CompSci 590.6

Understanding Data:
Theory and Applications

Lecture 24
Feature Selection in Analytics

Instructor: Sudeepa Roy
Email: sudeepa@cs.duke.edu

Fall 2015

Today's Reading

Materialization Optimizations for Feature Selection Workloads

Zhang-Kumar-Re

SIGMOD 2014 (Best Paper Award winner)

ref. Wiki

What is Feature Selection?

"In machine learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection,

is the process of selecting a subset of relevant features (variables, predictors)

for use in model construction."

i.e. remove redundant and/or irrelevant features.

- What are some examples?

ref. Wiki

Why need Feature Selection?

"Feature selection techniques are used for three reasons:

1. simplification of models to make them easier to interpret by researchers/users
2. shorter training times
3. enhanced generalization by reducing overfitting(formally, reduction of variance)"

This work: Motivation

- DB industry wants to support Analytics
- A crucial step is Feature Selection
  - which will be used to build a statistical model
  - helps an analyst understand and explore data
- Need to improve the <u>efficiency</u> of the feature selection process
- Presents COLUMBUS
  - data-processing system to support the enterprise feature-selection process

Background and Basis

- They interviewed analysts from enterprises who spend a lot of time in feature selection
  - insurance company
  - consulting firm
  - major db vendor's analytics customer
  - major e-commerce firm
- Feature selection is interactive
  - features can or cannot be selected due to several factors
  - e.g. statistical performance, explanatory power, legal reasons, etc.
- The analysts have to write low-level code
  - e.g. in R
  - sometimes R libraries for standard feature selection tasks

RELs and ROPs

- REL = R-Extension Layers

- Almost all db engine ships a product with some R extension
  - Oracle – ORE (Oracle R Enterprise)
  - IBM – SystemML (declarative large scale ML)
  - SAP – HANA
  - Hadoop/Teradata -- Revolution Analytics (open source)

- ROP = REL Operations
  - matrix-vector multiplication or determinants
  - scaling ROPs is a recent industrial challenge

ROP Optimization Limitations

- Missed opportunities for reuse and materialization
- Selecting materialization strategy is difficult for an analyst
  - depends on the reuse opportunities, error tolerance, data size, parallelism etc.
  - will vary across datasets for the same task

Key Ideas (details later)

- subsampling
- transformation materialization
- model caching

COLUMBUS

- Support feature selection (FS) dialogue
- Identify and use existing and novel optimizations for FS workloads as data management problems
- Develop a cost-based optimizer

COLUMBUS: Does and Doesn't

- Does
  - compiles and optimizes an extension of R for FS
  - compiles this language into a set of REL ops (ROPs)
    - implemented by language extenders ORE, Revolution Analytics etc
  - compiles into the most common ROPs

- Does NOT
  - optimize the execution of these ROPs
    - already studied and implemented

COLUMBUS programs

- User expresses FS program as a set of high-level constructs
- Language is a strict superset of R
  – can use the full power of R

Example Program

```
e  = SetErrorTolerance(0.01)              # Set Error Tolerance
d1 = Dataset("USCensus")                  # Register the dataset
s1 = FeatureSet("NumHouses", ...)         # Population-related features
l1 = CorrelationX(s1, d1)                 # Get mutual correlations
s1 = Remove(s1, "NumHouses")              # Drop the feature "NumHouses"
l2 = CV(lsquares_loss, s1, d1, k=5)       # Cross validation (least squares)
d2 = Select(d1,"Income >= 10000")         # Focus on high-income areas
s2 = FeatureSet("Income", ...)            # Economic features
l3 = CV(logit_loss, s2, d2, k=5)          # Cross validation with (logit loss)
s3 = Union(s1, s2)                        # Use both sets of features
s4 = StepAdd(logit_loss, s3, d1)          # Add in one other feature
Final(s4)                                 # Session ends with chosen features
```

Figure 2: Example Snippet of a Columbus Program.

