MVCAPA with correlations — meeting 3

October 14, 2019

Overview

I have:

- Implemented and run simulations comparing i.i.d. method with AR(1) for a single changepoint.
- Tried to find pruning criteria for AR(1) DP. Not much hope?
- Looked at sequential updating of the optimal subset $\hat{J}_n \to \hat{J}_{n+1}$.
- Extended dynamic programme to AR(2), then AR(r).

Next steps?

Implementation

$Q := \bar{x}\bar{x}^\intercal \circ \Sigma^{-1}$.

$$
\begin{aligned}
B_0(d,k) &= B(d-1,k) \\
B_1(d,k) &= \max(B_0(d-1,k-1) + q_{d,d}, \\
&\qquad\quad B_1(d-1,k-1) + q_{d,d} + 2q_{d,d-1}) \\
B(d,k) &= \max(B_0(d,k), B_1(d,k))
\end{aligned}
$$

- Keep track of which subsets that are optimisers for each (d,k): $J(d,k), J_1(d,k)$.
- $S(s,e) = \max\limits_{k \leq k^* \wedge k = p} (e-s)B(p,k,\bar{x}_{s+1:e}) - \sum_{i=1}^{k} \beta_i$. $O(pk^*) \approx O(p^{3/2})$
- Single change-point with known end-point: $\max\limits_{n-M \leq s \leq n-l} S(s,n)$

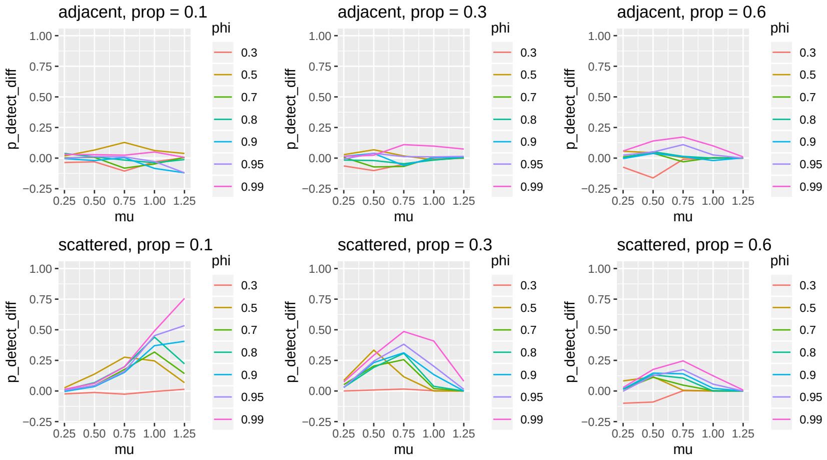
Results

- $n = 200, p = 10, M = 50, l = 5$.
- Calibrated thresholds by a scaling factor at probability of false alarms 0.05.
- True change at $s + 1 = n - 20$ to $e = n - 1$.

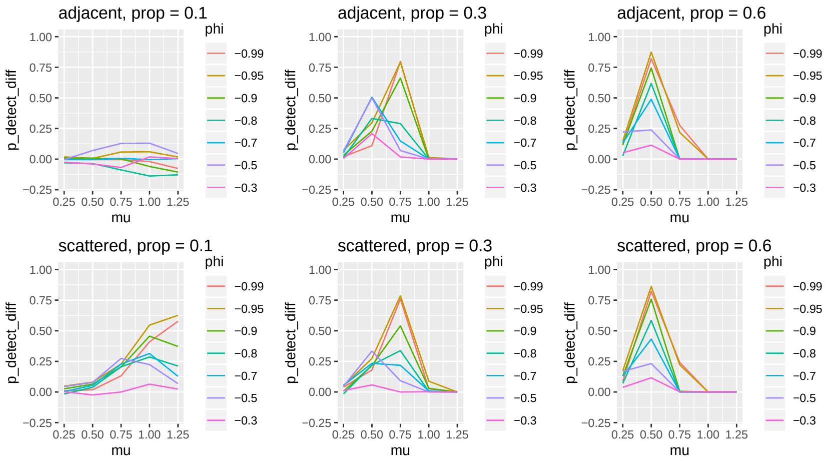
Important for AR(1):

- Difference between boundary and interior points in $\{1, \ldots, p\}$.
- Difference between adjacent and scattered changes.
- Difference between positive and negative auto-correlation.

Results for positive auto-correlation



Results for negative auto-correlation



Pruning criteria. Not much hope?

If $q_{d,d}$ for $d = 1, \dots, p$ are sorted and $\text{sign}(-\rho \bar{x}_d \bar{x}_{d-1} = -1)$, quicker updating from $B(d, k)$ to $B(d, k+1)$ can be done. But very unrealistic assumption that all means are positive for sparse changes, since most means will vary around 0.

