Monte Carlo testing with Big Data

Patrick Rubin-Delanchy

University of Bristol & Heilbronn Institute for Mathematical Research

Joint work with:
Axel Gandy (Imperial College London)

with contributions from:
Nick Heard (Imperial College London), Dan Lawson (University of Bristol),
Niall Adams (Imperial College London), Melissa Turcotte (Los Alamos National Laboratory)

8th January 2015

Michael Jordan: "When you have large amounts of data, your appetite for hypotheses tends to get even larger." (IEEE Spectrum, 20th October 2014).

His point is about the risk of spurious discoveries when the number of possible hypotheses grows exponentially.

Whilst our focus is mostly on the algorithmic challenge of implementing many tests, the theory of multiple testing is at the heart of our approach.

The hypothesis testing framework

- The null hypothesis, H_0, gives a probabilistic description of the data if there is "no effect". The alternative, H_1, describes the case where the effect is present.
- We have a test statistic t which would tend be small under H_0, large under H_1.
- The p-value of the test is

$$p = \mathrm{P}(T \geq t),$$

  where T is a replicate of t under H_0.
- It is straightforward to prove that p is uniformly distributed on $[0, 1]$ under H_0 (if T is absolutely continuous). A low p-value, e.g. $1/100$, is considered evidence in favour of the alternative.

Monte Carlo test

Often $\mathrm{P}(T \geq t)$ cannot be calculated exactly.

A typical solution: simulate replicates of T under the null hypothesis, $T_1, \ldots, T_N$, and estimate

$$\hat{p} = \frac{1}{N} \sum_{j=1}^{N} \mathbb{I}(T_j \geq t),$$

where $\mathbb{I}$ is the indicator function, N large.

Permutation test

- Let A and B denote two groups of data.
- Under H_0, the data are exchangeable — the A/B labelling is arbitrary.
- Let t be some measure of discrepancy between A and B (e.g. the difference in means).
- Simulate T_j by randomly relabelling the data, and recomputing the discrepancy.
- Estimate

$$\hat{p} = \frac{1}{N} \sum_{j=1}^{N} \mathbb{I}(T_j \geq t).$$

The permutation test essentially works by combining two important ideas: exchangeability and conditioning.

1. If the false positive rate is controlled at a level α (often 5%) given $X = x$, for all x, then it is also controlled marginally.
2. The simulated T_j, in the permutation test, come from conditioning on the observed values, but not their labelling as A or B.

More generally, my view is that exchangeability and other sorts of stochastic orders are key to robust inference on Big Data.

Running a large number of Monte Carlo tests

Some possible principles:

1. Make each individual test faster, see e.g. Gandy (2009).
2. View task as a resource allocation problem: define an error and try to minimise its mean square.

Choosing the error without taking into account the special nature the problem, e.g. using the Euclidean distance from the true vector of p-values, doesn't make much sense. For example, we could easily end up investing most of the effort on p-values around 0.5, which are typically not of interest.

I will argue that for some Big Data problems it is natural to define the error on the basis of a method to combine p-values.

Running a large number of Monte Carlo tests

Some possible principles:

1. Make each individual test faster, see e.g. Gandy (2009).
2. View task as a resource allocation problem: define an error and try to minimise its mean square.

Choosing the error without taking into account the special nature the problem, e.g. using the Euclidean distance from the true vector of p-values, doesn't make much sense. For example, we could easily end up investing most of the effort on p-values around 0.5, which are typically not of interest.

I will argue that for some Big Data problems it is natural to define the error on the basis of a method to combine p-values.

Running a large number of Monte Carlo tests

Some possible principles:

1. Make each individual test faster, see e.g. Gandy (2009).
2. View task as a resource allocation problem: define an error and try to minimise its mean square.

Choosing the error without taking into account the special nature the problem, e.g. using the Euclidean distance from the true vector of p-values, doesn't make much sense. For example, we could easily end up investing most of the effort on p-values around 0.5, which are typically not of interest.

I will argue that for some Big Data problems it is natural to define the error on the basis of a method to combine p-values.

Combining p-values

Consider the global hypothesis test:

$\tilde{H}_0$: *all* hypotheses hold, the p-values are independent and uniformly distributed,

versus,

$\tilde{H}_1$: not all null hypotheses hold.

Combining p-values is testing the p-values, rather than the original data. Note that we are using the p-values together to make one, global, decision. We are ignoring the local errors of accepting or rejecting each individual hypothesis. This can be reasonable if:

1. The data are heterogeneous — there isn't an obvious way to combine them into an overall test statistic.
2. There is too much data — a divide-and-conquer strategy is necessary.

