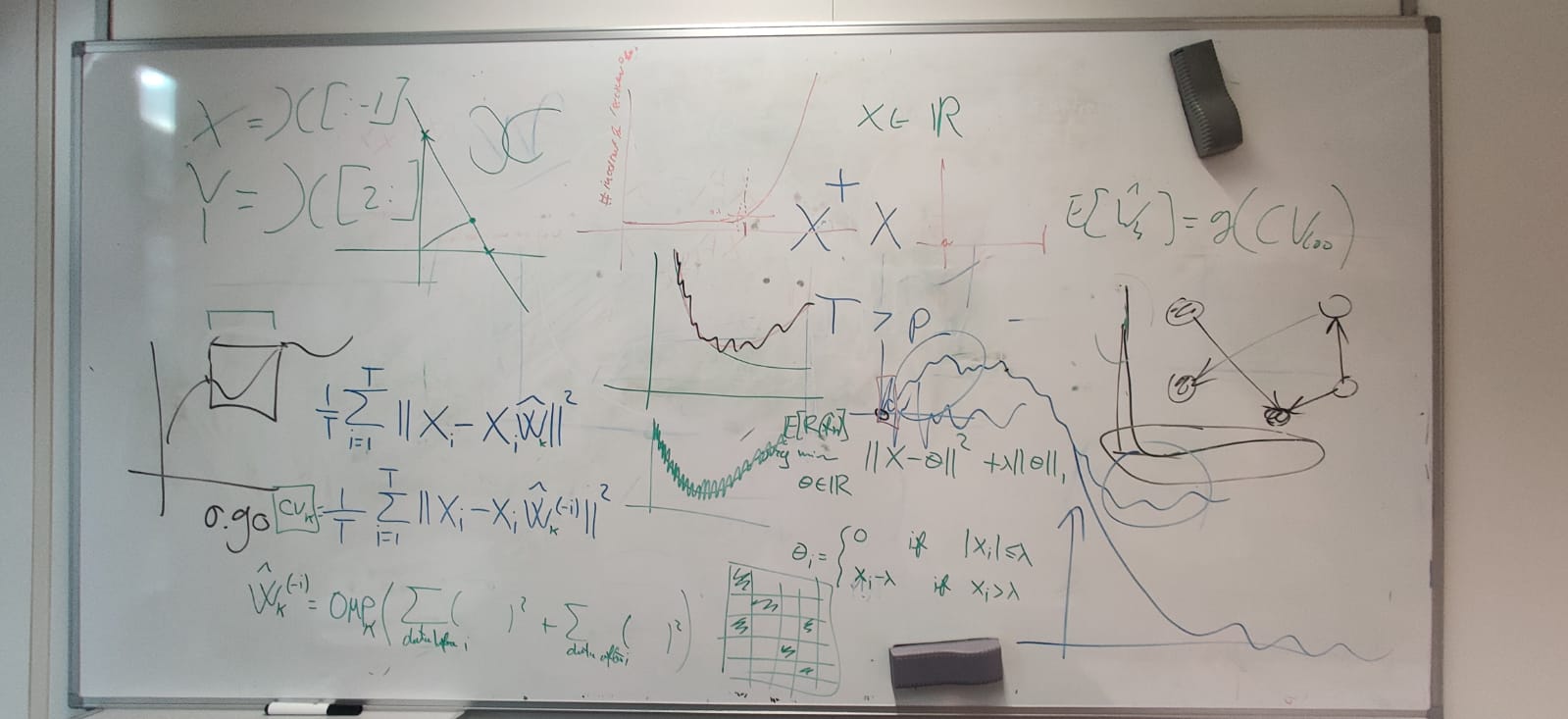
Prep Meeting 24



# Ill-posedness when few samples

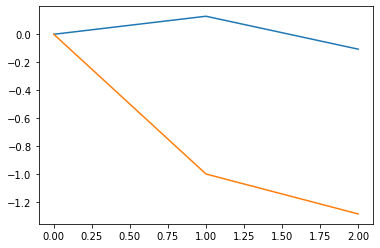
Indeed, when we are trying to estimate *x* coefficients from *y* samples, we will encounter issues when *x > y*. In this scenario, the inverse is not defined.

