Topic 2: Scaling up

Eric B. Laber

Department of Statistical Science, Duke University

Statistics 561

On big data

Across all of its platforms, Google collects 25 petabytes of data every single day. That amount of data was unfathomable just a few years ago. To put this into perspective, imagine that each bit were a grain of rice. In just 8 hours, we would have enough rice to cover...you know what, if you're the kind of person that has better intuition for astronomically large piles of rice than digital data, what's the point of telling you anything? Do you have better intuition about how that rice should be stored? Manipulated? Used to predict new kinds of rice? Get the hell out of my office.
—Cassie Kozyrkov, Chief Decision Scientist at Google

On computing

One thing a computer can do that most humans can't is be sealed up in a cardboard box and sit in a warehouse.
—Avie Tevanian, CTO Apple Inc.

Warm-up (5 minutes)

- Explain to your group
  - What is stochastic gradient descent?
  - What is streaming data? What about a streaming algorithm?
  - What is an online prediction problem?
- True or false
  - The Johnsonville-Liverwurst theorem is a key result in dimension reduction
  - Googling "SIS" when searching for Sure Independent Screening yields a bunch of terrible YouTube videos that should never be viewed by anyone
  - Valentine's day evolved from the ancient Roman festival of Lupercalia in which lovers showed their affection through gifts and songs. The new holiday was created because Lupercalia was often confused with the festival of Lupinlavatio in which peasants, chosen by lottery, were forced to bath wolves.

Roadmap

- It's time to relax: running sums
- Random projections
- Stochastic approximation

Roadmap

- **It's time to relax: running sums**
- Random projections
- Stochastic approximation

Review: fitting a linear model

- Ordinary least squares estimator as

$$\widehat{\beta}_n = \arg\min_{\beta \in \mathbb{R}^p} \mathbb{P}_n (Y - \boldsymbol{X}^\intercal \beta)^2 = \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (Y_i - \boldsymbol{X}_i^\intercal \beta)^2$$

- Alternatively, view $\widehat{\beta}_n$ as solution to

$$\mathbb{P}_n \boldsymbol{X} (Y - \boldsymbol{X}^\intercal \beta) = 0$$

- If $\mathbb{P}_n \boldsymbol{X}\boldsymbol{X}^\intercal$ is invertible $\widehat{\beta}_n = (\mathbb{P}_n \boldsymbol{X}\boldsymbol{X}^\intercal)^{-1} \mathbb{P}_n \boldsymbol{X} Y$

Scaling up

- Collecting, storing, accessing, manipulating data is increasingly easy
  - More data, faster
  - Large p, lots of measurements per obs.
  - Large n, many obs.
- Need to balance statistical and computational efficiency

Large p (Part I)

- Methods so far designed for setting where p is smaller than n
- If $p \gg n$ we may want to screen out seemingly unimportant variables first, then apply the methods we've seen so far
  - In first step we want to make sure we include important ones, don't worry too much about accidentally including unimportant ones

Sure independence screening (SIS)

- Rank variables based on their marginal association with Y and choose subset that are ranked highest
  - Assume $\mathbb{X}$, Y are centered and scaled

$$W_j = \mathbb{P}_n X_j Y$$

select set of variables $\widehat{\mathcal{J}}_M = \{j : |W_j| \text{ is among the M largest}\}$

- `R` code example: screen.R

Sure independence screening (SIS) cont'd

- How do choose number of variables M?
  - Domain knowledge/expert judgment
  - Could treat M as tuning parameter opt using CV etc.
  - Error rate control
    - Many methods exist to avoid accidentally including noise variables (this is not exactly what we want)
    - Typically based on p-values (FDR control, etc.)