Some well-known test statistics and their distributions under $\tilde{H}_0$ (when known):

1. Fisher's method: $-2\sum \log(p_i) \sim \chi^2_{2m}$ (Mosteller and Fisher, 1948).
2. Stouffer's score: $\sum \Phi^{-1}(1-p_i)/\sqrt{m} \sim \text{normal}(0,1)$ (Stouffer *et al.*, 1949).
3. Simes's test: $\min\{p_{(i)}\, m/i\} \sim \text{uniform}[0,1]$ (Simes, 1989).
4. Donoho's statistic (Donoho and Jin, 2004):

$$\max_{1\le i\le \alpha_0 m} \sqrt{m}\{i/m - p_{(i)}\} \Big/ \sqrt{p_{(i)}\{1-p_{(i)}\}}\,.$$

Probabilistic framework

1. Let $p = (p_1, \ldots, p_m)^T$ be the vector of true (and unobservable) p-values.
2. We can construct any number of independent Bernoulli variables with success probability p_i, for $i = 1, \ldots, m$.
3. Let f be the function used to combine p-values, and let $\text{var}_N\{f(\hat{p})\} = N^{1/2}\{f(\hat{p}) - f(p)\}$, where N is the total number of samples.
4. We want to adaptively allocate the number of samples per test in order to minimise $\text{var}_N\{f(\hat{p})\}$.

Potential gains over even allocation

For tests that focus attention on one p-value, e.g. $\min\{p_{(i)}\ m/i\}$ (Simes), we can reduce the asymptotic variance by a factor of m (the number of tests).

This is also true for some smooth functions of the p-values, including Fisher and Stouffer:

Lemma

If there exists $i \in 1, \ldots, m$ *such that*

$$\min_{k \neq i} \sup_{x \in [0,1]^m} |\nabla_i f(x)| / |\nabla_k f(x)| = \infty,$$

then

$$\sup_{p_i \in (0,1)^m} \frac{\mathrm{var}_\infty\{f(\hat{p}_{naïve})\}}{\mathrm{var}_\infty\{f(\hat{p}_{opt})\}} = m.$$

Outline of algorithm

For known p and f we can calculate the optimal asymptotic allocation. E.g. with Fisher's method p_i should get $\propto \{p_i/(1-p_i)\}^{1/2}$ of samples.

In batches of Δ:

1. Decide how to allocate Δ samples amongst m streams using current $\hat{p}$.
2. Sample test replicates.
3. Update $\hat{p}$.

Various optimisations are possible (and necessary), e.g. how to estimate p, how to estimate allocation from p, whether we try to 'correct' for previously wrongly allocated effort.

The main point is that our algorithm seems to achieve the optimal asymptotic variance.

Outline of algorithm

For known p and f we can calculate the optimal asymptotic allocation. E.g. with Fisher's method p_i should get $\propto \{p_i/(1-p_i)\}^{1/2}$ of samples.

In batches of Δ:

1. Decide how to allocate Δ samples amongst m streams using current $\hat{p}$.
2. Sample test replicates.
3. Update $\hat{p}$.

Various optimisations are possible (and necessary), e.g. how to estimate p, how to estimate allocation from p, whether we try to 'correct' for previously wrongly allocated effort.

The main point is that our algorithm seems to achieve the optimal asymptotic variance.

Outline of algorithm

For known p and f we can calculate the optimal asymptotic allocation. E.g. with Fisher's method p_i should get $\propto \{p_i/(1-p_i)\}^{1/2}$ of samples.

In batches of Δ:

1. Decide how to allocate Δ samples amongst m streams using current $\hat{p}$.
2. Sample test replicates.
3. Update $\hat{p}$.

Various optimisations are possible (and necessary), e.g. how to estimate p, how to estimate allocation from p, whether we try to 'correct' for previously wrongly allocated effort.

The main point is that our algorithm seems to achieve the optimal asymptotic variance.

Outline of algorithm

For known p and f we can calculate the optimal asymptotic allocation. E.g. with Fisher's method p_i should get $\propto \{p_i/(1-p_i)\}^{1/2}$ of samples.

In batches of Δ:

1. Decide how to allocate Δ samples amongst m streams using current $\hat{p}$.
2. Sample test replicates.
3. Update $\hat{p}$.

Various optimisations are possible (and necessary), e.g. how to estimate p, how to estimate allocation from p, whether we try to 'correct' for previously wrongly allocated effort.

The main point is that our algorithm seems to achieve the optimal asymptotic variance.

Cyber-security example: change detection in Netflow

- Traffic from one computer on Imperial College's network (with thanks to Andy Thomas) over a day.
- The data has an artificial changepoint where the user knowingly changed behaviour.
- We split the computer's traffic by edge (the other IP address), bin the data per hour, and throw away any edge with less than three bins.
- This results in approximately 100 time series.

Cyber-security example: change detection in Netflow cont'd

1. For any proposed changepoint, count the absolute difference between the number of flows before and after the changepoint on each edge, resulting in a statistic t_i.
2. For each edge:
   1. Randomly permute the binned data.
   2. Re-compute the absolute difference between the number of flows, resulting in a simulated statistic T_{ij}.
   3. Get a running estimate of the changepoint p-value for that edge, $\hat{p}_i$.
3. Use our algorithm to combine the p-values using Fisher's score.

