Insert here your thesis' task.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Probabilistic algorithms for computing the LTS estimate

Martin Jenč

Department of Applied Mathematics
Supervisor: Ing. Karel Klouda, Ph.D.

March 25, 2019

Acknowledgements

THANKS to everybody

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on March 25, 2019 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Jenč, Martin. *Probabilistic algorithms for computing the LTS estimate*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

V několika větách shrňte obsah a přínos této práce v českém jazyce.

Klíčová slova LTS odhad, lineÃąrnÃŋ regrese, optimalizace, nejmenÅąÃŋ usekanÃĺ ÄĎtvrece, metoda nejmenÅąÃŋch ÄĎtvercÅŕ, outliers

Abstract

The least trimmed squares (LTS) method is a robust version of the classical method of least squares used to find an estimate of coefficients in the linear regression model. Computing the LTS estimate is known to be NP-hard, and hence suboptimal probabilistic algorithms are used in practice.

Keywords LTS, linear regressin, robust estimator, least trimmed squares, ordinary least squares, outliers, outliers detection

Contents

List of Figures

Introduction

Goal of this thesis is to research currently used algorithms for calculation Least trimmed squares. Least trimmed squares was first introduced in 1984 and since that time couple of researchers came up couple of solutions. Beside that we'll propose extension of current algorithms which was never used before. We'll show that this updated algorithm is fast enough and also gives same or better results than currently used algorithms. We'll also compare speed and performance of this algorithms on various data sets both from literature and also using our custom data generator which provides very flexible way of generating data with outliers. We'll extend all those algorithms and implement all of them both in C++ an python. We'll implement python library with C++ bac end which will provide all of the currently used algorithms.

In first chapter we'll introduce linear regression and ordinary least squares method and it's downfalls. We'll introduce robust statistic and methods of evaluating robust models. In second chapters we'll analyze all of the algorithms together with is time complexity etc

Conclusion

There is still lot of future works on least trimmed squares. Proof of couple of thoughts is still about to come.

Introduction

Chapter 1

Least trimmed squares

In this chapter will be introduced one of the most common regression analysis models which is linear regression model. This model tries to model relationship between one variable which is considered to be dependent and one or more variables which are considered to be explanatory. Relationship is based on model function with parameters which are not known in advance and are estimated from data. We will also introduce one of the most commons methods of finding those parameters in this model, namely ordinary least squares.

1.1 Linear regression model

Definition 1. Linear regression model is

$$y_i = \boldsymbol{x_i}^T \boldsymbol{w} + \varepsilon_i, \quad i = 1, 2, \ldots, n \tag{1.1}$$

where $y_i \in \mathbb{R}$ is random variable which we denote as *dependent variable*, vector $\boldsymbol{x_i}^T = (x_1, x_2, \ldots, x_p)$ is column vector of *explanatory variables* and $\varepsilon \in \mathbb{R}$ random variable called *noise* or *error*. Vector $\boldsymbol{w} = (w_1, w_2, \ldots, w_p)$ is vector of parameters called *regression coefficients*. It is common to write whole model in a matrix form.

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\varepsilon} \tag{1.2}$$

where

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \boldsymbol{X} = \begin{bmatrix} \boldsymbol{x_1}^T \\ \boldsymbol{x_2}^T \\ \vdots \\ \boldsymbol{x_n}^T \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \ldots & x_{1p} \\ x_{21} & x_{22} & x_{23} & \ldots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \ldots & x_{np} \end{bmatrix}, \boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}, \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

We consider $\boldsymbol{x_{i1}} = 1$ to be a constant. Corresponding w_1 we call *intercept*. We refer this model as *model with intercept*. It is possible to have model

without intercept but then hyperplane generated by this model goes through origin. That is usually rare case so we always use intercept unless we explicitly mention it. Due to this we can assume that in model with intercept expected value is $\mathbb{E}(\varepsilon_i) = 0$.

1.1.1 Prediction with linear regression model

Linear regression model contains vector of real regression coefficients which we don't know and which we need to estimate in order to be able to use model for prediction. So let us assume that we already have estimated regression coefficients which we mark as $\hat{\boldsymbol{w}}$. Then we're able to predict value of y by

$$\hat{y} = \hat{\boldsymbol{w}}^T \boldsymbol{x} \tag{1.3}$$

$\hat{y}$ denotes that it is predicted value. Real value of y is given by

$$y = \boldsymbol{w}^T \boldsymbol{x} + \varepsilon \tag{1.4}$$

Because we assume linear dependence between dependent variable y and explanatory variables $\boldsymbol{x}$ then what makes y random variable is actually random variable ε. Because we use model with intercept thus with $\mathbb{E}(e) = 0$ we can see that

$$\mathbb{E}(y) = \mathbb{E}(\boldsymbol{x}^T \boldsymbol{w}) + \mathbb{E}(\varepsilon) = \mathbb{E}(\boldsymbol{x}^T \boldsymbol{w}) \tag{1.5}$$

so $\hat{y}$ is actually a point estimation of expected value of y.

1.2 Ordinary least squares

We want to estimate $\boldsymbol{w}$ so that error of the model will be minimal. Measurement of this error is most often done by *loss function*

$$L : \mathbb{R}^2 \to \mathbb{R} \tag{1.6}$$

Which in case of ordinary least squares is quadratic loss function $L(y, \hat{y}) := (y - \hat{y})^2$. We refer to $r(\hat{\boldsymbol{w}}) = y - \hat{y} = y - \hat{\boldsymbol{w}}^T \boldsymbol{x}$ as to residual.