Perhaps if taking the penalty into account also? Related to updating of $\hat{J}_n$.

$\hat{J}_n \to \hat{J}_{n+1}$

$$\begin{aligned}\hat{J}_n &= \operatorname*{argmax}_J \, n\bar{x}_n(J)^\intercal A\bar{x}_n(J) - \sum_{i=1}^{|J|}\beta_i \\ &= \operatorname*{argmax}_k \operatorname*{argmax}_{J:|J|=k} nB(p,k,n) - \sum_{i=1}^{k}\beta_i\end{aligned}$$

No simple way of updating max and argmax for any $\bar{x}_n \to \bar{x}_{n+1}$.

Naive strategy: - Sequentially update $\bar{x}_n \to \bar{x}_{n+1}, q_{d,d,n} \to q_{d,d,n+1}$ and $q_{d,d-1,n} \to q_{d,d-1,n+1}$, then rerun DP.

$\hat{J}_n \to \hat{J}_{n+1}$

In simulations, $\hat{J}_n = \hat{J}_{n+1}$ very often. If $q_{d,i,n} \to q_{d,i,n+1} = q_{d,i,n} + \delta_{d,i}$, possible to find a (useful) condition on $\delta_{d,i}$, $d = 1, \ldots, p$, $i = d, d-1$ such that $\hat{J}_n = \hat{J}_{n+1}$ can be guaranteed?

Ideally, something like:

$B(p,k,n) + \max(\delta(J(k))) + \frac{1}{n+1}\sum_{j=1}^{k}\beta_j \leq$ $B(p,\hat{k}_n,n) + \min(\delta(J(\hat{k}_n))) + \frac{1}{n+1}\sum_{j=1}^{\hat{k}_n}\beta_j$ for all $k \neq \hat{k}_n$. (Least and worst perturbation of the maximums for each k.)

Have tested several simple conditions in R without success.

$\hat{J}_n \rightarrow \hat{J}_{n+1}$

If I know $\hat{J}_n|k = \hat{J}_{n+1}|k,\ k = 1, \ldots, l \leq p$, then $\hat{J}_n|k$ can be used to produce $B(p, k, n+1)$ and $S(n+1)$ (penalised savings). Subgoal: Condition on $\delta_{d,i}$ such that the previous holds. Stronger than necessary. Non-changing variates might cause trouble and more reruns than wanted.

Have found conditions for $\hat{J}(d, k, n) = \hat{J}(d, k, n+1)$ for all (d, k), but just as many conditions as rerunning the entire program. Not sure if these conditions can result in a useful bound. Also, much stronger than necessary, as the equality only needs to hold for $d = p$.

$AR(1) \to AR(2)$

$$\begin{aligned}
B(d,k) &= \max(B_0(d,k), B_1(d,k)) \\
B_0(d,k) &= B(d-1,k) \\
B_1(d,k) &= \max(B_{0,0}(d-1,k-1) + q_{d,d}, \\
&\qquad B_{0,1}(d-1,k-1) + q_{d,d} + 2q_{d,d-2} \\
&\qquad B_{1,0}(d-1,k-1) + q_{d,d} + 2q_{d,d-1} \\
&\qquad B_{1,1}(d-1,k-1) + q_{d,d} + 2(q_{d,d-1}q_{d,d-2}))
\end{aligned}$$

$$\begin{aligned}
B_{0,0}(d,k) &= B_0(d-1,k) \\
B_{0,1}(d,k) &= B_1(d-1,k) \\
B_{1,0}(d,k) &= B_0(d-1,k-1) + q_{d,d} \\
B_{1,1}(d,k) &= B_1(d-1,k-1) + q_{d,d} + 2q_{d,d-1}
\end{aligned}$$

$AR(2) \to AR(r)$

Runtime: $O(2^r lp)$.

For all $u \in \{0,1\}^r$, need to define variables $B_u(d,k)$: Maximum savings at dim d with change subset size k for a given "history" u.

$$
\begin{aligned}
B_1(d,k) &= \max_u B_u(d-1,k-1) + q_{d,d} + 2u^\intercal Q_{d,(d-1):(d-r)} \\
B_u(d,k) &= B_{u_{2:r}}(d-1,k-u_1) + u_1(q_{d,d} + 2u_{2:r}^\intercal Q_{d,(d-1):(d-r+1)}) \\
&\vdots
\end{aligned}
$$

Extremely memory heavy. Not straight forward to implement efficiently for any r.

What now?

- More time on finding condition for $\hat{J}_n = \hat{J}_{n+1}$?
- Implement and test BQP solvers?
- Implement and test AR(2) and AR(r)?
- Try harder to find a pruning strategy for AR(1)?
- Block diagonal covariance matrix?
- L1-penalised cost function?
- ...
- Multiple change-points?
- Incorporating lags?
- Point anomalies?
- Penalties?