Prep Meeting 19

Kernel OMP

What is Kernel OMP?

> *"All the steps in OMP could be expressed in the form of inner products between* ***y*** *and* $\boldsymbol{x}_i$*, i = 1, 2, ⋯ , N."*
>
> *"Inspired by this interesting finding, we could play the kernel trick easily for OMP."*
>
> *Therefore, we can draw the conclusion that Kernel* ***OMP achieves sparse representation*** *in any high-dimensional (possibly infinite) Hilbert space F, which is implicitly determined by* ***a proper kernel function****.*

So, we can solve

The KOMP iteratively achieve the high-dimensional SR via performing the following optimization.

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\|_0, \text{ s.t. } \|\mathbf{U}\boldsymbol{\beta} - \Phi(y)\|_2 \leq \varepsilon, \qquad (13)$$

Incredibly fast for any Kernel Space Phi. Note that we can also just do the "normal" inner product, so that we can do K-OMP *incredibly* fast. OMP Algorithm:

Algorithm 2: Kernel OMP

Input:

- A kernel matrix $\Psi \in \mathbb{R}^{N \times N}$ defined in (11).
- A kernel vector $\kappa \in \mathbb{R}^N$ defined in (12).
- A kernel scalar value $\theta = \Phi(\mathbf{y})^\top \Phi(\mathbf{y})$.
- A recovery residual $0 < \varepsilon \ll 1$.
- A sparsity upper bound $0 < \eta < N$.

begin
  Initialize the index set $\Lambda_0 = \varnothing$;
  for $t \leftarrow 1$ **to** η **do**
    $\lambda_t = \underset{i=1,\ldots,N}{\operatorname{argmax}}(\kappa_i - \beta_t^\top \Psi[i, \Lambda_t])$;
    $\Lambda_t = [\Lambda_{t-1} \ \lambda_t]$;
    Obtain the subset of Ψ and κ:
      $\Psi_t = \Psi[\Lambda_t, \Lambda_t], \quad \kappa_t = \kappa[\Lambda_t]$;
    Solve the least-squares problem:
      $\beta_t = \Psi_t^{-1} \kappa_t$;
    Calculate the new residual:
      $\|\mathbf{r}_t\|^2 = \theta - \kappa_t^\top \beta_t$;
    if $\|\mathbf{r}_t\|^2 < \varepsilon^2$ **then** break;
  Retrieve signal β according to β_t and Λ_t;
end
Output:

- Kernel SR coefficient $\beta \in \mathbb{R}^N$.

Matrices: N x N (so p x p) instead of T x p. Least Squares Problem is very easily solved!
Downside: We cannot explicitly calculate residual, but we can do residual norm squared.

Rewritten to DAG-K-OMP.