So the idea of this method lies in fact that we want to minimize error given by sum of squared residuals commonly known as residual sum of squares RSS

$$RSS = \sum_{i=1}^{n} r_i^2 = (y_i - \boldsymbol{w}^T \boldsymbol{x_i})^2 \tag{1.7}$$

Definition 2. If we think about RSS as the function of $\boldsymbol{w}$ we'll get *objective function* for ordinary least squares.

$$\mathrm{OF}^{(OLS, \boldsymbol{X}, \boldsymbol{y})}(\boldsymbol{w}) = \sum_{i=1}^{n} (y_i - \boldsymbol{w}^T \boldsymbol{x_i})^2 \tag{1.8}$$

Where $\boldsymbol{w}$ has the meaning of function argument, not the real regression coefficients.

And minimum of this function we can denote as

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \mathrm{OF}^{(OLS,\boldsymbol{X},\boldsymbol{y})}(\boldsymbol{w}) \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} (y_i - \boldsymbol{w}^T \boldsymbol{x_i})^2 \tag{1.9}$$

which is basically a definition of ordinary least squares estimate.

Definition 3. Estimate of regression parameters using ordinary least squares (OLS) on n observations is defined as

$$\hat{\boldsymbol{w}}^{(\boldsymbol{OLS},\boldsymbol{n})} = \sum_{i=1}^{n} (y_i - \tilde{\boldsymbol{w}}^T \boldsymbol{x_i})^2 \tag{1.10}$$

Let's now talk about finding solution for this. If we want to find minimum of this function, first we need to find gradient by calculating all partial derivatives

$$\frac{\partial \mathrm{OF}^{(OLS,\boldsymbol{X},\boldsymbol{y})}}{\partial w_j} = \sum_{i=1}^{n} 2(y_i - \boldsymbol{w}^T \boldsymbol{x_i})(-x_{ij}), \quad j \in \{1, 2, \ldots, p\}. \tag{1.11}$$

By this we obtain gradient

$$\nabla \mathrm{OF}^{(OLS,\boldsymbol{X},\boldsymbol{y})} = -\sum_{i=1}^{n} 2(y_i - \boldsymbol{w}^T \boldsymbol{x_i})\boldsymbol{x_i}. \tag{1.12}$$

As a *normal equation* we mark gradient equal to zero.

$$-\sum_{i=1}^{n} 2(y_i - \boldsymbol{w}^T \boldsymbol{x_i})\boldsymbol{x_i} = 0 \tag{1.13}$$

Which we can write in matrix form as

$$\boldsymbol{X}^T \boldsymbol{y} - \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{w} = 0 \tag{1.14}$$

proof it is a local minimum

Let's now construct hessian matrix using second-order partial derivatives. With

$$\frac{\partial^2 \mathrm{OF}^{(OLS,\boldsymbol{X},\boldsymbol{y})}}{\partial w_h \partial w_j} = \sum_{i=1}^{n} 2(-x_{ik})(-x_{ij}), \quad h \in \{1, 2, \ldots, p\}. \tag{1.15}$$

we get

$$\boldsymbol{H}_{\mathbf{OF}^{(\boldsymbol{OLS},\boldsymbol{X},\boldsymbol{y})}} = 2\boldsymbol{X}^T \boldsymbol{X}. \tag{1.16}$$

We can see that hessian $\boldsymbol{H}_{\mathbf{OF}(\boldsymbol{OLS},\boldsymbol{X},\boldsymbol{y})}$ is always positive semi-definite because for all $\boldsymbol{s} \in \mathbb{R}^p$

$$\boldsymbol{s}^T(2\boldsymbol{X}^T\boldsymbol{X})\boldsymbol{s} = 2(\boldsymbol{X}\boldsymbol{s})^T(\boldsymbol{X}\boldsymbol{s}) = \tag{1.17}$$

It's easy to proof that twice differentiable function is convex if and only if the hessian of such function is positive semi-definite. Due to that we can say that solution of (1.14) gives us not only local minimum but global minimum.

If we assume that $\boldsymbol{X}^T\boldsymbol{X}$ is regular matrix, then its inverse exists and solution can be explicitly written as

$$\hat{\boldsymbol{w}}^{(OLS,n)} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{1.18}$$

where (OLS, n) denotes that $\hat{\boldsymbol{W}}$ is estimate of $\boldsymbol{w}$ calculated using ordinary least squares on n observations.

Moreover we can see that if $\boldsymbol{X}^T\boldsymbol{X}$ is regular matrix, then a hessian $\boldsymbol{H}_{\mathbf{OF}(\boldsymbol{OLS},\boldsymbol{X},\boldsymbol{y})}$ is positive definite because for all nonzero $\boldsymbol{s} \in \mathbb{R}^p$

$$\boldsymbol{s}^T 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{s} > 0 \tag{1.19}$$

thus $\hat{\boldsymbol{w}}^{(OLS,n)}$ in (1.18) is strict global minimum.

ols estimate is BLUE if we assume this and that

We've shown that $\hat{\boldsymbol{w}}^{(OLS,n)}$ can be computed directly by multiplying $\boldsymbol{A} = \boldsymbol{X}^T\boldsymbol{X}$ and $\boldsymbol{B} = \boldsymbol{X}\boldsymbol{y}$ and then by finding inversion of $\boldsymbol{A}$ and finally multiplying $\boldsymbol{A}^{-1}\boldsymbol{B}$. This is certainly possible but not the best solution because this method is not numerically stable. We'll describe this in detail in section 2.1

1.3 Robust statistics

To work properly standard statistic methods expects some assumptions and fail if those assumptions are not met. Robust statistic are statistics that produce acceptable results even if data are from some unconventional distributions or if data contains errors which are not normally distributed. We should be highly motivated to use such methods because in the real world we are often forced to work with such data. Classical statistics methods provide poor results with such data.