We also see this in the condition number. The condition number of the matrix to inverse increases when we add more coefficients.

Let us consider a setting where ***p = T – 1***. When we continue until a full DAG, issues will arise when we will estimate the *p*th coefficient of a row *j* of *W*. The off-diagonals are equal to each other in Psi. There, we see that the condition number shoots up.

## 2D Example

Consider the scenario where *p = 2* and hence, *T = 3*. OMP will first estimate the diagonal, and there are no issues. However, once we will estimate the first off-diagonal, we have that the dimension of Psi is equal to T – 1, and we see that the condition number of Psi shoots up to 1e16, which means the matrix is invertible in practice. See the following example:

True A:

[[0.85 0. ]

[0.85 0.85]]

**Iteration 1.**

Gains: Shape of Psi\_F: (1, 1).

[[0.10625419 1.2831491 ] Psi\_F: [[0.01659077]]

[0.10625419 1.2831491 ]]. Condition Number: 1.0.

Beta\_1:

[[ 0. -9.962]

[ 0. 0. ]].

**Iteration 2.**

Gains: Shape of Psi\_F: (1, 1).

[[ 0.10625419 -1. ] Psi\_F: [[0.01659077]]

[-1. 0. ]]. Condition Number: 1.0.

Beta\_2:

[[-0.825 -9.962]

[ 0. 0. ]].

**Iteration 3.**

Gains: Shape of Psi\_F: (2, 2).

[[-1. -1.] Psi\_F: [[ 0.01659077 -0.12855476]

[-1. 0.]]. [-0.12855476 0.99611551]]

Condition Number: 1.83e+16.

Beta\_3:

[[-0.825 -0.163]

[ 0. 1.265]].

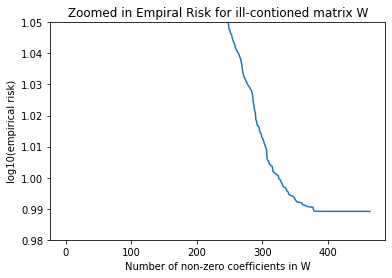
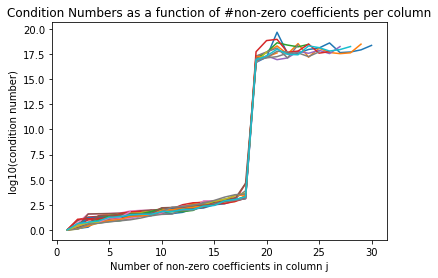
Hence, using the inverse in the computation is infeasible when *p < T*. Therefore, we might go for the *pseudo-inverse*.

We also see that the *gain* for all the coefficients is of the order 1e-16, which is numerically zero. Therefore, we should actually not even add these, as they yield *no significant gain*.

## 10D Example with Plot

Consider a larger example to see what happens to the mean squared errors when we resort to the pseudo-inverse and we have a *p << T*.

We see that in the first *10* iterations, a coefficient is assigned to each column, so there is no issue. However, at the *11th* iteration, we will for the first time encounter a column with *T – 1* non-zero coefficients, problems are encountered. Again, the condition number shoots up to 1e16. This is the case for all iterations until we have a DAG on *(10 \* 11) / 2 = 55* edges, so for the 45 added coefficients. What happens to these?



We also see that the gain is 1e-16 for the coefficients that will cause problems, whereas these gains are originally of the order 1e-2. Therefore, these 1e-16 can considered to be zero.

Interestingly, the MSE does not change, **at all**. This makes sense as the gain is 1e-16, meaning that the gain is simply zero. Not even to the smallest decimal place. I think this is the issue that Rui and Alex warned for.