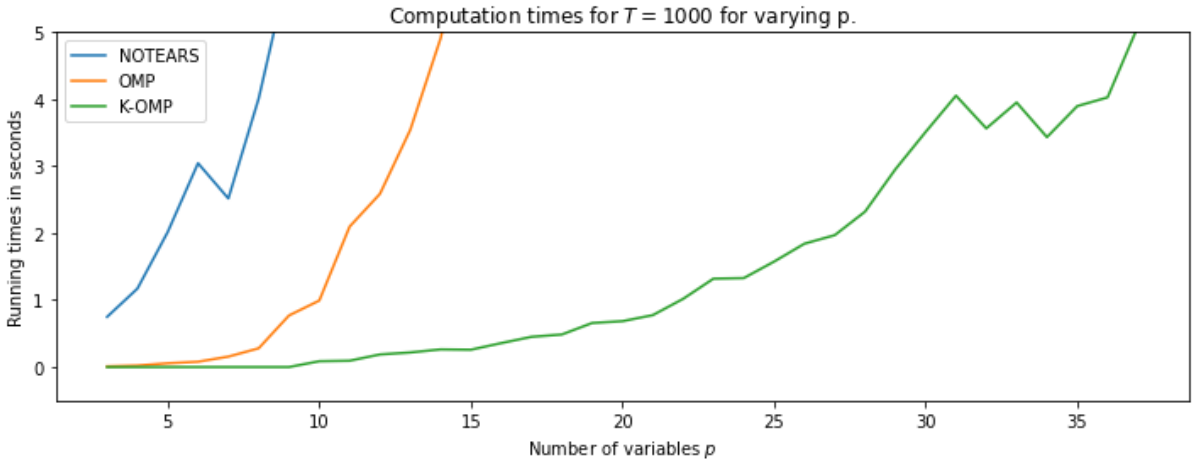
Results

<u>Massive Improvements in Time.</u>

Especially for large T, enormous time increase. Our data matrices are now condensed to (p, p) rather than (p, T). K-OMP is about 100 – 1.000 times faster than OMP, and about 1.000.000 times faster than NOTEARS.

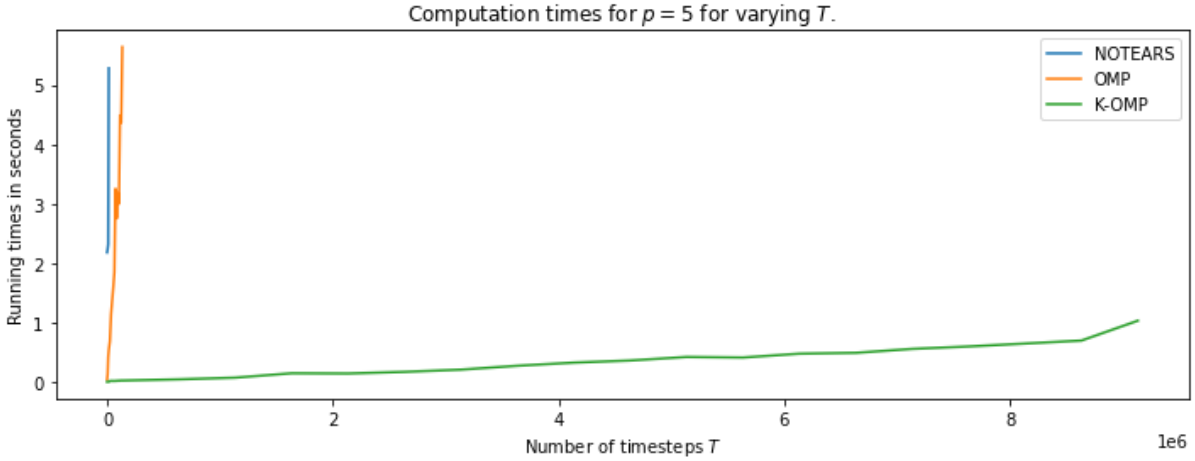
Plot of time varying for T = 10 until 100.000, p = 3.

How many variables **p** can we have with **T** = 1000 within **5** seconds?



NOTEARS: 8, OMP: 14, K-OMP: 37.

How many timesteps **T** can we have with **p** = 5 within **5** seconds?



NOTEARS: 15.000, OMP: 150,000, K-OMP: ?? 10,000,000 took 1 second.

Conclusion: K-OMP seems to be thousands times faster!

Data Transformations

Linear Combinations of data transformations.

We can now also easily solve

The KOMP iteratively achieve the high-dimensional SR via performing the following optimization.

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\|_0, \text{ s.t. } \|\mathbf{U}\boldsymbol{\beta} - \Phi(y)\|_2 \leq \varepsilon, \qquad (13)$$

For any Kernel function Phi.

Quick example: Phi , $\Phi(\cdot) = exp(\cdot)$, such that we have $Y = \exp(X)\ W$ rather than Y = X W.
We can quickly solve this. Doing this with Kernel Phi and W = [[1, 2, 3][4, 5, 6][7, 8, 9]] yields it back:

```
[[1.04 1.87 2.93]
```

```
 [4.05 5.04 5.97]
 [6.89 8.06 9.07]]
```
However, this is just an easy data transformation, nothing nonlinear. Perhaps this is still useful though? What about “real” Kernel functions, such as RBF or Polynomial Kernel?