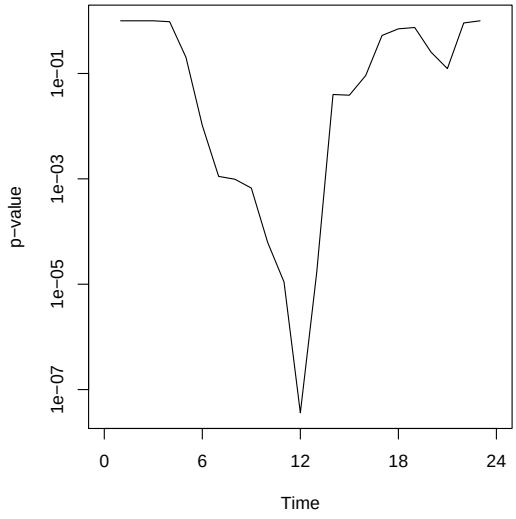

Figure : Overall p-value in favour of a changepoint

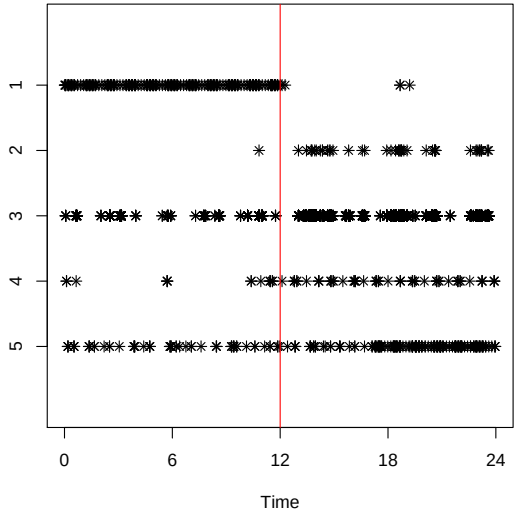

Figure : Most significant edges. Samples taken: 15039, 14767, 11598, 7985, 6931

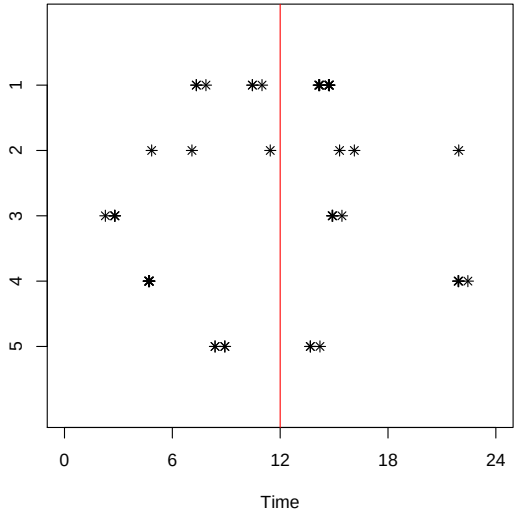

Figure : Least significant edges. Samples taken: 55, 56, 58, 50, 61

Other applications in Big Data

- Detecting changes in user authentification graphs (joint work with Melissa Turcotte and Los Alamos National Laboratory).
- Hypothesis tests on networks (e.g. finding link predictors, scoring a graph motif).
- Information flow (e.g. correlations in timing, matching words or phrases).
- Generic model-checking and anomaly detection.
- and much more...

Conclusion

The proposed procedure has two advantages:

1. it focusses simulation effort on the 'interesting' hypotheses.
2. it gives a running estimate of overall significance of the experiment, guarding against multiple testing.

Some questions: how to integrate with a search (e.g. for a changepoint)? Can this procedure arrive at significance in less samples than a theoretically optimal test?

David Donoho and Jiashun Jin.
Higher criticism for detecting sparse heterogeneous mixtures.
Annals of Statistics, pages 962–994, 2004.

Axel Gandy.
Sequential implementation of Monte Carlo tests with uniformly bounded resampling risk.
Journal of the American Statistical Association, 104(488):1504–1511, 2009.

Michael I. Jordan.
Machine-learning maestro Michael Jordan on the delusions of Big Data and other huge engineering efforts.
IEEE Spectrum:
http://spectrum.ieee.org/robotics/artificial-intelligence/machinelearning-maestro-michael-jordan-on-the-delusions-of-big-data-and-other-huge-engineering-efforts, 10 2014.
Accessed: 2015-01-03.


Frederick Mosteller and R. A. Fisher.
Questions and answers.
The American Statistician, 2(5):pp. 30–31, 1948.


R John Simes.
An improved bonferroni procedure for multiple tests of significance.
Biometrika, 73(3):751–754, 1986.


Samuel A Stouffer, Edward A Suchman, Leland C DeVinney, Shirley A Star, and Robin M Williams Jr.
The American soldier: adjustment during army life.(Studies in social psychology in World War II, Vol. 1.).
Princeton Univ. Press, 1949.