Ordinary least squares method has such assumptions about data. This method expects that

$$\varepsilon_i \sim \mathcal{N}(\mu, \sigma^2), \quad i \in \{1, 2, \ldots, n\}. \tag{1.20}$$

Condition of expected value to be zero is met, if we use model with intercept. But we can't be sure if errors are normally distributed. Moreover we expect that variance stays the same for all $\varepsilon_i, i \in \{1, 2, \ldots, n\}$. That is what we call *homoscedasticity*. Beside assumptions about ε we expect that

explanatory variables $x_1, x_2, \dots n_p$ are not linear dependent, or in terms of statistics, uncorrelated.

Before we explain what happens if those conditions are not met or met only partially let's describe one of the most common reason why assumptions are false.

1.3.1 Outliers

We stated a lot of assumptions that are required for ordinary least squares to give good estimate of $\hat{\boldsymbol{w}}$. Unfortunately in real conditions these assumptions are often false so that ordinary least squares don't produce acceptable results. One of the most common reasons of false assumptions are abnormal observations called outliers.

Outliers are very common and they are for instance erroneous measurements such as transmission errors or noise. Other very common reason is that nowadays we are mostly given data which were automatically processed by computers. Sometimes we are also presented data which are heterogenous in the sense that they contain data from multiple regression models. In certain sense outliers are inevitable. One would say that we would be able to eliminate them by precise examination, repair or removal of such data. That is possible in some cases, but most of the data we are dealing with are simply too big to check and more often we don't know how the data should look. In higher dimensions it is very difficult to find outliers. There are some methods that tries to find outliers but such methods are only partially efficient. Let us note that robust models are sometimes not only useful to create models that are not being unduly affected by presence of outliers but also capable of identifying data which seems to be outliers.

We have some terminology to describe certain types of outliers. We use terminology from [1]. Let's have observation $(y_i, \boldsymbol{x_i})$. If observation is not outlying in any direction we call it *regular observation* . If it is outlying in $\boldsymbol{x_i}$ direction we call it *leverage point*. We have two types of leverage points. If $\boldsymbol{x_i}$ is outlying but $(y_i, \boldsymbol{x_i})$ follows liner pattern we call it *good leverage point*. If it does not follow such a pattern we call it *bad leverage point*. Finally if $(y_i, \boldsymbol{x_i})$ is outlying only in y_i direction, we call it a *vertical outliers*.

1.3.2 Measuring robustness

There are couple of tools to measure robustness of statistics. The most popular one is called *breakdown point*. Then there are *empirical influence function* and *influence function and sensitivity curve*. For sake of simplicity we'll talk only about breakdown point right now.

Definition 4. Let T be a functional, $\boldsymbol{x} = (x_1, x_2, \dots, x_n), x_i \in \mathcal{X}$ be an n-dimensional random sample and $T_n(\boldsymbol{x})1$ value of this functional with parameter $\boldsymbol{x}$. The breakdown point of T at sample x can be defined using sample

$\boldsymbol{x_{new}}^{(k)}$ where we exchange k points from original sample $\boldsymbol{x}$ with random values x_i. We get $T_n(\boldsymbol{x_{new}}^{(k)})$. Then the *breakdown point* is

$$\text{bdpoint}(T, \boldsymbol{x_n}) = \frac{1}{n} \min S_{T,\boldsymbol{x_n}} \tag{1.21}$$

where

$$S_{T,\boldsymbol{x_n},D} = \{k \in \{1, 2, \ldots, n\} : \sup_{\boldsymbol{x_{new}}^{(k)}} \left\| T_n(\boldsymbol{x}), T_n(\boldsymbol{x_{new}}^{(k)}) \right\| = \infty\} \tag{1.22}$$

This definition is very general but let's specify it to our linear regression problem. It basically says that breakdown point is proportion of minimal number of observations needed to be switched for some others so then the estimator will give incorrect results. More robust estimators have higher breakdown point.

It is intuitive that breakdown point cannot be higher than 0.5 [2] because if we exchange more than 50% of the data, we wouldn't be able to distinguish between the original and exchanged data. Moreover the original data would become minority over exchanged.

In context of ordinary least squares estimator (OLS) it's easy to show that one outlier is enough to increase value of T to any desired value thus

$$\text{bdpoint}(OLS, \boldsymbol{x_n}) = \frac{1}{n} \,. \tag{1.23}$$

include some nice image showing 1/n robustness

For increasing number of data samples n this tends to zero. We can see that ordinary least squares estimator is not resistant to outliers at all. Due to this fact multiple estimators similar to OLS have been proposed.

1.4 Least trimmed squares

Least trimmed squares (LTS) estimator builds on OLS but is more robust. In this section we'll define LTS estimator and show that it's breakdown point is variable and can go upt to maximum possible value of breakdown point, thus 0.5.

Definition 5. Let's have $\boldsymbol{X} \in \mathbb{R}^{n,p}$, $\boldsymbol{y} \in \mathbb{R}^{n,1}$, $\boldsymbol{w} \in \mathbb{R}^p$ and h, $n/2 \leq h \leq n$. Objective function of LTS is then denoted as

$$\text{OF}^{(LTS,\boldsymbol{X},\boldsymbol{y})}(\boldsymbol{w}) = \sum_{i=1}^{h} r^2_{i:n}(\boldsymbol{w}) \tag{1.24}$$

Even though that objective function of LTS seems similar to the OLS objective function, finding minimum is much more complex because the smallest residuals are dependent on $\boldsymbol{w}$. This fact makes from finding LTS estimate non-convex optimization problem and finding global minimum is NP-hard.

reference

From this form of objective function we can hardly see how to optimize it, because we have h smallest residuals r_i which are dependent on vector of regression coefficients $\boldsymbol{w}$. Let's try to transform this objective function to another form and from which it'll be clear what we need to do in order to minimize this objective function.