Data Types

- Three major data types

1. A data set
   - a relational table R(A1, . . . , Ad)
2. A feature set F
   - a subset of the attributes F $\subseteq$ {A1, . . . , Ad}.
3. A model for a feature set
   - a vector that assigns each feature a real-valued weight

COLUMBUS Operations

1. Data Transformation Operations
   – produce new data sets;
2. Evaluate Operations
   – evaluate data sets and models
3. Regression Operations
   – produce a model given a feature set
4. Explore Operations
   – produce new feature sets

	Logical Operators
Data Transform	Select, Join, Union, ...
Evaluate	Mean, Variance, Covariance, Pearson Correlations Cross Validation, AIC
Regression	Least Squares, Lasso, Logistic Regression
Explore	Feature Set Operations Stepwise Addition, Stepwise Deletion Forward Selection, Backward Selection

Figure 3: Summary of Operators in Columbus.

1. Data Transform

- standard data manipulations to slice and dice
  - select, join, union
- Columbus is only aware of the schema and cardinality of these operations
  - these operations are executed and optimized directly using a standard RDBMS or main-memory engine
- A **data frame** in R is used for storing data tables. It is a list of vectors of equal length.
  - n = c(2, 3, 5)
  - s = c("aa", "bb", "cc")
  - b = c(TRUE, FALSE, TRUE)
  - df = data.frame(n, s, b)   # df is a data frame containing three vectors n, s, b
- In R, the frames can be interpreted either as a table or an array in the obvious way.
  - These two representations are mapped from the one to the other freely

2. Evaluate

- Obtain various numeric scores
  - given a feature set including descriptive scores for the input feature set
  - e.g., mean, variance, Pearson correlations, cross-validation error (e.g., of logistic regression), and Akaike Information Criterion (AIC)

- Columbus can optimize these calculations by batching several together

3. Regression

- Obtain a model given a feature set and data
  - e.g., models trained by using logistic regression or linear regression
  - The result of a regression operation is often used by downstream “explore” operations, which produces a new feature set based on how the previous feature set performs
  - These operations also take a termination criterion
    - like R
    - either the number of iterations or until an error criterion is met

4. Explore

- enable an analyst to traverse the space of feature sets
  - e.g. a StepDrop operator takes as input a data set and a feature set
  - outputs a new feature set that removes one feature from the input
  - by training a model on each candidate feature set
- optimizations leverage the fact that these operations operate on features in bulk

Basic Blocks

- A user's program is compiled into a DAG (dataflow graph) with two types of nodes

1. R functions
   - opaque to COLUMBUS
2. Basic blocks
   - unit for optimization
   - Formal defn next through "Tasks"

Task

- A task is a tuple $t = (A, b, \ell, \epsilon, F, R)$
  - $A \in R^{N \times d}$ is a data matrix
  - $b \in R^N$ is a label (or target)
  - $\ell : R^2 \rightarrow R^+$ is a loss function
  - $\epsilon > 0$ is an error tolerance
  - $F \subseteq [d]$ is a feature set
  - $R \subseteq [N]$ is a subset of rows

Task and Basic Block

- A task t = (A, b, ℓ,ϵ, F, R) specifies a regression problem of the f

$$L_t(x) = \sum_{i \in R} \ell(z_i, b_i) \;\; s.t. \;\; z = A\Pi_F x$$

Here Π_F is the axis-aligned projection that selects the columns or feature sets specified by F.[4] *Denote an optimal solution of the task $x_*(t)$ defined as*

$$x_*(t) = \operatorname*{argmin}_{x \in \mathbb{R}^d} L_t(x)$$

---

[4]For $F \subseteq [d]$, $\Pi_F \in \mathbb{R}^{d \times d}$ where $(\Pi_F)_{ii} = 1$ if $i \in F$ and all other entries are 0.

Our goal is to find an $x(t)$ that satisfies the error[5]

$$\|L_t(x(t)) - L_t(x_*(t))\|_2 \leq \epsilon$$

A basic block, B, *is a set of tasks with common data (A, b) but with possibly different feature sets $\bar{F}$ and subsets of rows $\bar{R}$.*

Example

1. e = SetErrorTolerance(0.01)
   - # Set Error Tolerance
6. l2 = CV(lsquares_loss, s1, d1, k=5)
   - # Cross validation (least squares)

EXAMPLE 2.1. *Consider the 6^{th} line in Figure 2, which specifies a 5-fold cross validation operator with least squares over data set d_1 and feature set s_1.* COLUMBUS *will generate a basic block* B *with 5 tasks, one for each fold. Let $t_i = (A, b, l, \epsilon, F, R)$. Then, A and b are defined by the data set d_1 and $l(x, b) = (x - b)^2$. The error tolerance ϵ is given by the user in the 1^{st} line. The projection of features $F = s_1$ is found by a simple static analysis. Finally, R corresponds to the set of examples that will be used by the i^{th} fold.*