Big n (and possibly big p too)

- In some settings it may not be feasible to fit the model to all the data in one batch
  - n is very large, e.g., several GBs or PBs
  - Data are streaming in over time
- Trade-off statistical and computational/memory efficiency
- Key ideas
  - Divide, distribute, and conquer
  - Streaming, online updates

Divide, distribute, and conquer

- Simplest strategy: draw a sample of size $m \ll n$
  - Fit model, perform inference/validation etc.
  - Pros: representative sample, computationally feasible
  - Cons: fail to identify weak signals, lower quality model, may underestimate predictive accuracy
- Second simplest strategy: partition data into manageable chunks
  - Fit model on each partition (can do this in parallel)
  - Aggregate estimators
    - With parametric model, average parameter estimates
    - More generally, with a predictive model you can average predicted valued from each model

Divide, distribute, and conquer: example 1

- Sample mean $\mathbb{P}_n \boldsymbol{X}$
  - Create partition $C_1, \ldots, C_K$ of $\{1, \ldots, n\}$, compute

$$S_j = \sum_{i \in C_j} \boldsymbol{X}_i \quad \text{(serially or in parallel)}$$

then $\mathbb{P}_n \boldsymbol{X} = n^{-1} \sum_{j=1}^{K} S_j$

  - Note that $\{\boldsymbol{X}_i\}_{i \in C_j}$, $j = 1, \ldots, K$ do not need to be loaded into memory at the same time

Divide, distribute, and conquer: example 2

- Least squares $\widehat{\beta}_n = (\mathbb{P}_n \boldsymbol{X}\boldsymbol{X}^\intercal)^{-1} \mathbb{P}_n \boldsymbol{X} Y$
  - Create partition $C_1, \ldots, C_K$ of $\{1, \ldots, n\}$, compute

$$\begin{aligned} \Sigma_j &= \sum_{i \in C_j} \boldsymbol{X}_i \boldsymbol{X}_i^\intercal \\ \Gamma_j &= \sum_{i \in C_j} \boldsymbol{X}_i Y_i \end{aligned}$$

then $\widehat{\beta}_n = \left\{ \sum_{j=1}^{K} \Sigma_j \right\}^{-1} \sum_{j=1}^{K} \Gamma_j$

Divide, distribute, and conquer: example 3

- Lasso estimator $\widehat{\beta}_n^\tau = \arg\min_\beta \mathbb{P}_n(Y - \boldsymbol{X}^\intercal\beta)^2 + \tau||\beta||_1$
  - Fact: $\widehat{\beta}_n^\tau$ is completely determined by $\mathbb{P}_n\boldsymbol{X}\boldsymbol{X}^\intercal$ and $\mathbb{P}_n\boldsymbol{X}Y$
  - We can use distd least squares algorithm for lasso

PF:

$$\begin{aligned}\mathbb{P}_n(Y - \boldsymbol{X}^\intercal\beta)^2 + \tau||\beta||_1 &= \mathbb{P}_n(Y - \boldsymbol{X}^\intercal\widehat{\beta}_n)^2 + \mathbb{P}_n(\boldsymbol{X}^\intercal\widehat{\beta}_n - \boldsymbol{X}^\intercal\beta)^2 + \tau||\beta||_1 \\ &= \mathit{Const} + (\widehat{\beta}_n - \beta)^\intercal\mathbb{P}_n\boldsymbol{X}\boldsymbol{X}^\intercal(\widehat{\beta}_n - \beta) + \tau||\beta||_1\end{aligned}$$

Divide, distribute, and conquer: example 3 cont'd

- Write $\widehat{\beta}_n^{\tau} = \arg\min_{\beta} (\widehat{\beta}_n - \beta)^{\mathsf{T}} \mathbb{P}_n \boldsymbol{X}\boldsymbol{X}^{\mathsf{T}} (\widehat{\beta}_n - \beta) + \tau ||\beta||_1$
  - Create partition $C_1, \ldots, C_K$ of $\{1, \ldots, n\}$, compute $\Sigma_j = \sum_{i \in C_j} \boldsymbol{X}_i \boldsymbol{X}_i^{\mathsf{T}}$, $\Gamma_j = \sum_{i \in C_j} \boldsymbol{X}_i Y_i$, and $\widehat{\beta}_n$ using distd least squares algorithm
  - Compute

$$\widehat{\beta}_n^{\tau} = \arg\min_{\beta} n^{-1} (\widehat{\beta}_n - \beta)^{\mathsf{T}} \left(\sum_{j=1}^{K} \Sigma_j \right) (\widehat{\beta}_n - \beta) + \tau ||\beta||_1$$

  - Can be solved as quadratic program or using subgrad descent

Divide, distribute, and conquer: code

- `R` code example: distributed.R
- In class exercise: with your neighbor design and implement distributed ridge regression
- Group discussion: how to implement distributed screening?

