Prep Meeting 18

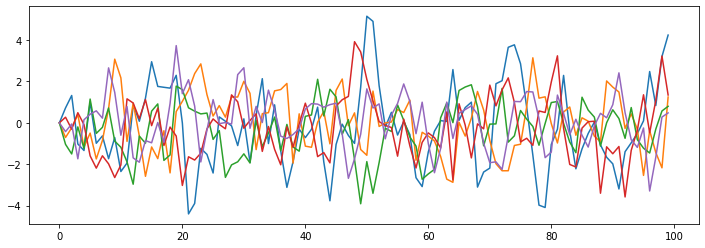
1. Wrote Some more.
2. OMP for SEM.

**What did I do:** OMP on SEMS, such as Sachs and simulated data, compare to NOTEARS.

**Results:** OMP not that far off of NOTEARS. OMP has some benefits, such as an ordering of importance, and allows for regularization using the threshold.

Also: OMP in the Noiseless setting,

# Bootstrapping.

**What was the problem**. Basically, DAG-OMP continues until we have a *dense* DAG. However, we can most likely achieve a comparable predictive power with a much *sparser* DAG. The question, where do we cut off?

**Example**

True W:

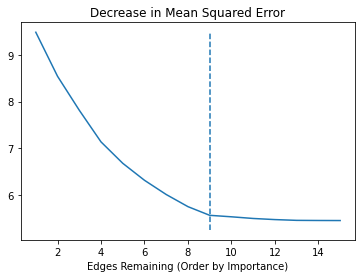
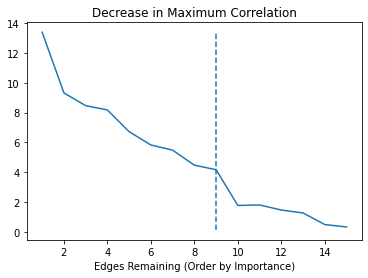
[[ 0.4 0. 0. 0. 0. ]

[-0.44 0.4 0. 0. 0. ]

[ 0. 0. 0.4 0. 0. ]

[ 0.58 0. 0. 0.4 0. ]

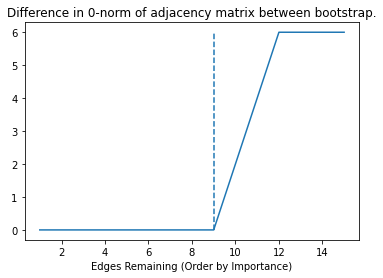
[ 0. 0.55 0. -0.61 0.4 ]].

**** ****

**Approaches tried.** The procedure looks as follows. Let us assume we have one set of data **X** with **p** dimensions. We apply OMP on this data until we have a *dense* DAG, so a DAG with **p(p+1)/2 := K** edges. Let **W(k)** denote the matrix on **k** edges that the OMP algorithm returns on the data **X**, for **k = 1, …, K**. Now, the question is, which **k** should be pick?

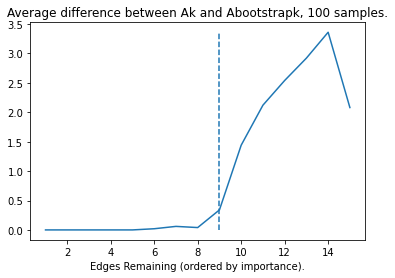
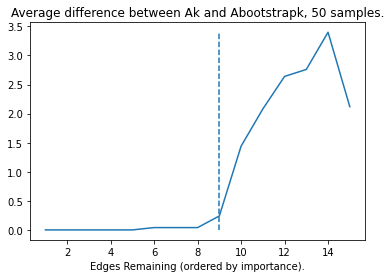
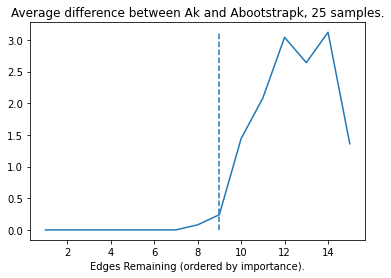
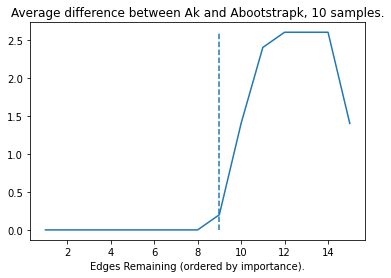
*The bootstrapping approach*.

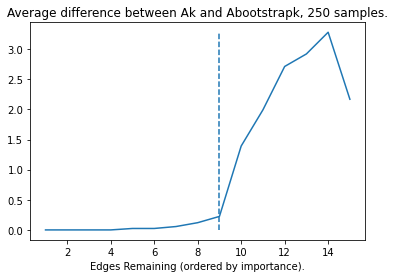
1. For each **k**, we generate new data according to our VAR(1) model. For this, we use the matrix **W(k)** that we got from OMP. So, we will get new data **X(k)**.
2. We now apply OMP on this simulated data, yielding again **K** matrices, which we call **, l = 1, …, K.** This yields in total **K2** matrices from a total of **K** OMP procedures on newly generated data **X**, and a total of **K** matrices from one OMP procedure on the original data **X**.