Executing a COLUMBUS program

1. parser
   – The output of the parser is as a DAG, in which the nodes are either basic blocks or standard ROPs, and the edges indicate data flow dependency

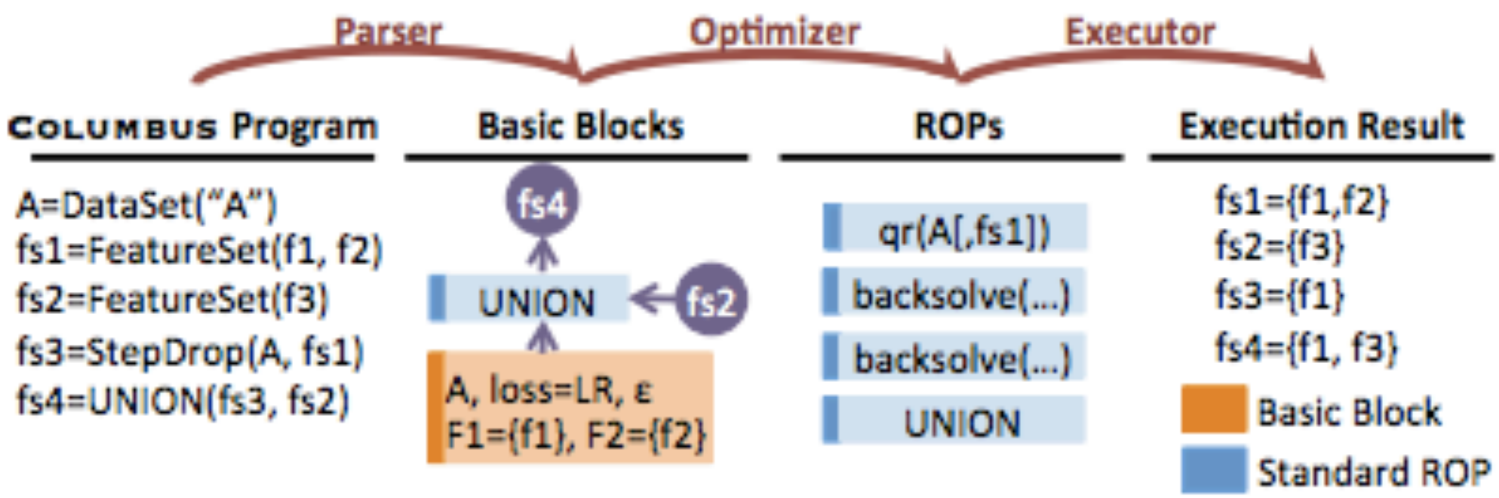

Figure 4: Architecture of Columbus.

Executing a COLUMBUS program

2. optimizer
   – responsible for generating a "physical plan"
   – defines which algorithms and materialization strategies are used for each basic block
   – The optimizer may also merge basic blocks together -- called multiblock optimization