Let's assume for now that we know $\hat{\boldsymbol{w}}^{\boldsymbol{LTS,h,n}}$ vector of regression coefficients. With this in mind create permutation of $\hat{n} = \{1, 2, \ldots, n\}$ such that

$$r_{i:n} = r_{\pi(j)}, \quad j \in \hat{n} \tag{1.25}$$

Next let's mark set

$$Q^{(n,h)} = \{\boldsymbol{m} \in \mathbb{R}^n, m_i \in \{0, 1\}, i \in \hat{n}, \mathbf{1}\boldsymbol{m} = h\} \tag{1.26}$$

which is simply set of all vectors $\boldsymbol{m} \in \mathbb{R}^n$ which contain h ones and $n - h$ zeros. We can now mark $\boldsymbol{m}_{LTS} \in Q^{(n,h)}$ such that $m_j^{(LTS)} = 1$ when $\pi(j) \leq h$ and $m_j^{(LTS)} = 0$ otherwise. Then

$$\hat{\boldsymbol{w}}^{\boldsymbol{(LTS,h,n)}} = \sum_{i=1}^{h} r_{i:n}^2(\boldsymbol{w}) = \sum_{i=1}^{n} m_i^{(LTS)} r_i^2(\boldsymbol{w}) \tag{1.27}$$

What that simply means that if we know vector $\boldsymbol{m}_{LTS}$ than we can simply compute LTS estimate as OLS estimate with $\boldsymbol{X}$ and $\boldsymbol{Y}$ multiplied by $\boldsymbol{M}_{LTS} = diag(\boldsymbol{m}_{LTS}^T)$. Thus

$$\hat{\boldsymbol{w}}^{\boldsymbol{(LTS,h,n)}} = (\boldsymbol{X}^T \boldsymbol{M}_{LTS}^T \boldsymbol{X} \boldsymbol{M}_{LTS})^{-1} \boldsymbol{X}^T \boldsymbol{M}_{LTS}^T \boldsymbol{M}_{LTS} \boldsymbol{y} \tag{1.28}$$

That means finding minimum of LTS objective function is equal to solving OLS over all vectors $\boldsymbol{m} \in Q^{(n,h)}$. Thus if we mark $\tilde{X} = \boldsymbol{M}\boldsymbol{X}$ and $\tilde{y} = \boldsymbol{M}\boldsymbol{y}$, then

$$\begin{aligned}
\hat{\boldsymbol{w}}^{\boldsymbol{(LTS,h,n)}} &= \sum_{i=1}^{n} m_i r_i^2(\boldsymbol{w}) \\
&= \min_{\boldsymbol{m} \in Q^{(n,h)}} \left(\arg\min_{\boldsymbol{w} \in \mathbb{R}^p} \mathrm{OF}^{(OLS, \boldsymbol{MX}, \boldsymbol{My})} \right) \\
&= \min_{\boldsymbol{m} \in Q^{(n,h)}} \left(\min_{\boldsymbol{w} \in \mathbb{R}^p} \|\boldsymbol{My} - \boldsymbol{MXw}\|^2 \right)
\end{aligned}$$

Finally we can minimize inner part by substituting $\boldsymbol{w}$ with 1.27. By that we get

$$\begin{aligned}
\hat{\boldsymbol{w}}^{\boldsymbol{(LTS,h,n)}} &= \min_{\boldsymbol{m} \in Q^{(n,h)}} \left(\left\| \boldsymbol{My} - \boldsymbol{MX}(\boldsymbol{X}^T \boldsymbol{M}^T \boldsymbol{MX})^{-1} \boldsymbol{X}^T \boldsymbol{M}^T \boldsymbol{My} \right\|^2 \right) \\
&= \min_{\boldsymbol{m} \in Q^{(n,h)}} \left(\left\| \boldsymbol{My} - \boldsymbol{MX}(\boldsymbol{X}^T \boldsymbol{MX})^{-1} \boldsymbol{X}^T \boldsymbol{My} \right\|^2 \right)
\end{aligned}$$

We can easily see, that we've got objective function with argument $\boldsymbol{m} \in Q^{(n,h)}$. This objective function we'll mark as

$$\mathrm{J}(\boldsymbol{m}) = \left\| \boldsymbol{M}\boldsymbol{y} - \boldsymbol{M}\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{M}\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{M}\boldsymbol{y} \right\|^2 . \tag{1.29}$$

We can also clearly see that solution to minimizing this OF could by done straightforwardly by iterating over the $Q^{(n,h)}$ set.

Unfortunately this set is very big namely $\binom{n}{h}$ so this approach will fail if the n will be greater that 30 or 40. To overcome this problem multiple algorithms was proposed. Majority of them are probabilistic algorithms but beside that some exact algorithms were proposed.

Before moving to the next chapter and start describing those algorithms lets point out some fact about the number of residuals h and how it actually makes least trimmed squares robust estimator. LTS reaches maximum breakdown point when $h = [(n/2] + [(p + 1)/2]$ (where [.] denotes largest integer function) and that is which 0.5. That means that up to 50% of the data can be outliers. In practice this number is usually lower and if upper bound of percentage of outliers is known, then h should be set to match this percentage.

Chapter 2

Algorithms

In previous chapter we've covered necessary theory needed to implement algorithms that are in this chapter. Let's quickly recap most important fact that we know so far.