Furthermore, the true risk also increases when using this pseudo-inverse. However, it is unsure whether this is even meaningful.

## When are we in trouble?

We are in trouble when we are about to estimate the *T – 1*th coefficient of a column. So, when we want to continue until a dense DAG, assuming a lower triangular matrix, we can only estimate max(T – 2, p – j + 1) elements for each column j = 1, …, p. This total sums to.

Total = (T – 2)(p – (T – 2) + 3) + (T – 2)(T – 1) / 2.

Which is also shown in the drawing to the right. Therefore, we know that we can only meaningfully estimate Total number of variables before all gains become zero.

### Proposed Solution

When the number of non-zero coefficients in a column reaches *T – 1,* there are two options:

1. We simply do not continue anymore. We set fix the other coefficients to zero and will never consider them again. This means that we will never have a full DAG. Furthermore, from a structural point of view, we might not retrieve the original edges. However, recovering the true edges is impossible.
2. When continue with the pseudo-inverse. The empirical risk will not improve, and it also feels strange to pick edges that have zero gain. No sensible edges are to be added.

# Bootstrapping

Some resources:

[Properties of the nonparametric autoregressive bootstrap](https://www.econstor.eu/bitstream/10419/61278/1/721981283.pdf)

[The bootstrap approach to autoregressive time series analysis](https://repository.up.ac.za/bitstream/handle/2263/29608/dissertation.pdf;sequence=1)

[The Bootstrap for the Functional Autoregressive Model FAR(1)](https://kluedo.ub.uni-kl.de/frontdoor/deliver/index/docId/4410/file/Thesis.pdf)

[New Bootstrap Method for Autoregressive Models](https://www.researchgate.net/publication/263648838_New_Bootstrap_Method_for_Autoregressive_Models?enrichId=rgreq-f807f223ee86e0e879574e78ff8379a4-XXX&enrichSource=Y292ZXJQYWdlOzI2MzY0ODgzODtBUzoxNDU0NzQ2NDQzNTMwMjVAMTQxMTY5NTI2NjkzMA%3D%3D&el=1_x_2&_esc=publicationCoverPdf)

[Discussion: Bootstrap methods for dependent data: A review](http://mapageweb.umontreal.ca/goncals/documents/Goncalves-Politis-2011.pdf)

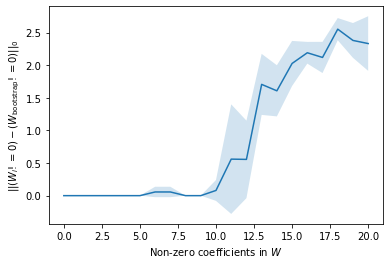
[3.3 Boostrap methods for time series](https://lbelzile.github.io/timeseRies/boostrap-methods-for-time-series.html)

Procedure:

1. Do OMP, and get a sequence of Ws = [W\_1, …, W\_K], where W\_i has i non-zero coefs.
2. For each W\_i:
   1. Compute residuals from X, centralize them.
   2. For the desired amount of iterations:
      1. Use these with W\_i to generate ~X.
      2. Do OMP on ~X, get sequence ~Ws.
      3. 1: Compare ~Ws\_i to W\_i. See if ~Ws\_i recoverts W\_i
      4. 2: Compare W\_i to ~W\_i. See where MSE diff stabilizes
   3. Save means / covariances of 1 and 2.
3. Plot 1. And 2. With their covariances.

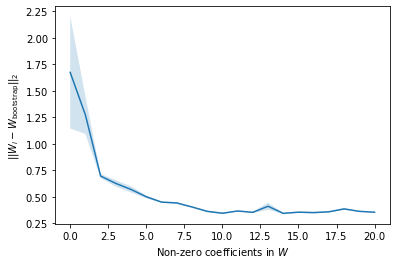
Bootstrap: We estimate a time series X using W. We calculate the residuals, and we simulate with W using the old residuals. We simulate multiple boostrap sets, and calculate their W.