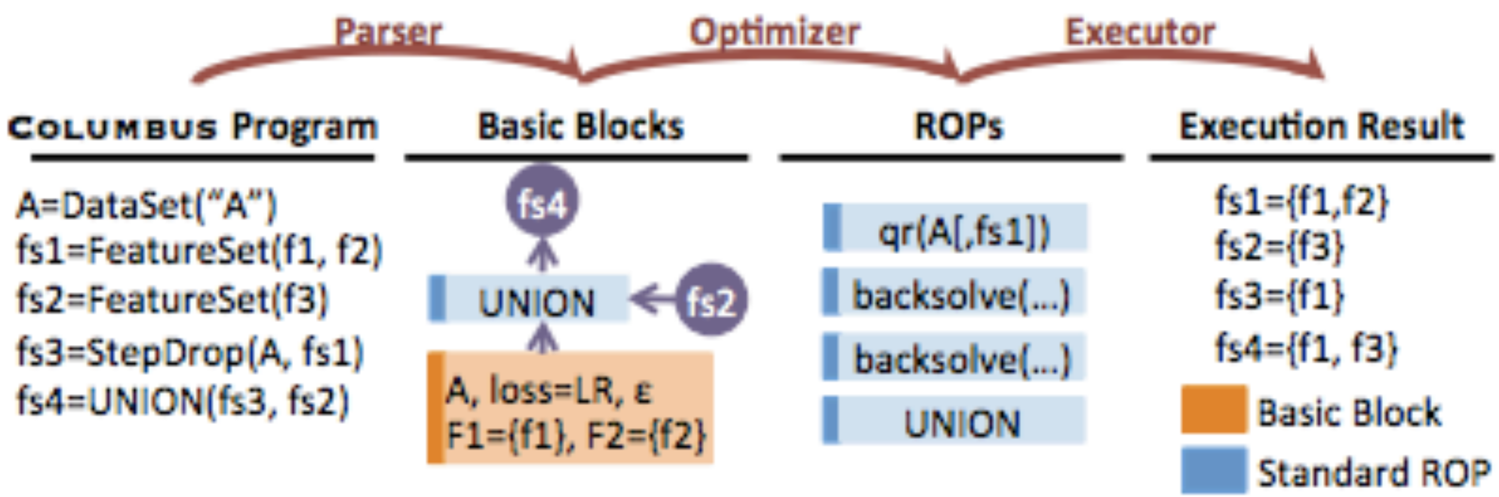

Figure 4: Architecture of Columbus.

Executing a COLUMBUS program

3. executor
   – manages the interaction with the REL and issues concurrent requests.

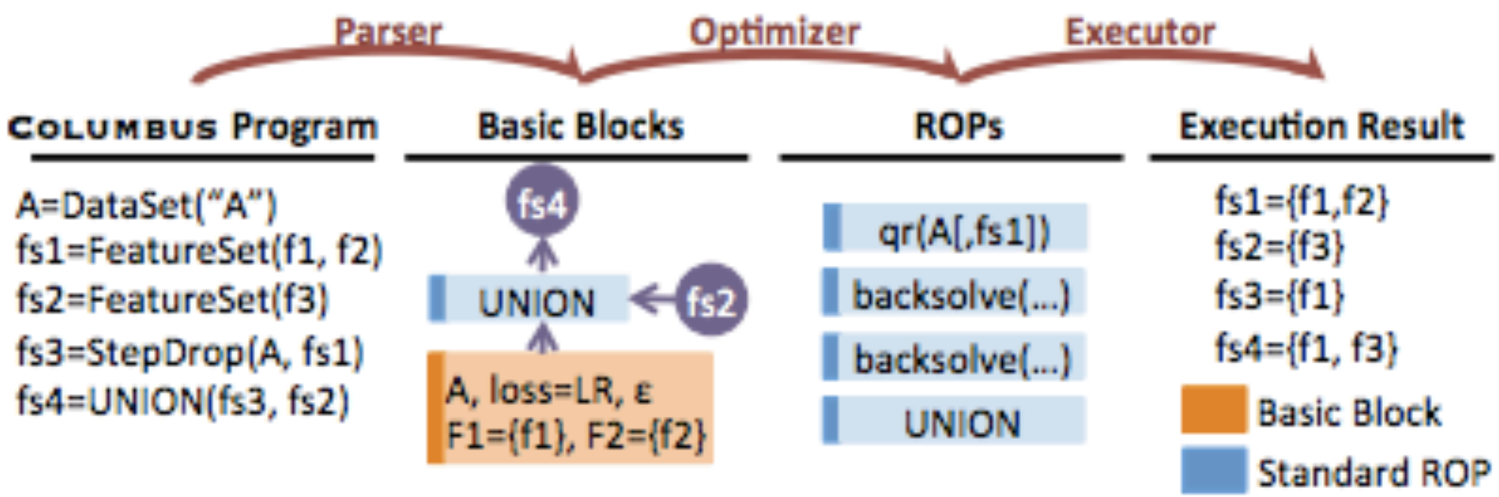

Figure 4: Architecture of Columbus.