Divide, distribute, and conquer: KM

- Suppose we observed right censored data $\{T_i, \delta_i\}_{i=1}^n$
  - T is (discrete) censoring or failure time
  - δ denotes censoring indicator ($\delta = 1$ if failure)
  - T has density $f(t)$ and survival fn $S(t)$
- Kaplan-Meier estimator

$$\widehat{S}(t) = \prod_{\ell \,:\, t_\ell < t} \left(1 - \frac{d_\ell}{n_\ell}\right),$$

where d_ℓ is the number of failures and n_ℓ is the number at risk at time t_ℓ

Distributed KM

- Distributed computation of KM
  - Create partition $C_1, \ldots, C_K$ of $\{1, \ldots, n\}$, compute

$$
\begin{aligned}
n_{\ell,j} &= \sum_{i \in C_j} 1_{t_i > t_\ell} \\
d_{\ell,j} &= \sum_{i \in C_j} 1_{t_i = t_\ell} \delta_i,
\end{aligned}
$$

then $n_\ell = \sum_{j=1}^{K} n_{\ell,j}$ and $d_\ell = \sum_{j=1}^{K} d_{\ell,j}$

Exact distributed computing

- So far, we've developed distributed algorithms that give us *exactly* the same solution as if we had used all th data at once
  - Linear estimators (or fns of linear estimators)
  - Counts
- Unfortunately, not all estimators lend themselves to nice analytic decompositions that are easily distributed
- However, many statistical estimators minimize (maximize) an objective that sums over data
  - Exploit this structure to construct general distd algorithms
  - Our primary tool will be approximate gradient descent

Quick review (sub)grad. descent

- Goal: minimize the function $f(\theta)$ over $\theta \in \mathbb{R}^p$
  - $f(\theta)$ decreases most rapidly in the direction of $\nabla f(\theta)$
  - Gradient descent update: $\theta^{(k+1)} = \theta^{(k)} - \alpha^{(k)} \nabla f(\theta^{(k)})$
- Choosing the step-size $\alpha^{(k)}$ is non-trivial
  - Line-search $\alpha^{(k)} = \arg\min_{\alpha} f\left\{\theta^{(k)} - \alpha \nabla f(\theta^{(k)}\right\}$
  - Barzilai-Borwein

$$\alpha^{(k)} = \frac{(\theta^{(k)} - \theta^{(k-1)})^\intercal \left\{\nabla f(\theta^{(k)}) - \nabla f(\theta^{(k-1)})\right\}}{||\nabla f(\theta^{(k)}) - \nabla f(\theta^{(k-1)})||^2}$$

- If $f(\theta)$ convex but not differentiable, replace gradient with a subgradient (this is what we did with distd lasso)

M-estimators

- Many statistical estimators have the form

$$\widehat{\theta}_n = \arg\min_{\theta} g\left\{\mathbb{P}_n f(\mathbf{Z}_i; \theta)\right\}$$

- Canonical examples
  - $g(u) = u$, $\boldsymbol{Z} = (\boldsymbol{X}, Y)$ and $f(\mathbf{Z}; \theta) = (Y - \boldsymbol{X}^{\intercal}\theta)^2$
  - $g(u) = u$, $f(\boldsymbol{Z}; \theta)$ is negative log-density (max LH)
  - $g(u) = ||u||^2$, and $\mathbb{P}_n f(Z; \theta) = 0$ is est. eqn.