With robust linear regression problem we assume that our model with intercept

$$y_i = \boldsymbol{w}^T \boldsymbol{x_i} + \varepsilon_i \tag{2.1}$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ is $i.i.d.$ random variable has some displacements in upmost half of the explanatory $\boldsymbol{x_i}$ or dependent y_i variables. Thus only some subset $\tilde{\boldsymbol{X}} = \boldsymbol{M}\boldsymbol{X}$ with corresponding $\tilde{\boldsymbol{y}} = \boldsymbol{y}$ where $\boldsymbol{M} = diag(\boldsymbol{m})$ and $\boldsymbol{m} \in Q^{(n,h)}$ can be perceived so that

$$\tilde{y}_i \sim \mathcal{N}(\boldsymbol{w}^T \tilde{\boldsymbol{x}}_{\boldsymbol{i}}, \sigma^2). \tag{2.2}$$

We'll denote this subset simply as $hsubset$ of data in order to simplify following text. We've also learnt that finding solution of LTS we need to find correct h subset and then calculate estimate of regression coefficients $\tilde{w}$. So to find exact solution of LTS we need to go through all h subsets, find the correct one and calculate ordinary least squares fit to get the regression coefficients. There are not much of other options because objective function is non-differentiable and non-convex with lots of local minima.

We also know that exhaustive approach will fail due exponential size of $Q^{(n,h)}$. So what are the possibilities? First attempts were based on iterative removal data samples whose residuum had the highest value based on OLS fit on whole dataset. Such attempts were proven to be completely wrong because initial OLS fit is usually already heavily affected by outliers and we my end up removing data samples which represents original model.

Then there are algorithms based on purely on random approach.

On such algorithm is Random solution algorithm [3] which is basically randomly selects L h subsets and subsequently compute OLS fit on each of them and selecting fit with smallest RSS and consider it a approximate solution.

Such approach is very simple, but in general probability of selecting at least one such subset from L subsets which don't contain outliers thus has a chance of producing good result goes to zero for increasing number of data samples n as we'll describe in detail in 15.

Another very similar algorithm called Resampling algorithm introduced in [4] have basically just a little difference and that it select vectors from $Q^{(n,p+1)}$ instead of $Q^{(n,h)}$. This minor tweak has not only higher chance to succeed because number of vectors in this set is significantly lower than in $Q^{(n,h)}$ (at least if h conservatively chosen thus $h = [(n/2] + [(p + 1)/2])$ but also because probability of selecting L subsets of size which is independent of n gives nonzero probability of selecting at least one subset such that it don't include outliers see 16 for more details.

Generating all possible h subsets is computationally hard and relying on selecting random subsets don't produce sufficiently good results. So what are our options? In [5] two criterions called *necessary conditions* are introduced. They talk about necessary properties which some h subset must satisfy so it could be set which leads to global optima of LTS. Let's introduce those two necessary conditions. For that it's convenient not to only label h subset of used observations but also complementary subset of not used observations. We'll refer to this complementary subset as g subset.

Theorem 6. Strong necessary condition. The criterion cannot be improved by exchanging any of the observations from g subset for any of the currently used observations in h subset. Thus $\boldsymbol{m} \in Q^{(n,h)}$ meets the criterion if $J(\boldsymbol{m}) \leq J(\boldsymbol{m_{swap}})$ where $\boldsymbol{m_{swap}}$ is any vector from $Q^{(n,h)}$ such that it has same values except one swapped.

Proof. Trivial. LTS uses subset of h observations that minimize it's objective function. To be this true none of the swaps between observations from h subset and g subset must not improve (reduce) it's objective function. □

Based on this idea algorithm can be created. We'll discuss it in detail in 2.4.

Second necessary condition named *weak necessary condition*

Theorem 7. $\boldsymbol{m} \in Q^{(n,h)}$ meets the criterion if for each observation from h subset has smaller (or equal) squared residual than any observation from g subset.

Again, based on this criterion an algorithm can be crated. Corollary of this criteria together with proof can be found in 2.3.

Very interesting consequence which we'll use later gives us following lemma.

Lemma 8. Strong necessary condition is not satisfied unless weak necessary condition in satisfied. Thus if strong condition is satisfied then weak is also.

Proof. We'll make proof by contradiction. Let's assume that we have $\boldsymbol{m} \in Q^{(n,h)}$ and $J(\boldsymbol{m})$ for which strong necessary condition is satisfied but weak

necessary condition is not. That means there exists $\boldsymbol{x_i}$ with y_i from h subset and $\boldsymbol{x_j}$ and y_j from g subset such that $r_j^2 < r_i^2$. Thus

$$\mathrm{J}(\boldsymbol{m}) > \mathrm{J}(\boldsymbol{m}) + r_j^2 - r_i^2 \tag{2.3}$$

Now we just need to show that $\boldsymbol{m_{swap}}$ vector that is created by swapping that jth observation from g subset with ith observation from h subset leads to

$$\mathrm{J}(\boldsymbol{m}) + r_j^2 - r_i^2 \geq \mathrm{J}(\boldsymbol{m_{swap}}). \tag{2.4}$$

That's indeed trivial because $\mathrm{J}(\boldsymbol{m_{swap}})$ is in fact just OLS hat minimize objective function on given subset of observations. Thats of course contradiction with our assumption which says that strong necessary condition is already satisfied. □

When we'll discuss algorithms based on this conditions we'll show that algorithm based on weak necessary condition is much faster than algorithm based on strong necessary condition which will lead us to another algorithm where we'll use 8.

Now we've covered all necessary theoretical background and it's time to introduce currently popular algorithms of computing LTS estimate.

2.1 Computing OLS

In this section we'll describe what algorithms of computing OLS exists. We'll see that beside computing OLS directly from the objective function there are better ways. Let's now start with this straightforward approach.

