Big Data: Big N or Very Large Sample Case

VC

May 13, 2016

Examples of Very Big Data

- Congressional record text, in 100 GBs
- Nielsen's scanner data, 5TBs
- Medicare claims data are in 100 TBs
- Facebook 200,000 TBs

Map Reduce & Hadoop

The basic idea is that you need to divide work among the cluster of computers since you can't store and analyze the data on a single computer.

Simple but powerful algorithm framework. Released by Google around 2004; Hadoop is an open-source version.

Map-Reduce algorithm has the following steps:

1. Map: processes "chunks" of data to produce "summaries"
2. Reduce: combines "summaries" from different chunks to produce a single output file

Formally, if data $D = \cup_{i=1}^{K} D_i$,

$$f(D) = R(M(D_1), ..., M(D_K)),$$

where

- $M(\cdot)$ is the mapper,
- $R(\cdot)$ is the reducer.

Examples

- Count words in doc i, D_i. Map: $M : D_i \mapsto M(D_i)$, set of $(word, count)$ pairs, also called "key-value" pairs. Reduce: Aggregate $\{M(D_i)\}_{i=1}^{K}$ by summing over *count* within *word*.
- Records at hospital i, D_i. Map: $M : D_i \mapsto M(D_i)$, say records for patients who are 65+. Reduce: take union over elements of $\{M(D_i)\}$, i.e.

$$f(D) = R(\{M(D_i)\}_{i=1}^{k}) = \cup_{i=1}^{k}\{M(D_i)\}.$$

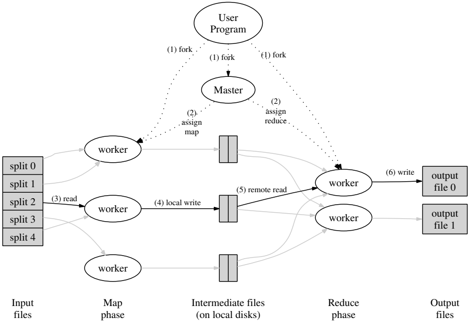
- Compute the minimum of data $D = \cup_{i=1}^{k} D_i$, where D_i are vectors. Map: $M : D_i \mapsto M(D_i) = min(D_i)$. Reduce: take minimum over elements of $\{M(D_i)\}$, i.e.

$$f(D) = R(\{M(D_i)\}_{i=1}^{k}) = \min_{i=1}^{k}\{\min(D_i)\}.$$

Map-Reduce Functionality

- Partitions data across machines
- Schedules execution across nodes
- Manages communication across machines
- Handles errors, machine failure

MapReduce: Model



Amazon Web Services

- Data centers owned and run by Amazon. You can rent "virtual computers" minute-by-minute basis
- more than 80% of the cloud computing market
- nearly 3,000 employees
- cost per machine: 0.01 to 4.00 /hour
- Several services in AWS
- S3 (Storage)
- EC2 (Individual Machines)
- Elastic Map Reduce
- distribute the data for Hadoop clusters

Distributed and Recursive Computing of Estimators

We want to compute the least squares estimator

$$\hat{\beta} \in \arg\min_b n^{-1} \sum_{i=1}^{n} (y_i - x_i' b)^2.$$

The sample size n is very large and can't load the data into a single machine. What could we do if we have a single machine or many machines?

> Use the classical sufficiency ideas to distribute jobs across machines, spatially or in time.

The OLS Example

- We know that
$$\hat{\beta} = (X'X)^{-1}(X'Y).$$
- Hence we can do everything we want with just:
$$X'X, \quad X'Y, \quad n, \quad S_0,$$
where S_0 is a "small" random sample $(Y_i, X_i)_{i \in I_0}$ with sample size n_0, where n_0 is large, but small enough that the data can be loaded in the machine.
- We need $X'X$ and $X'Y$ to compute the estimators to compute the estimator.
- We need S_0 to compute robust standard errors and we need to know n to scale these standard errors appropriately.