*Suitable metric.* Now, the question is, which metric will help us most in recovering a suitable k? The idea is that if **W(k)** is a good estimator, then will closely resemble it. We therefore decide that a suitable metric is the *difference in edges*. Since we expect quite some difference in *value* due to *noise*, we will pick the difference in their adjacency matrices, so

*Suitable metric v2.* The reasoning is that *if an edge is important, then it will be re-discovered after bootstrapping*. This implies that when our *k* is too low, we still only have important edges, and therefore, we will re-discover them all. The interesting part is when we reach the ***first unimportant edge***. When we reach this edge, it is not important enough that it will be rediscovered. Therefore, as soon as we reach such an edge, we know that we have reached our optimal value. Therefore, we need to award a small bonus for *how many edges we can add, while still re-discovering all edges*.

*Bootstrapping multiple times.* An advantage of bootstrapping is that we can do step 2. *multiple times*. That is, we can sample more than just one data **X(k)** and performing OMP on this data, for each **k = 1, …, K.** We can also run sample a total of **N** times our data **X(k)**, and performing OMP **N** times on this data, for each **k = 1, …, K.** We can then *average* the **N** outcomes into one average, for each **k**, yielding more reliable bootstrapping estimates.





*Cutoff* As you can see, it is not so clear cut anymore. However, we propose the following threshold: “More than half of the time, the edges coincide with the re-discovered edges. This translates to the cut-off of at **y = 1.0**, which here yields the correct answer.

*Some questions*:

1. What should the length **T** be of the bootstrapped time series **X(k)**? For now, I argued, equal to the original data length **X.** If you pick **T(k)** too large, then the metrics will not make sense. For large enough time series, we will perfectly recover any matrix **X**(k), so we need to make sure there is enough randomness left in the time series.
2. Seems to work well when the “visual plots” (decrease in MSE, decrease in gain) also work well, so it might not be very useful when.
3. Other metrics that did not work well:
   1. Any other norm than the zero norm in adjacency matrices, e.g. p-norm between W\_k and W\_bootstrap\_k.
   2. MSE when using W\_bootstrap\_k on X.

# OMP for SEM

Seems to work quite well. Problem is now more difficult. As the *directionality* of the edge is more difficult to determine. Do we have X = 2Y or Y = X/2? Still, results seem comparable to NOTEARS.

**Comparing** How do you compare OMP to NOTEARS, as there are 1 and 2 tuning parameters resp. Difficult to get a fair comparison. Also depends on *metric*. We like *predictive power / model fit*, they prefer *structural / graph recovery* as metric.

**Simulated data** Simulate T samples p-dimensional data, using a p x p dimensional coefficient matrix W. Estimate using OMP and using NOTEARS, compare results. Ideally, apply OMP to the graphs of NOTEARS, but their experiments are not included ☹.

OMP and NOTEARS seem quite comparable, NOTEARS has the edge, but has more drawbacks: less clear what happens, order of the variables is not clear, thresholding is less mathematically grounded.

OMP seems to perform better w.r.t. MSE, NOTEARS better w.r.t. accuracy, tpr, tnr, etc.

**Real-life data**. Used a real-life dataset that works well with SEM models, also used in NOTEARS, by Sachs et al (1000+ citations, mainly by learning of Bayesian networks). Compared OMP to NOTEARS, but some caveats:

* NOTEARS claims there are 20 edges, most literature claims 17 edges, how to compare?
* I was unable to *exactly* reproduce NOTEARS, no info given, only SHD and #edges.
* Normalize / Zero-Mean? NOTEARS zero-means its data beforehand.

**Results:** The results were quite competitive for both. The lambda regularization did not affect results (due to 7000+ samples), so only the threshold mattered. For any threshold, we saw that the ‘optimal’ threshold for NOTEARS was comparable to the ‘optimal’ OMP, so OMP is not bad for SEM it seems.

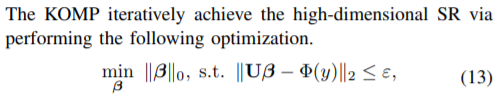
# Kernel OMP

OMP also suitable with Kernels!

*“All the steps in OMP could be expressed in the form of inner products between* ***y*** *and* ***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.*



Using a Kernel space:



Interestingly quite old (2011), not often cited, yet it is very fast and perhaps is a good solution to transform OMP to nonlinear problems!

# Proofs

Did not work on that yet, unfortunately.

# DON’T FORGET

1. **Unavailable next Tuesday 12:45 – 16:15, can we reschedule?**
2. Online / Offline?
3. Teaching next quarter unfortunately Tuesday 12:45 – 13:30 and Wednesday 08:45 – 10:30. **When do we meet in Q3?**

Monday **January 24th** 13:30 Online

**Tuesday 14:15 Feb 01 Onwards Online**

1. Graduation plan, officially approved!
2. Mid evaluation, progress meeting at the end of Q2 (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.