Lemma 9. Time complexity of OLS on $\boldsymbol{X}^{n\times p}$ and $\boldsymbol{Y}^{n\times 1}$ is $O(p^2 n)$.

Proof. Normal equation of OLS is $\hat{\boldsymbol{w}} = (\boldsymbol{X^T X})^{-1}\boldsymbol{X^T Y}$. Time complexity of matrix multiplication $\boldsymbol{A}^{m\times n}$ and $\boldsymbol{B}^{n\times p}$ is $\sim \mathcal{O}(mnp)$. Time complexity of matrix $\boldsymbol{C}^{m\times m}$ is $\sim \mathcal{O}(m^3)$ So we need to compute $\boldsymbol{A} = \boldsymbol{X^T X} \sim \mathcal{O}(p^2 n)$ and $\boldsymbol{B} = \boldsymbol{X^T Y} \sim \mathcal{O}(pn)$ and $\boldsymbol{C} = \boldsymbol{A}^{-1} \sim \mathcal{O}(p^3)$ and finally $\boldsymbol{CB} \sim \mathcal{O}(p^2)$. That gives us $\mathcal{O}(p^2 n + pn + p^3 + p^2)$. Because $\mathcal{O}(p^2 n)$ and $\mathcal{O}(p^3)$ asymptotically dominates over $\mathcal{O}(p^2)$ and $\mathcal{O}(pn)$ we can write $\mathcal{O}(p^2 n + p^3)$.

□

CO zo toho je vic? Neni casove narocnejsi vynasobeni $\boldsymbol{X^T X}$ nez inverze, kdyz bereme v uvahu $n >> p$???

complete this section with describing computation of OLS using matrix decomposition

2.2 Computing LTS

Note 10. When discussing following algorithms, we'll refer to given $\boldsymbol{X}$ and y as to ***data set*** and to y_i with corresponding $\boldsymbol{x_i}$ as to ***data sample*** or ***observation***. Sometimes it's also useful to refer to multiple observations as to subset of observations. When we want to mark subset of observations y_i , $\boldsymbol{x_i}$, $i \in H$, $H \subset \{1, 2, \ldots, n\}$ we can simply refer to it as to subset of observations H. Sometimes it's also useful to mark matrix $\boldsymbol{X}$ with only some subset of observations which we'll do by $\boldsymbol{X_H}$.

2.3 FAST-LTS

In this section we will introduce FAST-LTS algorithm[6]. It is, as well as in other cases, iterative algorithm. We will discuss all main components of the algorithm starting with its core idea called concentration step which ' authors simply call C-step.

2.3.1 C-step

We will show that from existing LTS estimate $\hat{\boldsymbol{w}}_{\boldsymbol{old}}$ we can construct new LTS estimate $\hat{\boldsymbol{w}}_{\boldsymbol{new}}$ which objective function is less or equal to the old one. Based on this property we will be able to create sequence of LTS estimates which will lead to better results.

Theorem 11. Consider dataset consisting of $\boldsymbol{x_1}, \boldsymbol{x_2} \ldots, \boldsymbol{x_n}$ explanatory variables where $\boldsymbol{x_i} \in \mathbb{R}^p, \forall \boldsymbol{x_i} = (f x_1^i, x_2^i, \ldots, x_p^i)$ where $x_1^i = 1$ and its corresponding $y_1, y_2, \ldots, y_n$ response variables. Let's also have $\hat{\boldsymbol{w}}_{\boldsymbol{0}} \in \mathbb{R}^p$ any p-dimensional vector and $H_0 = \{h_i; h_i \in \mathbb{Z}, 1 \leq h_i \leq n\}, |H_0| = h$. Let's now mark $RSS(\hat{\boldsymbol{w}}_{\boldsymbol{0}}) = \sum_{i \in H_0} (r_0(i))^2$ where $r_0(i) = y_i - (w_1^0 x_1^i + w_2^0 x_2^i + \ldots + w_p^0 x_p^i)$. Let's take $\hat{n} = \{1, 2, \ldots, n\}$ and mark $\pi : \hat{n} \to \hat{n}$ permutation of $\hat{n}$ such that $|r_0(\pi(1))| \leq |r_0(\pi(2))| \leq \ldots \leq |r_0(\pi(n))|$ and mark $H_1 = \{\pi(1), \pi(2), ...\pi(h)\}$ set of h indexes corresponding to h smallest absolute residuals $r_0(i)$. Finally take $\hat{\boldsymbol{w}}_{\boldsymbol{1}}^{\boldsymbol{OLS(H_1)}}$ ordinary least squares fit on H_1 subset of observations and its corresponding $RSS(\hat{\boldsymbol{w}}_{\boldsymbol{1}}) = \sum_{i \in H_1} (r_1(i))^2$ sum of least squares. Then

$$RSS(\hat{\boldsymbol{w}}_{\boldsymbol{1}}) \leq RSS(\hat{\boldsymbol{w}}_{\boldsymbol{0}}) \tag{2.5}$$

Proof. Because we take h observations with smallest absolute residuals r_0, then for sure $\sum_{i \in H_1} (r_0(i))^2 \leq \sum_{i \in H_0} (r_0(i))^2 = RSS(\hat{\boldsymbol{w}}_{\boldsymbol{0}})$. When we take into account that Ordinary least squares fit OLS_{H_1} minimize objective function of H_1 subset of observations, then for sure $RSS(\hat{\boldsymbol{w}}_{\boldsymbol{1}}) = \sum_{i \in H_1} (r_1(i))^2 \leq \sum_{i \in H_1} (r_0(i))^2$. Together we get

$$RSS(\hat{\boldsymbol{w}}_{\boldsymbol{1}}) = \sum_{i \in H_1} (r_1(i))^2 \leq \sum_{i \in H_1} (r_0(i))^2 \leq \sum_{i \in H_0} (r_0(i))^2 = RSS(\hat{\boldsymbol{w}}_{\boldsymbol{0}})$$