X = 2Y or Y = 1/2X?

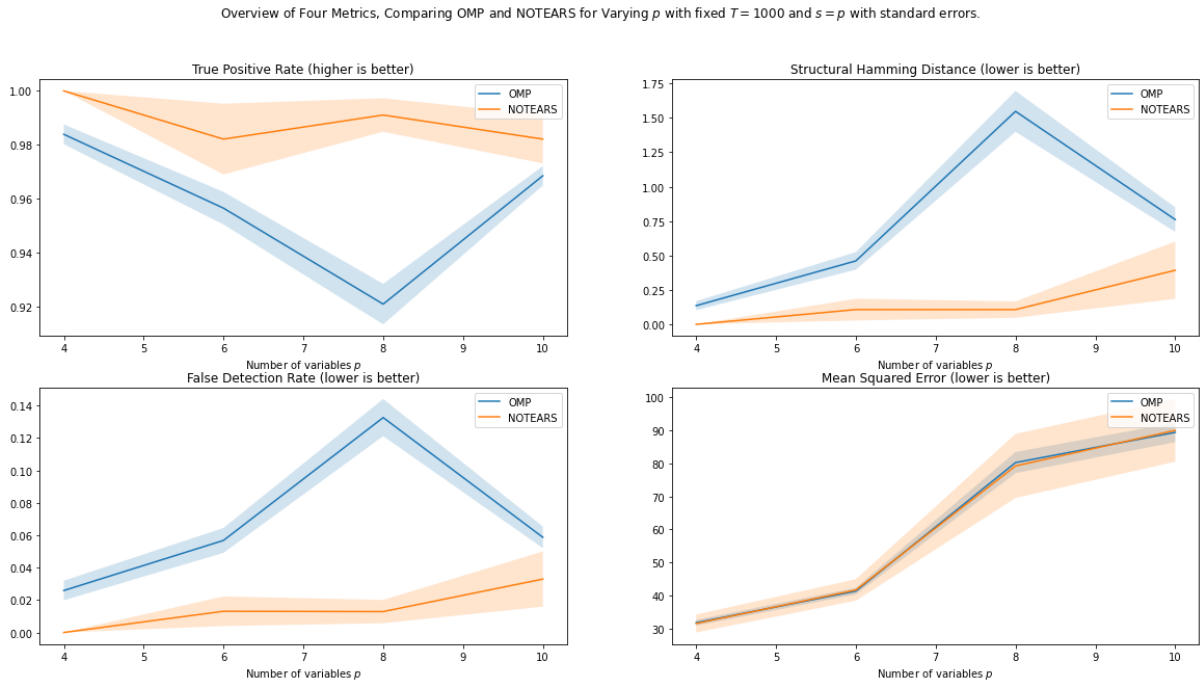
X = 2Y + noise(1)

Y = noise(4)