Distributed approximate gradient descent

- Gradient using all the data

$$\frac{\partial}{\partial \theta^{\mathsf{T}}} g\left\{\mathbb{P}_n f(\boldsymbol{Z}_i;\theta)\right\} = g'\left\{\mathbb{P}_n f(\boldsymbol{Z}_i;\theta)\right\} \mathbb{P}_n \nabla f(\boldsymbol{Z}_i;\theta)$$

- Distributed approximate gradient descent
  1. Create partition $C_1, \ldots, C_K$ of $\{1, \ldots, n\}$
  2. Set $\theta^{(1)}$ to starting value, set $k = 1$
  3. Randomly select partition j from $\{1, \ldots, K\}$, compute

$$\delta^{(k)} = g'\left\{ n_j^{-1} \sum_{i \in C_j} f(\boldsymbol{Z}_i;\theta^{(k)}) \right\} n_j^{-1} \sum_{i \in C_j} \nabla f(\boldsymbol{Z}_i;\theta^{(k)})$$

  - Update $\theta^{(k+1)} = \theta^{(k)} - \alpha^{(k)} \delta^{(k)}$

Distributed Cox-PH

- Full data are $\{(\boldsymbol{X}_i, T_i, \delta_i)\}_{i=1}^n$ where $\boldsymbol{X}$ are covariates, T is obs. time, and δ is censoring indicator
  - Recall that the Cox-PH model postulates

$$\lambda(t|\boldsymbol{x}) = \lambda_0(t) \exp\{\boldsymbol{x}^\intercal \theta\}$$

- Define $Y_\ell(t) = 1_{T_\ell \geq t}$, Cox-PH estimator

$$\widehat{\theta}_n = \arg\max_\theta \sum_{i=1}^n \delta_i \left[\boldsymbol{X}_i^\intercal \theta - \log \left\{ \sum_{\ell=1}^n Y_\ell(T_i) \exp\left(\boldsymbol{X}_\ell^\intercal \theta\right) \right\} \right],$$

Recall that the above is the log partial likelihood

Distributed Cox-PH cont'd

- Evaluate partial likelihood on subset C_j using

$$\sum_{i \in C_j} \delta_i \left[\boldsymbol{X}_i^\top \theta - \log \left\{ \sum_{\ell \in C_j} Y_\ell(T_i) \exp\left(\boldsymbol{X}_\ell^\top \theta\right) \right\} \right],$$

- Gradient of negative partial log-likelihood on subset C_j is

$$U_j(\theta) = -\sum_{i \in C_j} \delta_i \left\{ \boldsymbol{X}_i - \overline{\boldsymbol{X}}_{C_j}(T_i, \theta) \right\},$$

where
$$\overline{\boldsymbol{X}}_{C_j}(t, \theta) = \sum_{\ell \in C_j} \boldsymbol{X}_\ell Y_\ell(t) \exp\left(\boldsymbol{X}_\ell^\top \theta\right) / \sum_{\ell \in C_j} Y_\ell(t) \exp\left(\boldsymbol{X}_\ell^\top \beta\right)$$

Distributed Cox-PH cont'd

- Distributed approximate gradient descent Cox-PH
  1. Create partition $C_1, \ldots, C_K$ of $\{1, \ldots, n\}$
  2. Set $\theta^{(1)}$ to starting value, set $k = 1$
  3. Randomly select partition j from $\{1, \ldots, K\}$
  - Update $\theta^{(k+1)} = \theta^{(k)} - \alpha^{(k)} U_j^{(k)}(\theta^{(k)})$
- `R` code break: coxph.R

Distributed computing discussion

- Reviewed basic distd optimization
  - Exact distd programming
  - Approximate gradient descent
- These methods can be applied to data streaming in real-time (the partition elements $C_1, \ldots, C_k$ can be arriving as data accumulates)
- Alternative strategy is to fit separate models and aggregate/average estimators

Break: algorithm quiz

- Recall that the median satisfies $\hat{m}_n = \arg\min_m \mathbb{P}_n |Y - m|$
  - With your neighbor, design an algorithm that estimates the median using only one data point at a time
  - Implement in `R`, how does your algorithm compare with median fn using all the data in one batch?