□

Corollary 12. Based on previous theorem, using some $\hat{\boldsymbol{w}}^{\boldsymbol{OLS(H_{old})}}$ on H_{old} subset of observations we can construct H_new subset with corresponding $\hat{\boldsymbol{w}}^{\boldsymbol{OLS(H_{new})}}$ such that $RSS(\hat{\boldsymbol{w}}^{\boldsymbol{OLS(H_{new})}}) \leq RSS(\hat{\boldsymbol{w}}^{\boldsymbol{OLS(H_{old})}})$. With this we can apply above theorem again on $\hat{\boldsymbol{w}}^{\boldsymbol{OLS(H_{new})}}$ with H_{new}. This will lead to the iterative sequence of $RSS(\hat{\boldsymbol{w}}_{\boldsymbol{old}}) \leq RSS(\hat{\boldsymbol{w}}_{\boldsymbol{new}}) \leq \ldots$. One step of this process is described by following pseudocode. Note that for C-step we actually need only $\hat{\boldsymbol{w}}$ without need of passing H.

include and reference nice image showing one c-step

Algorithm 1: C-step

Input: dataset consiting of $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ and $\boldsymbol{y} \in \mathbb{R}^{n \times 1}$, $\hat{\boldsymbol{w}}_{\boldsymbol{old}} \in \mathbb{R}^{p \times 1}$
Output: $\hat{\boldsymbol{w}}_{\boldsymbol{new}}$, H_{new}

```
R ← ∅;
for i ← 1 to n do
    R ← R ∪ {|y_i − ŵ_old x_i^T|};
end
H_new ← select set of h smallest absolute residuals from R;
ŵ_new ← OLS(H_new);
return ŵ_new , H_new;
```

Observation 13. Time complexity of algorithm C-step 1 is the same as time complexity as OLS. Thus $O(p^2 n)$

create better proof. And take into account both versions - directly vs. using decomposition

Proof. In C-step we must compute n absolute residuals. Computation of one absolute residual consists of matrix multiplication of shapes $1 \times p$ and $p \times 1$ that gives us $\mathcal{O}(p)$. Rest is in constant time. So time of computation n residuals is $\mathcal{O}(np)$. Next we must select set of h smallest residuals which can be done in $\mathcal{O}(n)$ using modification of algorithm QuickSelect. Finally we must compute $\hat{w}$ OLS estimate on h subset of data. Because h is linearly dependent on n, we can say that it is $\mathcal{O}(p^2 n + p^3)$ which is asymptotically dominant against previous steps which are $\mathcal{O}(np + n)$. □

reference or define quick select

As we stated above, repeating algorithm C-step will lead to sequence of $\hat{\boldsymbol{w}}_{\boldsymbol{1}}, \hat{\boldsymbol{w}}_{\boldsymbol{2}} \ldots$ on subsets $H_1, H_2 \ldots$ with corresponding residual sum of squares $RSS(\hat{\boldsymbol{w}}_{\boldsymbol{1}}) \geq RSS(\hat{\boldsymbol{w}}_{\boldsymbol{2}}) \geq \ldots$. One could ask if this sequence will converge, so that $RSS(\hat{\boldsymbol{w}}_{\boldsymbol{i}}) == RSS(\hat{\boldsymbol{w}}_{\boldsymbol{i+1}})$. Answer to this question will be presented by the following theorem.

Theorem 14. Sequence of C-step will converge to $\hat{\boldsymbol{w}}_{\boldsymbol{m}}$ after maximum of $m = \binom{n}{h}$ so that $RSS(\hat{\boldsymbol{w}}_{\boldsymbol{m}}) == RSS(\hat{\boldsymbol{w}}_{\boldsymbol{n}}), \forall n \geq m$ where n is number of data samples and h is size of subset H_i.

Proof. Since $RSS(\hat{\boldsymbol{w}}_{\boldsymbol{i}})$ is non-negative and $RSS(\hat{\boldsymbol{w}}_{\boldsymbol{i}}) \leq RSS(\hat{\boldsymbol{w}}_{\boldsymbol{i+i}})$ the sequence will converge. $\hat{\boldsymbol{w}}_{\boldsymbol{i}}$ is computed out of subset $H_i \subset \{1, 2, \ldots, n\}$. When

there is finite number of subsets of size h out of n samples, namely $\binom{n}{h}$, the sequence will converge at the latest after this number of steps. □

Above theorem gives us clue to create algorithm described by following pseudocode.

Algorithm 2: Repeat-C-step

Input: dataset consiting of $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ and $\boldsymbol{y} \in \mathbb{R}^{n \times 1}$, $\hat{\boldsymbol{w}}_{\boldsymbol{old}} \in \mathbb{R}^{p \times 1}$, H_0
Output: $\hat{\boldsymbol{w}}_{\boldsymbol{final}}$, H_{final}

```
ŵ_new ← ∅;
H_new ← ∅;
RSS_new ← ∞;
while True do
    RSS_old ← RSS(ŵ_old);
    ŵ_new , H_new ← X , y , ŵ_old;
    RSS_new ← RSS(ŵ_new);
    if RSS_old == RSS_new then
        break
    end
    ŵ_old ← ŵ_new
end
return ŵ_new, H_new;
```

include sime nice grap which show this. or table ?