Originally, for model selection, we argued that if an entry w is important enough, then it will be recovered from the bootstrap sample. If it is not important enough, then it will not be recovered.

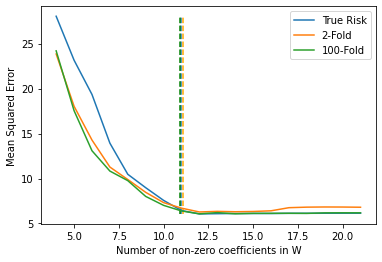


Additionally, we are now checking at the difference between W and its bootstrap in the norm. If we are close to the true datagenerating matrix, the residuals should be good, and also the estimation procedure should become **stable**.

We are therefore looking for a point where the difference between the estimated W and the bootstrapped W will become stable.



Comparing these to the cross-validation settings, we get similar results.



We see that #non-zero = 11 is really the first part where the line becomes flat. All parts after that achieve similar scores, but are not detrimental. This is similar to the bootstrap that recognizes incorrectly recovered. We see that from 10 onwards (which corresponds to 11 here), the line is going up, but the covariance indicates that there still quite often we recover all until 12 (which corresponds to 13 here). So, that bootstrap approach gives a nice way to look at the importance of edges.

# Time Series Regularization

Some sources:

[A survey of cross-validation procedures for model selection](https://arxiv.org/pdf/0907.4728.pdf)

[A note on the validity of cross-validation for evaluating autoregressive time series prediction](https://www.sciencedirect.com/science/article/pii/S0167947317302384)

[DATA-DEPENDENT ESTIMATION OF PREDICTION FUNCTIONS](https://onlinelibrary.wiley.com/doi/10.1111/j.1467-9892.1992.tb00102.x)

[Hyndman Blog: Cross-validation for time series](https://robjhyndman.com/hyndsight/tscv/)

[Prediction intervals from bootstrapped residuals](https://otexts.com/fpp2/prediction-intervals.html#prediction-intervals)

[Forecasting: Theory and Practice](https://tue.on.worldcat.org/atoztitles/link?url_ver=Z39.88-2003&ctx_ver=Z39.88-2003&ctx_enc=info:ofi/enc:UTF-8&rft_id=info:doi/10.1016%2fj.ijforecast.2021.11.001&rft_val_fmt=info:ofi/fmt:kev:mtx:journal&rft.aulast=Petropoulos&rft.aufirst=F.&rft.issn=01692070&rft.isbn=&rft.volume=&rft.issue=&rft.date=2022&rft.spage=&rft.epage=&rft.pages=&rft.artnum=&rft.title=International+Journal+of+Forecasting&rft.atitle=Forecasting%3a+theory+and+practice&rfr_id=info:sid/Elsevier:Scopus)

[Approximate leave-future-out cross-validation for Bayesian time series models](https://www.tandfonline.com/doi/full/10.1080/00949655.2020.1783262)

[A cross-validatory method for dependent data.](https://www.ssc.wisc.edu/~bhansen/718/BurmanChowNolan1994.pdf)

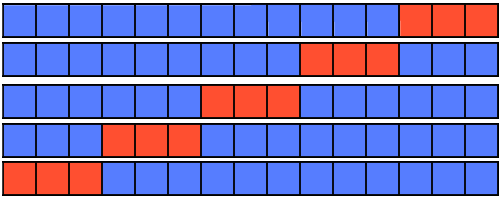
## Cross Validation

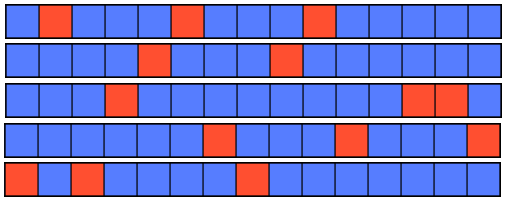
Options:

1. Out Of Sample (OOS) evaluation. We simply use the last part as our test sample. We also remove the middle to remove dependency. We treat this *as if* we have two parts. A training sample and a test sample *that is independent of the training sample*, as if the test set is *out of sample*. We can “make” the test set independent by removing “enough” data sets in between.