Optimizer

- Consider least-squares cost first
- Optimization axes
  1. Error tolerance ϵ
  2. Sophistication
     - e.g. loss function, #feature sets, #rows selected
  3. Reuse
     - degree to reuse computation
     - depends on amount of overlap of features and #threads available

A Single, Linear Basic Block

1. classical database optimizations
   – unaware of the FS process
2. sampling-based optimizations
3. transformation-based optimizations
   – leverage FS process/regression

- assume least-square loss $\ell(x, b) = (x-b)^2$
- Goal: compile the basic block into a set of ROPs

1. Classical Database Optimizations

- $F^{\cup} = \bigcup_{F \in \mathbf{F}} F$, $R^{\cup} = \bigcup_{R \in \mathbf{R}} R$ in the basic block
- Matrix A may contain more columns than $F^{\cup}$ and more rows than $R^{\cup}$
  - Then project away these extra rows and columns
  - Analogous to materialized views of queries that contain selections and projections

Lazy and Eager strategies

- The Lazy strategy will compute projections at execution time
- Eager will compute these projections at materialization time
  - use them directly at execution time
- Eager has a higher materialization cost than Lazy
- Lazy has a slightly higher execution cost than Eager
  - one must subselect the data
- If there is ample parallelism (#threads >= #feature sets), then Lazy dominates
- Selected by cost-based methods
  - If there are disjoint feature sets $F1 \cap F2 = \varnothing$, then it may be more efficient to materialize these two views separately
  - NP-hard
  - heuristics: split into disjoint sets

Sampling-Based Optimizations

- Saves time because one is operating on a smaller dataset

- can be modeled by adding a subset selection ($R \in \bar{R}$) to a basic block

- two popular methods

1. naïve random sampling
2. importance-sampling method called coresets

Naïve Sampling

- Matrix A has N rows and d columns
- Select some fraction of the N rows (say 10%)
- The cost model for both materialization and its savings of random sampling is straightforward
  - same job only on a smaller matrix

Coresets

- Each row is sampled from $A \in R^{Nxd}$ with different probability
- Sample size is
  - proportional to d
  - independent of N
  - d << N
  - e.g. sample size $m > 2/\epsilon^2 \, d \log d$
- As d increases or ϵ decreases, becomes costlier than naïve sampling
  - when N = d, no advantages
  - there is a cross-over point

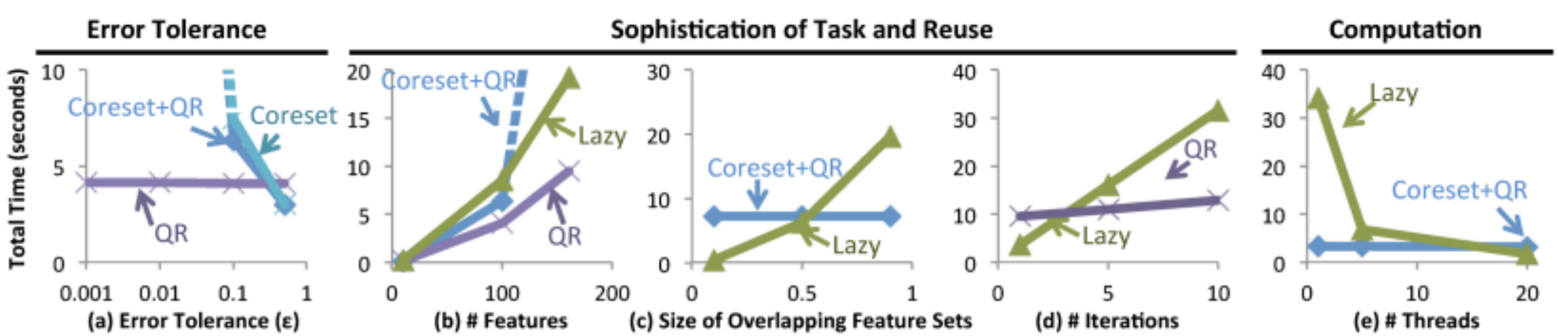

Figure 6: An Illustration of the Tradeoff Space of Columbus, which is defined in Section 3.

Transformation-Based Optimizations: QR

- Decomposition methods to solve repeated least-squares efficiently
  - use across many different feature sets

- Thin QR Factorization
  - The QR decomposition of a matrix $A \in R^{N \times d}$ is a pair of matrices (Q, R)
  - where $Q \in R^{N \times d}$, $R \in R^{d \times d}$, and $A=QR$
  - Q is an orthogonal matrix, i.e., $Q^T Q = I$
  - R is upper triangular.

