draw
- Matrix multiplication: We can break matrix into chunks of rows and do the multiplication for each chunk
- Split-apply-combine: We can assign each 'group' in `group_by` to a different core
- You are simulating at a large number of parameter settings.
- Grid search optimization....

K-FOLD CV

We split our data up in K non-overlapping chunks $(\mathcal{D}_k, k = 1, \dots, K)$

Now, we can allocate to each core the following:

1. Fit the model on $\mathcal{D}_1, \dots, \mathcal{D}_{k-1}, \mathcal{D}_{k+1}, \dots, \mathcal{D}_K$
2. Get the performance estimate on \mathcal{D}_k, \hat{E}_k

After all the cores are done, we can recombine:

$$\hat{E} = \frac{1}{K}(\hat{E}_1 + \dots, \hat{E}_K)$$

SOME CONSIDERATIONS

- Make sure the problem is large enough. Using parallelism on small problems can **increase** the processing time
(This generally means the processing time is longer than you can accept)
- Pay attention to load balancing
- Make sure the processes are independent
(Example: for K-fold, if you train using one processor and get the performance estimate on another)
- It's best practice to run in 'batch mode' without having an IDE/interpreter open
(For this class, don't worry about this point. This is more for your future careers. Batch mode would be, e.g. **R CMD BATCH script.R &)**