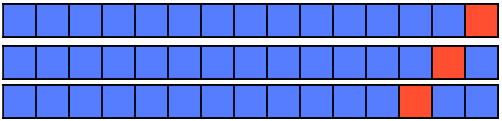


1. K-fold cross validation. We randomly distribute our data in *K* parts. We train on *K – 1* folds and test on 1 fold for each of the *K* folds. For example for *K = 3*. Note, this can be done like below, or randomly.

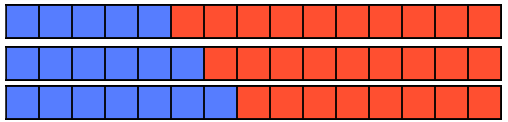




1. Leave-One-Out cross validation (LOOCV). We randomly leave one data element out, and train on the remaining folds and test on the left out fold. We do this for each of the elements. In fact equal to *K* – fold cross validation where now *K* is the number of samples.



1. Leave-Future-Out cross validation. We only train on the future. This was also mentioned in the blog by Hyndman: <https://robjhyndman.com/hyndsight/tscv/>



## Extra Things to keep in mind

### M-Step Ahead Prediction

Perhaps an extra measure, instead of focusing on how well we can predict the next measure using the current measure, we might want to investigate how well we can predict the Mth next measure using the current measure. Might make more sense in longer dependent time series, as an AR(1) model is not very useful at M-step ahead prediction for M > 1.

### Remove Dependencies

We remove the dependencies by leaving data between the test and training set out. This is often done by specifying an integer *h*, and then removing all entries from the training set that are at most *h* distance away from the training set. Example:



A more detailed description is given in [Burman, Chow, and Nolan (1994)](https://www.ssc.wisc.edu/~bhansen/718/BurmanChowNolan1994.pdf). Here, they consider three ways to do *h-block cross validation*.

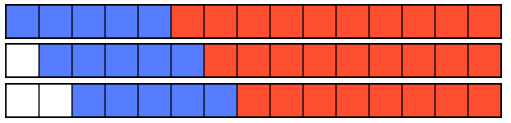
1. Remove the *h* training samples preceding the fold to be tested.
2. Remove the *h* training samples postceding the fold to be tested.
3. Remove the *h* training samples preceding and the *h* training samples postceding the fold to be tested.

Note, this can also be applied to a LOOCV fashion, where we train on T – 2h – 1 samples and test on one sample.

How to select h? Enough so that the dependencies are removed. For an AR(p) process, h = p generally.

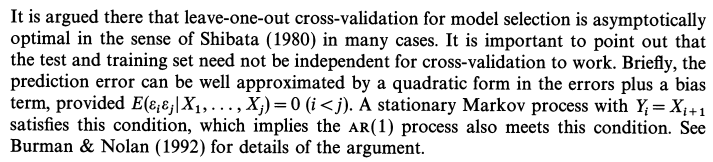
### Sliding Window

Sliding Window. Only use the sliding window of length *k* to estimate. Keeps the length of the training set the same for all model fits. However, does not utilize the full availability of the data.



### Leave One Out Cross Validation for AR models

Subset of K-Fold CV for K = T – p, so its results also apply here. Also, see this old excerpt from



Source: [Shibate (1980)](https://projecteuclid.org/journals/annals-of-statistics/volume-8/issue-1/Asymptotically-Efficient-Selection-of-the-Order-of-the-Model-for/10.1214/aos/1176344897.full)

### K-Fold CV for AR models

Paper by Bergmeir, Hyndman, and Koo (2015) they show that K-fold cross validation (and therefore also LOOCV) is possible in purely autoregressive models as long as uncorrelated errors are considered.

