Inference on randomized algorithms

Miles Lopes

UC Davis, Department of Statistics

melopes@ucdavis.edu

BIRS Workshop

Randomized algorithms arise frequently in large-scale data analysis.

Examples.

- Random forests and bagging
- Stochastic gradient methods
- Random projection for dimension reduction
- Sketching methods for data streams and numerical linear algebra
- ...

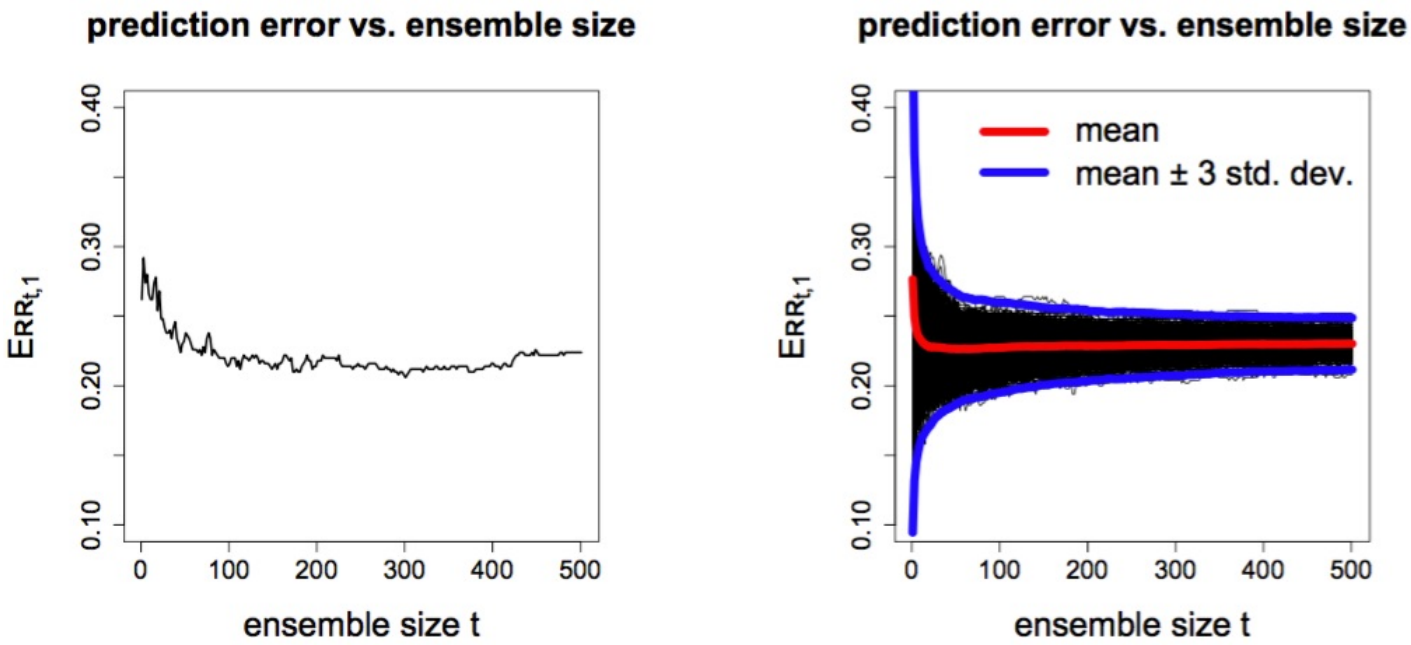
In order to know if these algorithms are working in practice, we need to estimate their (random) errors, relative to an "ideal solution".

This is often an *inferential problem*.

Example 1: Random forests

A large collection of randomized decision trees are trained, and then predictions are made by aggregation.

Problem: How does prediction error fluctuate as a function of ensemble size?



cf. Wager, Hastie, & Efron 2014, Lopes 2016, Mentch and Hooker 2016, Scornet 2016

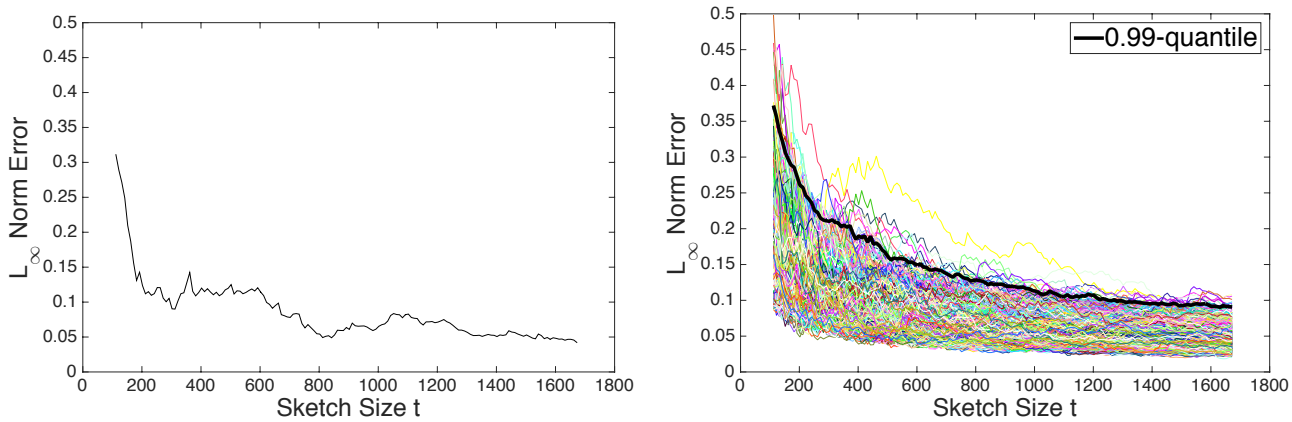
Example 2: matrix sketching

Suppose we want to compute $\mathbf{A}^\top \mathbf{B}$, where $\mathbf{A}$ and $\mathbf{B}$ are extremely tall matrices.

Sketching involves sampling rows to obtain short "sketches" $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}$.

Then, we use $\tilde{\mathbf{A}}^\top \tilde{\mathbf{B}}$ as a fast approximation to $\mathbf{A}^\top \mathbf{B}$.

Problem: How does sketching error depend on the number of sampled rows?



cf. Halko, Martinsson & Tropp 2011, Lopes, Wang & Mahoney 2017

General remarks

- For many randomized algorithms, there are few methods available for diagnosing convergence in a precise way.
- In order to balance statistical error and algorithmic error, we need uncertainty quantification for both.
- Although theoretical convergence results sometimes exist, they are often only qualitative and are not adaptive to a particular dataset.
- Many classical tools from statistical inference are relevant. However, inference on randomized algorithms is unconventional insofar as it is conditional on the data.
- Computational constraints play an important role – because we do not want the cost of checking convergence to dominate the cost of the original algorithm.