Basis expansions

- Linear model viewed as a first-order approximation
  - I.e., $f(\boldsymbol{x}) = f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)^\intercal(\boldsymbol{x} - \boldsymbol{x}_0) + \text{HOT}$
  - Natural approach add flexibility is to include HOT
- Monomial regression
  - No interactions: $f(\boldsymbol{x}) \approx \beta_0 + \sum_{j=1}^{p}\sum_{k=1}^{r}\beta_{j,k}x_i^k$
  - Pairwise interactions:
    $f(\boldsymbol{x}) \approx \beta_0 + \sum_{j=1}^{p}\sum_{k=1}^{r}\beta_{j,k}x_i^k + \sum_{j=1}^{p}\sum_{\ell=1}^{p}\sum_{k=1}^{r-1}\beta_{j,\ell,k}x_j^k x_\ell^{r-k}$
  - Intuitive but unstable as degree of polynomial increases

Basis expansions cont'd

- Consider approximations of the form

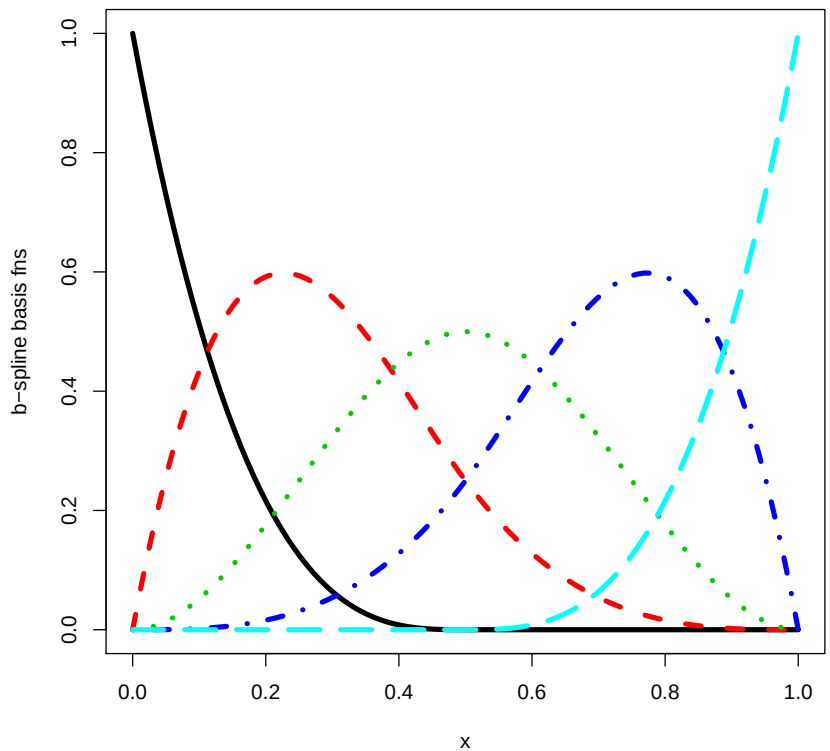
$$f(\boldsymbol{X}) \approx \beta_0 + \sum_{j=1}^{q} \beta_j b_j(\boldsymbol{X}),$$

  where $b_j : \mathbb{R}^p \to \mathbb{R}$ are basis functions

  - E.g., $b_j(\boldsymbol{X}) = X_{k(j)}^{d(j)}$ in polynomial w/o interactions
  - Typically choose basis $\{b_j(\cdot)\}_{j \geq 1}$ to be dense in some space of interest, e.g., the space of continuous functions $[0, 1]$ etc.

B-splines

- Polynomial segments between knots
  $0 = a_0 < a_1 < \cdots < a_K = 1$
- Constrained to be smooth at knots
- Specification of a B-spline basis requires order of polynomial segments and knots



B-splines cont'd

- Univariate `R` code example: nonlinearReg.R

Other basis systems

- Fourier basis $\{(\sin(k\omega x), \cos(k\omega x)\}_{k \geq 1}$
  - Periodic may not be good for growth curves etc.
- Wavelet basis (many variants)
  - Widely used for "denoising" in signal processing
  - A number of deep theoretical results exist for nonparametric regression with wavelet bases

Multiple predictors

- Given basis system $\{b_j(\cdot)\}_{j\geq 1}$ we create 'features' $\{b_j(X_k)\}_{j\geq 1}$, $k = 1, \ldots, p$
  - We can construct a model with interactions among basis functions etc.
  - Number of terms in the model can grow quickly, e.g., with J basis fns and pairwise interactions $O(p^2 J^2)$ terms
- Use all of the model building tools we already talked about to deal with large p

Thank you.

eric.laber@duke.edu

laber-labs.com