1. Assume the AR(p) process is *stationary* and *ergodic*.
2. The LS-estimator is a consistent estimator.
3. The errors are MDS => Satisgied for i.i.d. Gaussian noise.

Then,

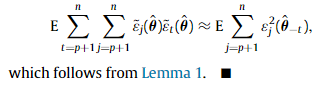
“The K-Fold cross validation estimator is a consistent estimator.”

How their proof works: They write out the MSE as an integral, and state that the emprirical distribution F\_n is asymptotically identical to the data generating distribution F. Remains to show:

 is equal to .

Error of LS estimate. Error of LOOCV estimate.

Rewriting this yields the requirement that



Source: <https://www.sciencedirect.com/science/article/pii/S0167947317302384>

References:

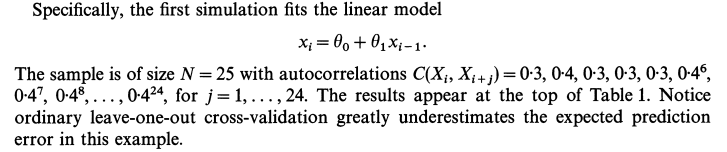
[Data dependent estimation of prediction functions (Burman, Bolan, 1992):](https://www.ssc.wisc.edu/~bhansen/718/BurmanNolan1992.pdf)

### Leave Future Out Cross Validation

<https://www.tandfonline.com/doi/epub/10.1080/00949655.2020.1783262?needAccess=true>

LOO-CV can use the future to predict the past, which makes the LOO-CV estimates overly optimistic in practice.

## Cross-Validation Gone Wrong

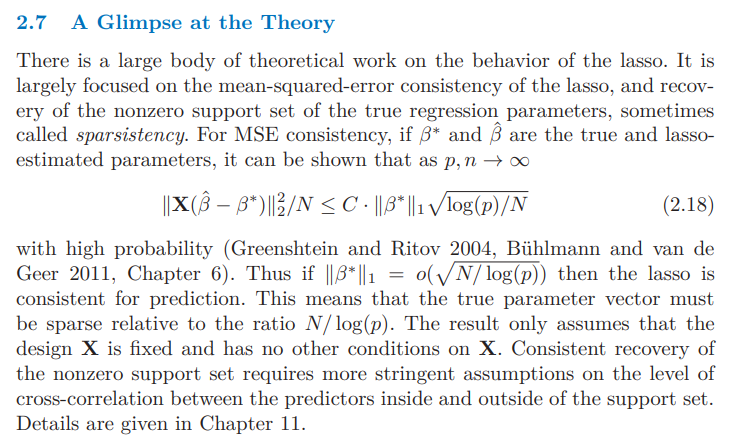


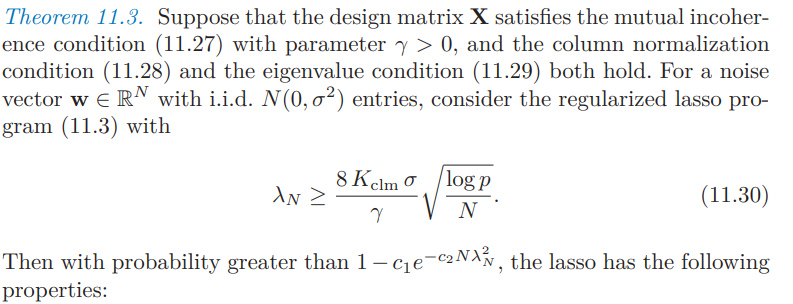
# CV for model selection

# LASSO-DAG Regularization

Can we capitalize on DAG-ness?