QR in COLUMBUS

- Solve $Ax = b$
- $QRx = b$
- $Rx = Q^T b$
- Use the property that R is upper triangular
- Does not need computing inverse of R
- Running time from cubic to quadratic

Transformation-Based Optimizations/QR

Tradeoffs

- QR's materialization cost is similar to importance sampling.
- QR can be much faster than coresets
  - techniques can also be combined, further modifies the optimal tradeoff point.
  - QR does not introduce errors like sampling methods

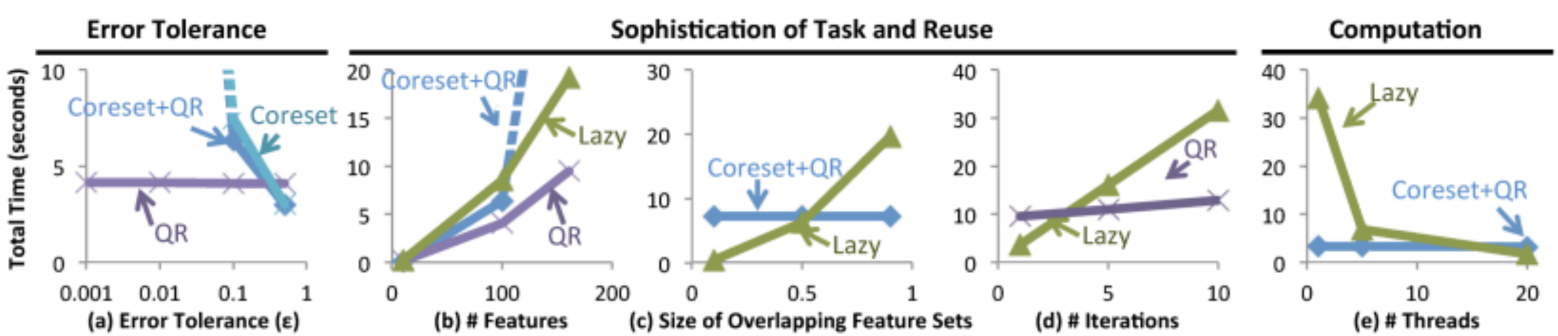

Figure 6: An Illustration of the Tradeoff Space of Columbus, which is defined in Section 3.

In the paper..

- Single, Non-Linear Basic Block
- Non-linear loss functions

Warm-starting by Model Caching

- FS workloads solve a model after having solved many similar models

- Three situations where these similar models can be partially reused:
  1. Down-sample the data, learn a model on the sample, and then train a model on the original data
  2. Perform stepwise removal of a feature in feature selection -- the "parent" model with all features is already trained
  3. Examine several nearby feature sets interactively.

- Difficult for an analyst to implement reuse effectively

Warm-starting by Model Caching

- Columbus can use warm-start to achieve up to 13x performance improvement for iterative methods without user intervention

- Given a cache of models, to choose a model:
  – computing the loss of each model in the cache on a sample of the data is inexpensive
  – select the model with the lowest sampled loss
  – To choose models to evict, simply use an LRU strategy

Multi-block Optimization

- Across blocks:
  - We need to decide on how coarse or fine to make a basic block
  - we need to execute the sequence of basic blocks across the backend.

Multi-block Logical Optimization

- e.g. Cross validation is merged into a single basic block
- Greedily improve the cost
- The problem of deciding the optimal partitioning of many feature sets is NP-hard
- Heuristics

Cost-based Execution

- The executor of Columbus executes ROPs
  - by calling the required database or main-memory backend.
  - responsible for executing and coordinating multiple ROPs in parallel
  - simply creates one thread to manage each of these ROPs
- The actual execution of each physical operator is performed by the backend statistical framework
  - e.g., R or ORE
- Experimented for the tradeoffs of how coarsely or finely to batch the execution
  - NP-hard, but a simple greedy strategy was within 10% of the optimal schedule obtained by a brute-force search
  - batch as many operators as possible, i.e., operators that do not share data flow dependencies

Conclusion

- Columbus is the first system to treat the feature selection dialogue as a database systems problem
- Gives a declarative language for feature selection,
  - informed by conversations with analysts over the last two years
- Observed that there are reuse opportunities in analysts' workloads
  - not addressed by today's R backends.
- Simple materialization operations could yield orders of magnitude performance improvements
- FS may be a pressing data management problem with analytics popularity
  - more in the next class