The OLS Example Continued

- The terms like $X'X$ and $X'Y$ are sums that can be computed by distribution of jobs over many machines:

1. Suppose machine j stores sample $S_j = (X_i, Y_i)_{i \in I_j}$ of size n_j.
2. Then we can map S_j to the sufficient statistics

$$T_j = \Big(\sum_{i \in I_j} X_i X_i', \sum_{i \in I_j} X_i Y_i, n_j \Big)$$

   for each j.
3. We then collect $(T_j)_{j=1}^M$ and reduce them further to

$$T = \sum_{j=1}^{M} T_j = (X'X, X'Y, n).$$

The LASSO Example

The Lasso estimator minimizes

$$(Y - X\beta)'(Y - X\beta) + \lambda\|\Psi\beta\|_1, \quad \Psi = \mathrm{diag}(X'X)$$

or equivalently

$$Y'Y - 2\beta'X'Y + \beta'X'X\beta + \lambda\|\Psi\beta\|_1.$$

Hence in order to compute Lasso and estimate noise level to tune λ we only need to know

$$Y'X, \quad X'X, \quad n, \quad S_0.$$

Computation of sums could be distributed across machines.

The Two Stage Least Squares

The estimator takes the form

$$(X'P_ZX)^{-1}X'P_ZY = (X'Z(Z'Z)^{-1}Z'X)^{-1}X'Z(Z'Z)^{-1}Z'Y.$$

Thus we only need to know

$$Z'Z, \quad X'Z, \quad Z'Y, \quad n, \quad S_0.$$

Computation of sums could be distributed across machines.

Exponential Families and Non-Linear Examples

Consider estimation using MLE based upon exponential families. Here assume data $W_i \sim f_\theta$, where

$$f_\theta(w) = \exp(T(w)'\theta + \varphi(\theta)).$$

Then the MLE maximizes

$$\sum_{i=1}^{n} log f_\theta(W_i) = \sum_{i=1}^{n} T(W_i)'\theta + \varphi(\theta) =: T'\theta + n\varphi(\theta).$$

The sufficient statistic T can be obtained via distributed computing. We also need an S_0 to obtain standard errors. Going beyond such quasi-linear examples could be difficult, but possible.

M- and GMM - Estimation

The ideas could be pushed forward using 1-step or approximate minimization principles. Here is a very crude form of one possible approach.

Suppose that $\hat{\theta}$ minimizes

$$\sum_{i=1}^{n} m(W_i, \theta).$$

Then given an initial estimator $\hat{\theta}_0$ computed on S_0 we could do Gradient descent iterations to approximate $\hat{\theta}$:

$$\hat{\theta}_{j+1} = \hat{\theta}_j - A_j \sum_{i=1}^{n} \nabla_\theta m(W_i, \hat{\theta}_j).$$

where A_j are matrices, e.g. $A_j = a_j I$.

We could do also do Newton iterations to approximate $\hat{\theta}$:

$$\hat{\theta}_{j+1} = \hat{\theta}_j - \left(\sum_{i=1}^{n} \nabla_\theta^2 m(W_i, \hat{\theta}_j) \right)^{-1} \sum_{i=1}^{n} \nabla_\theta m(W_i, \hat{\theta}_j).$$

M- and GMM - Estimation

Each Newton or gradient descent iteration involves sufficient statistics

$$\sum_{i=1}^{n} \nabla_{\theta}^{2} m(W_i, \hat{\theta}_j), \quad \sum_{i=1}^{n} \nabla_{\theta} m(W_i, \hat{\theta}_j)$$

which can obtained via distributed computing.

Conclusions

- We discussed the large p case, which is difficult. Approximate sparsity was used as a generalization of the usual parsimonius approach used in empirical work.
- A *sea of opportunities* for exciting empirical and theoretical work.
- We discussed the large n case, which is less difficult. Here the key is the distributed computing. Also big n samples often come in "unlabeled" form, so you need to be creative in order to make good use of them.
- This is an *ocean of opportunities*.