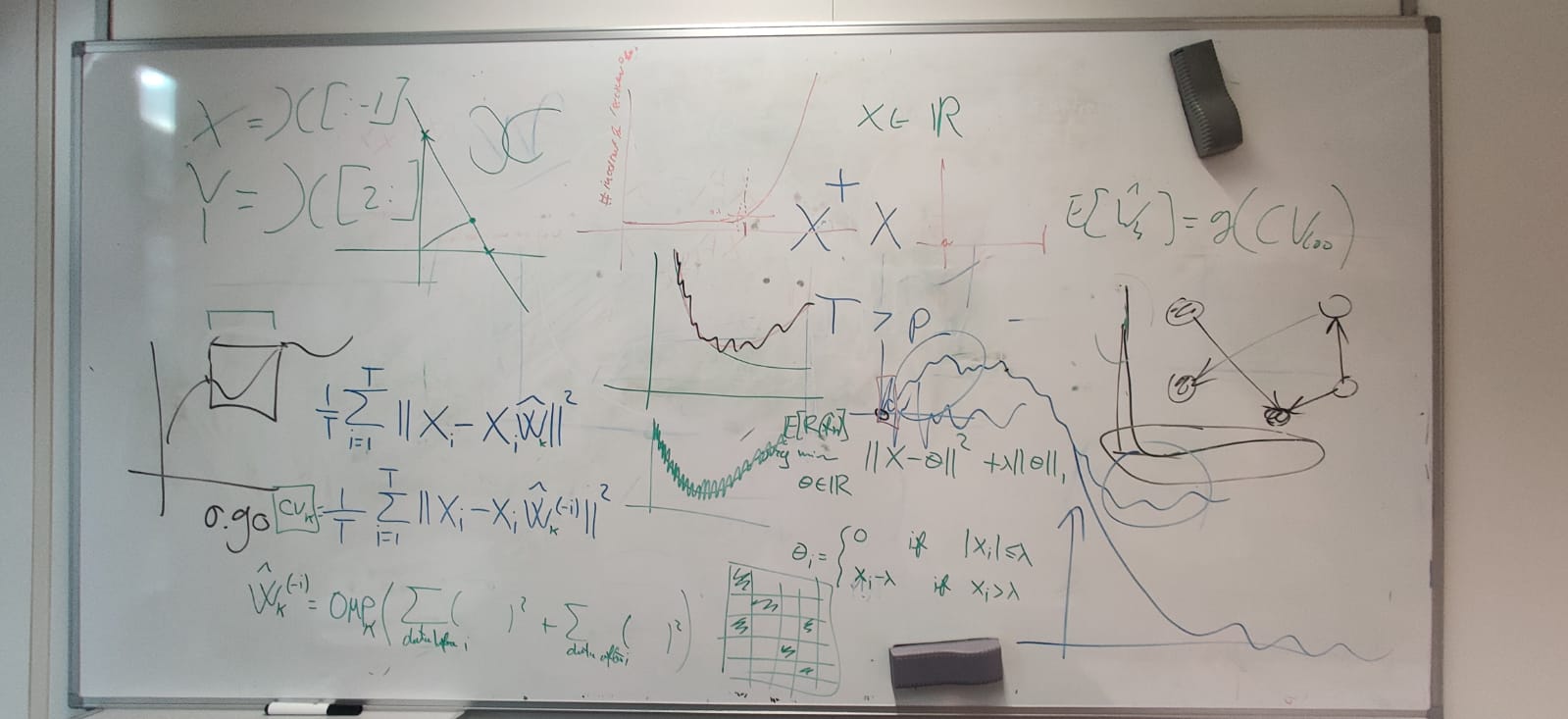
<https://arxiv.org/pdf/0806.0145.pdf>





Sources: [Statistical Learning with Sparsity The Lasso and Generalizations](https://hastie.su.domains/StatLearnSparsity_files/SLS.pdf)

# Bootstrapping



# Usage of DAGness.

So far, we have not really utilized the DAG-ness anywhere, which is unfortunate as it is such a big part of the assumptions. So far, what do we gain?