OMP for SEM

Benchmarking between OMP and NOTEARS

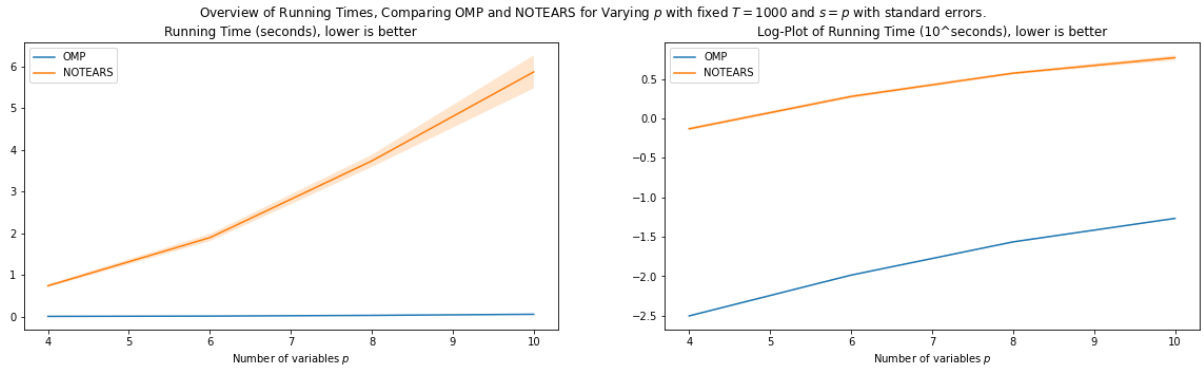
Overview of p = [4, 6, 8, 10] with T = 1000, ER-1-Graph, ~15 samples for NOTEARS, 150 for OMP.



Conclusion: OMP significantly faster than NOTEARS, and the results are not that far off! It seems that NOTEARS has the edge, especially in structure, but they are comparative in Mean Squared Error, which is also the quantity that NOTEARS claims to minimize.

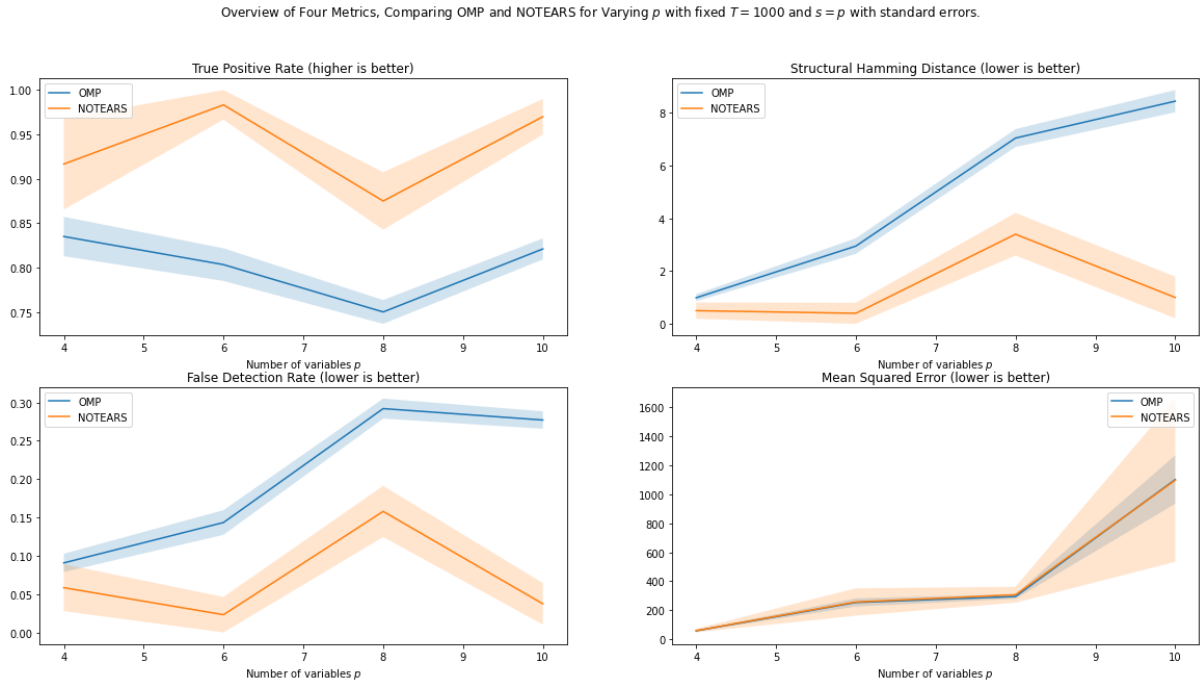
NOTEARS was always better in the structural metrics, but OMP was better in 3 out of 4 times in MSE. However, this better MSE could also be simply because it had more edges, but this does not seem to be the case, as SHD was around 0.5 when it was better.

