# Scalable Data Analysis

David M. Blei

April 26, 2012

## 1 Why?

- Olden days:
  - Small data sets
  - Careful experimental design
  - Challenge: Squeeze as much as we can out of the data
- Modern data analysis:
  - Very large data sets ("Big Data")
  - Lots of observational data
  - Challenge: How can we compute with large data sets?
- Today we discuss two strategies
  - Map-Reduce: Data analysis on clusters of machines
  - Stochastic optimization: Subsampling the data as part of the procedure

## 2 Map-Reduce

- Main idea: Computation that flows in the following way.
  1. Many *local* computations, made in parallel
  2. These computations combine to form result
- More frequently, this is iterated. Suppose there are $K$ nodes.

1. We have a state $\theta^{(t)}$, communicated to each node
2. Make local computations in parallel $z_i = f_i(\mathscr{D}, \theta^{(t)})$
3. Communicate results $f_i(\mathscr{D})$
4. Form new state $\theta^{(t+1)} = g(z_1, \ldots, z_K)$
5. Repeat

- Often the local computations involve subsets of the data.

- One stage example: The MLE of exponential families (e.g., Gaussian) requires a sum of sufficient statistics.

- By expressing a sum as a sum of sums we can use map-reduce

  1. Send subsets of the data to each node
  2. Form local sums at each node
  3. Combine the local sums to form the complete sum
  4. Find the MLE using the sum of sufficient statistics

- Iterative example: complex GLMs

  – GLMs (with canonical link) require gradient-based optimization, where the gradient depends on a sum over the data.
  – This is expensive each time we compute the gradient.
  – E.g., recall the logistic regression derivative

$$\frac{d\mathscr{L}}{d\beta_i} = \sum_{n=1}^{N} (y_n - \mu(\beta^\top x_n)) x_{ni} \tag{1}$$

  – (Note: linear regression is more like the MLE case; we only need to compute the sum of $x_n y_n$ once to get the exact solution.)

- Map reduce solution:

  1. Distribute the data across many nodes.
  2. At iteration $t$, the coefficients are $\beta^{(t)}$.
  3. For each subset, form predictions $\hat{y}_n = \mu(\beta^\top x_n)$
  4. Compute the sum $\sum(y_n - \hat{y}_n)x_n$ for each subset.
  5. Combine the sums to form the full gradient.
  6. Follow the gradient to obtain $\beta^{(t+1)}$.

7. Communicate $\beta^{(t+1)}$ to the nodes and repeat.

(It's very likely that this is how your spam filter was fit.)

- Another iterative example: EM

1. Distribute the data across many nodes.
2. At iteration $t$, the model parameters are $\theta^{(t)}$.
3. E-step
   - For each subset, estimate the posterior for each data point $p(z_n \mid x_n, \theta^{(t)}$.
   - E.g., compute the posterior cluster assignment
   - Compute the sum of expected sufficient statistics for the M-step.
   - E.g., compute the sums of data subsets, weighted by their class assignments.
4. M-step
   - Combine the sums of sufficient statistics.
   - Compute the expected MLE of the model parameters $\theta^{(t+1)}$.
   - Communicate the model parameters to the nodes

- Note: this is a good way to fit NMF, PCA, FA etc.

# 3 Stochastic optimization

- In several topis this semester, we have somewhat taken for granted that we can maximize a function using its gradient.

- Let $f(\theta)$ be an *objective function* and $f'(\theta)$ be its derivative or gradient.

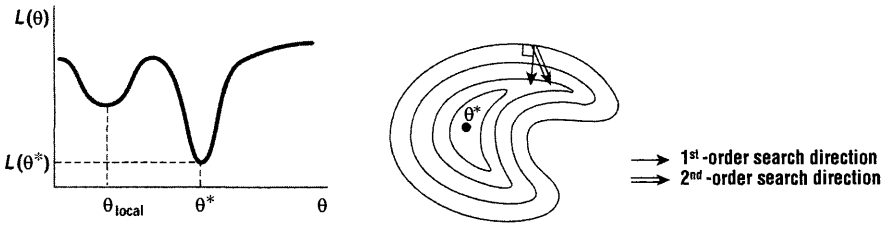
- The idea behind *gradient-based optimization* is to search for the optimal value of $\theta$ by following the direction of steepest ascent

$$\theta^{(t+1)} = \theta^{(t)} + a_{t+1} f'(\theta^{(t)}) \tag{2}$$

  Where $a_{t+1}$ is a step-size. With conditions, this procedure provably hits the optimum (or a local optimum).

