Distributed Monte Carlo testing

Patrick Rubin-Delanchy

September 6, 2014

Big data often takes the form of network data, for example, traffic generated on a computer network, messages and other connections made on social networks, mobile communications, the web, collaboration networks (e.g. in academia, music or film) and more.

A characteristic of such data, essentially because they relate to human behaviour, is that they are very highly *structured*. Simple models, for instance, the Erdos-Renyi graph or the homogeneous Poisson process, are inadequate because they fail to capture temporal dynamics, the structure of connectivity patterns, and all of the additional information present in these data. For example, a computer network is largely (but not completely) bi-partite: most computers spend most of their time in one role, either client or server — ignoring this is perilous.

At the same time, more involved modelling approaches have their own pitfalls. If we could restrict attention to a particular edge or node, then perhaps it would be conceivable to fit a sophisticated model to the data. However it quickly becomes apparent that to some degree every node and interaction is different. On a computer network, a node could be a printer, a domain name server, a laptop, and so on; it is hard to conceive of a modelling approach that would not have to address each case separately. This soon becomes a prohibitively painstaking endeavour.

In this talk we will argue that Monte Carlo testing offers a very promising approach to statistical inference for Big data. Monte Carlo tests allow us to answer simple questions, by a statistically sound paradigm, without extensive modelling. They can be used *en masse* for feature discovery, in combination to allow joint inference on heterogeneous data sources, and they can even be used in layers in a hierarchical model.

The fundamental "trick" that most Monte Carlo tests use is to condition on some aspect of the data, say X. If the error is controlled at a level α (often 5%) given $X = x$, for all x, then it is also controlled unconditionally. The great advantage of this approach is that we assume nothing about X. The permutation test is probably the most common example of such an approach. Suppose we are given two groups, A and B, and the task is to determine whether B's average is significantly larger than A's. By repeatedly randomly permuting elements between A and B we can estimate the probability than the simulated difference between the averages is as large as the difference observed. Asymptotically this yields an exact p-value that makes no distributional assumptions on A and B apart from exchangeability.

The main challenge of implementing Monte Carlo tests on Big data is to control the number of samples taken per test. This is because there could be millions of tests, thousands of samples per test, and every sample could require churning through Gigabytes of data. In this talk, we will present two principles by which the effort can be controlled.

The first principle relies on the idea that, in a multiple testing scenario, p-values that are not tiny are "not interesting". More precisely, we propose a mechanism for stopping computation when a confidence interval satisfying some user-supplied criteria has been met. The stopping rule generates a confidence interval with the correct coverage probability, that is, it does not ignore bias due to dependent stopping. The results are quite remarkable: for example, for a not-too-stringent confidence interval criterion we show that the algorithm can stop after just two simulations.

The second principle is applicable when we want to combine evidence from different tests. Here we view the tests as competing for computer resources and the task is to allocate wisely. We will show quite considerable gains in performance over the naïve, equal allocation strategy. In fact, the asymptotic variance of the combined p-value can be divided by almost n, where n is the number of tests.

The methodology we present will be illustrated in a few examples: detecting a change after phishing attack on a computer network, predicting collaborations in a music network, and modelling information flow on a communication network.

This is joint work with Axel Gandy (Imperial College London), with contributions from Nicholas Heard (Imperial College London), Niall Adams (Imperial College London), Daniel Lawson (University of Bristol) and Theo Dickson (University of Bristol).