The greedy-ness of OMP implies that more often an incorrect edge is taken, and this process cannot be reversed in OMP, but for NOTEARS, this is possible in the optimization procedure.

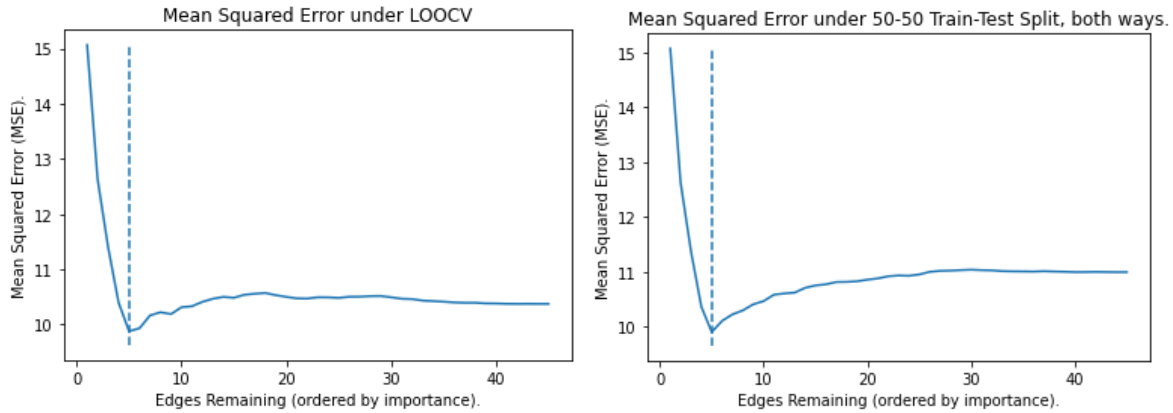


OMP was about a hundred times faster as well. This difference will most likely increase for larger T.

When we increase the number of edges, NOTEARS takes off in structural scores, remains competitive for MSEs. OMP better in MSE for 3 out of 4 Ps tried, but significantly worse for structural.



Cross Validation for SEM on T = 100



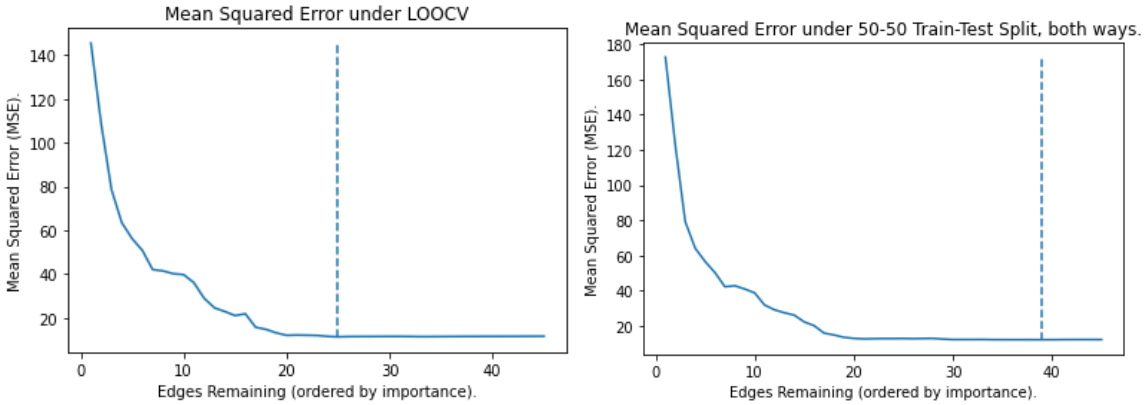
X = XW, T samples. LOOCV: Train on X[-i], Compare MSE on X[i], X[i] – X[i]W. Do this T times, average results.