- Note the (local) optimum is where $f'(\theta) = 0$.

- Examples:




Note the local optimum.

- Consider the generalized linear model (e.g. logistic regression). We stopped discussing fitting after writing down the gradient

$$\nabla_\beta \mathscr{L} = \sum_{n=1}^{N} (y_n - \mathrm{E}[Y \mid x_n, \beta]) x_n \tag{3}$$

- Gradient ascent is used everywhere e.g., fitting statistical models, neural networks, matrix factorization, etc. There are innovations which I won't describe
  - Second order updates
  - Line search

- Contrast to *coordinate ascent* optimization (e.g., EM and k-means).

- *Optimization* is a vital field to machine learning and statistics. In the past, statistics bundled the model and objective to the fitting procedure. Now, we separate these ideas.

- The complexity of gradient-based optimization depends on computing the gradient and objective function.

- What happens when we have massive data? Gradients like the GLM gradient are expensive to compute. (And note this is the same issue as the one that motivated map-reduce.)

- *Stochastic optimization* was invented in 1951 by Robbins and Monro.
  - We follow *noisy estimates* of the gradient with a decreasing step-size.
  - Subject to certain conditions, we will hit the optimum (or local optimum).

- Intuition: A drunk cross-country trip
  - You are traveling cross country

- Everyone is drunk (except you)
- Ask a drunkard for a direction and go that direction 1500 miles
- Repeat, each time moving fewer miles

- In more detail:
  - Let $G(\theta)$ be a noisy estimate of the gradient at $\theta$. It is a *random variable*.
  - Note that its expectation is the gradient, $\mathrm{E}[G(\theta)] = g(\theta)$.
  - Let $g_t(\theta)$ be a relalization from the random variable.
  - Robbins-Monro says

$$\theta^{(t+1)} = \theta^{(t)} + a_t g_t(\theta^{(t)}) \tag{4}$$

- This converges if the sum of squares of the step sizes converges and the sum of the step sizes diverges,

$$\sum_{t=1}^{\infty} a_t^2 < \infty \tag{5}$$

$$\sum_{t=1}^{\infty} a_t = \infty. \tag{6}$$

- For example,

$$a_t = (t)^{-\kappa} \tag{7}$$

  where $\kappa \in (0.5, 1)$

- This is neat, but why is it more scalable. *Key idea: repeated subsampling of the data.*

- Many gradients—like the one for the GLM—are expressed in terms of a sum over data points.

- We get a noisy gradient with the right expectation by sampling a data point and comuting a scaled version of its gradient
  - Let $I$ be a uniformly distributed index from 1...$N$.
  - Sample an index and consider this random gradient

$$\nabla_\beta \mathscr{L}_I = N(y_I - \mathrm{E}[Y \mid x_I, \beta])x_I \tag{8}$$

  - Note that this is easy to comute—we only need to process one data point.
  - Also note that the expectation is the true gradient.

$$\mathrm{E}[\mathscr{L}_I(\theta)] = \sum_{i=1}^{N} (y_i - \mathrm{E}[Y \mid x_i, \beta])x_i \tag{9}$$

- The stochastic optimization algorithm is to repeatedly
  1. Sample a data point $i$
  2. Set $a_i = i^{-\kappa}$
  3. Move in the direction of $\mathscr{L}_i$ with step size $a_i$

- Used in
  - Model fitting
  - Neural network training
  - Signal processing and experimental design
  - Matrix factorization (e.g., the in-painting example)

- Many innovations
  - Changing objective function
  - Streaming data, treated as sampled from a population
  - Minibatches and combined with map-reduce

- In practice, it often works even better than full gradient optimization. The stochastic nature of the algorithm lets it jump out of poor local optima.

- Originally motivated for places where we can only obtain random measurements. Then randomness was injected to improve convergence. Here, randomness is injected to improve computation too.