Train-Test Split both ways.:

1. Train on X[first half], Compare MSE on X[second half].
2. Do it also the other way around.

Same expected result as VAR, _-. However, as OMP is not as effective on SEM, I think the results will not be as good. Interestingly, 50-50 Split Both Ways seem to be a more effective choice than LOOCV.

More difficult one: LOOCV seems better, 20 edges.



Stopping Criterion Proofs

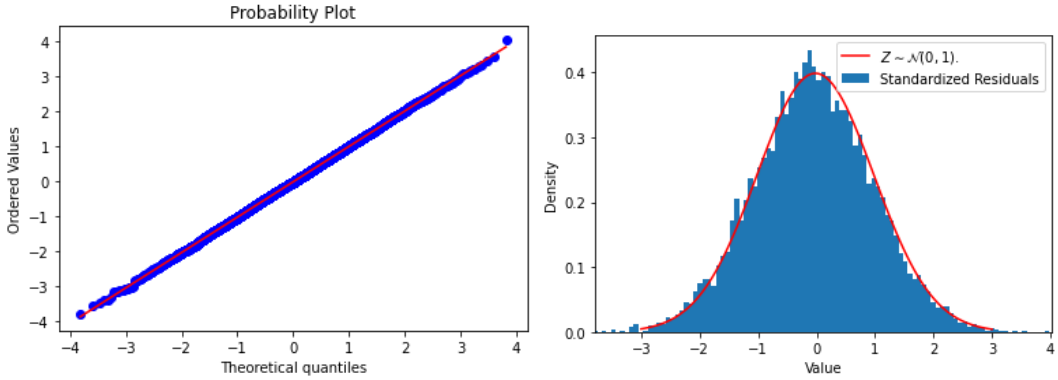
Distribution of residuals under Model Assumption.

Distribution of the residuals is normal under the population setting. The computable quantity for Kernel-OMP is the residual squared, which is also used as a stopping criterion in other papers.

Assume any *Gaussian additive noise model*, that is, $Y = X + E, E \sim N(0, \Sigma)$

Then, we know that **in the population setting**, $\|\varepsilon\|_2^2 \sim N(T * Tr(\Sigma),\ 2\,T * Tr(\Sigma)^2)$.

Verification: 5000 residuals, for a SEM with T = 50000 (close to population). Normality test: p = 0.96.



What do we gain from this? When we know the distribution of the residuals, we know when we can "expect" to stop adding atoms, so we get a nice stopping criterion, but we need to know what the additive noise is.

Some (not very strong) statement could be made like: Assume we have a large enough sample size. Then, with probability η, the stopping criterion $\|\varepsilon\|_2^2 \leq tol(\eta)$, will recover all true coefficients. Here, $tol(\eta) = \left(z_\eta + \mu_\varepsilon\right) * \sigma_\varepsilon$, where $\mu_\varepsilon = T * Tr(\Sigma)$, and $\sigma_\varepsilon = \sqrt{2T} * Tr(\Sigma)$.

However, using the residual as stopping criterion is often *not desirable*, as you do not know beforehand how well your model will fit. If you set your stopping criterion at 1.000, it could very well be that you never reach this.

DON'T FORGET

1. **Tuesday 14:15 Feb 01 Onwards, First One Online.**
2. Mid evaluation, progress meeting at the end of Q2 (in one or two weeks, 30 mins.)?

Extra Stuff

Random Walk Puzzle: Quite unrelated, but I briefly discussed using a random walk on the set of permutation matrices in the thesis, and I wanted to show that the initial permutation is important. E.g., if we start at P = [[p]] = [1, 2, …, p], it may take a long time to reach its reverse -P := [p, p – 1, …, 1].

Now, the question: Assume we can do random transpositions of two integers, what is the expected number fo steps to go from P to -P? It seems to be approximately p!.

We have for p = 2, 3, … that the answer is 1, 5, 27, 128, …, however I have not found a closed form solution and it irks me to not have an answer.