It is important to note, that although maximum number of steps of this algorithm is $\binom{n}{h}$ in practice it is very low, most often under 20 steps. That is not enough for the algorithm $Repeat-C-step$ to converge to global minimum, but it is necessary condition. That gives us an idea how to create the final algorithm. [6]

Choose a lot of initial subsets H_1 and on each of them apply algorithm Repeat-C-step. From all converged subsets with corresponding $\hat{w}$ estimates choose that which has lowest $RSS(\hat{w})$.

Before we can construct final algorithm we must decide how to choose initial subset H_1 and how many of them mean "*a lot of*". First let's focus on how to choose initial subset H_1.

2.3.2 Choosing initial H_1subset

It is important to note, that when we choose H_1 subset such that it contains outliers, then iteration of C-steps usually won't converge to good results, so we should focus on methods with non zero probability of selecting H_1 such that it won't contain outliers. There are a lot of possibilities how to create initial hH_1 subset. Lets start with most trivial one.

2.3.2.1 Random selection

Most basic way of creating H_1 subset is simply to choose random $H_1 \subset \{1, 2, \ldots, n\}$. Following observation will show that it not the best way.

Observation 15. With increasing number of data samples, thus with increasing n, the probability of choosing among m random selections of $H_{1_1}, \ldots, H_{1_m}$ the probability of selecting at least one H_{1_i} such that its corresponding data samples does not contains outliers, goes to 0.

Proof. Consider dataset of n containing $\epsilon > 0$ relative amount of outliers. Let $h = (n + p + 1)/2$ and m is number of selections random $|H| = h$ subsets. Then

$$\begin{aligned} P(\textit{one random data sample not outliers}) &= (1 - \epsilon) \\ P(\textit{one subset without outliers}) &= (1 - \epsilon)^h \\ P(\textit{one subset with at least one outlier}) &= 1 - (1 - \epsilon)^h \\ P(\textit{m subsets with at least one outlier in each}) &= (1 - (1 - \epsilon)^h)^m \\ P(\textit{m subsets with at least one subset without outliers}) &= 1 - (1 - (1 - \epsilon)^h)^m \end{aligned}$$

Because $n \to \infty \Rightarrow (1 - \epsilon)^h \to 0 \Rightarrow 1 - (1 - \epsilon)^h \to 1 \Rightarrow (1 - (1 - \epsilon)^h)^m \to 1 \Rightarrow 1 - (1 - (1 - \epsilon)^h)^m \to 0$ □

That means that we should consider other options of selecting H_1 subset. Actually if we would like to continue with selecting some random subsets, previous observation gives us clue, that we should choose it independent of n. Authors of algorithm came with such solution and it goes as follows.

2.3.2.2 P-subset selection

Let's choose subset $J \subset \{1, 2, \ldots, n\}, |J| = p$. Next compute rank of matrix $\boldsymbol{X}_{J:}$. If $rank(\boldsymbol{X}_{J:}) < p$ add randomly selected rows to $\boldsymbol{X}_{J:}$ without repetition until $rank(\boldsymbol{X}_{J:}) = p$. Let's from now on suppose that $rank(\boldsymbol{X}_{J:}) = p$. Next let us mark $\hat{\boldsymbol{w}}_{\boldsymbol{0}} = OLS(J)$ and corresponding $(r_0(1)), (r_0(2)), \ldots, (r_0(n))$ residuals. Now mark $\hat{n} = \{1, 2, \ldots, n\}$ and let $\pi : \hat{n} \to \hat{n}$ be permutation of $\hat{n}$ such that $|r(\pi(1))| \leq |r(\pi(2))| \leq \ldots \leq |r(\pi(n))|$. Finally put $H_1 = \{\pi(1), \pi(2), ...\pi(h)\}$ set of h indexes corresponding to h smallest absolute residuals $r_0(i)$.

Observation 16. With increasing number of data samples, thus with increasing n, the probability of choosing among m random selections of $J_{1_1}, \ldots, J_{1_m}$ the probability of selecting at least one J_{1_i} such that its corresponding data samples does not contains outliers, goes toho

$$1 - (1 - (1 - \epsilon)^h)^m > 0$$

Proof. Similarly as in previous observation. □

write reference to section where this'll be

Note that there are other possibilities of choosing H_1 subset other than these presented in [6]. We'll properly discuss them in chapter

Last missing piece of the algorithm is determining number of m initial H_1 subsets, which will maximize probability to at least one of them will converge to good solution. Simply put, the more the better. So before we will answer this question properly, let's discuss some key observations about algorithm.

2.3.3 Speed-up of the algorithm

In this section we will describe important observations which will help us to formulate final algorithm. In two subsections we'll briefly describe how to optimize current algorithm.

2.3.3.1 Selective iteration

The most computationally demanding part of one C-step is computation of OLS on H_i subset and then calculation of n absolute residuals. How we stated above, convergence is usually achieved under 20 steps. So for fast algorithm run we would like to repeat C-step as little as possible and in the same time didn't loose performance of algorithm.

Due to that convergence of repeating C-step is very fast, it turns out, that we are able to distinguish between starts that will lead to good solutions and those who won't even after very little C-steps iterations. ¡based on empiric observation, we can distinguish good or bad solution already after two or three iterations of C-steps based on $RSS(\hat{\boldsymbol{w}}_3)$ or $RSS(\hat{\boldsymbol{w}}_4)$ respectively.

So even though authors don't specify size of m explicitly, they propose that after a few C-steps we can choose (say 10) best solutions among all H_1 starts and continue C-steps till convergence only on those best solutions. This process is called Selective iteration.

include some nice table

We can choose m with respect to observation 16. In ideal case we would like to have probability of existence at least one initial H_1 subset close to 1. As we see m is exponentially dependent on p and at the same time in practice we don't know percentage of outliers in dataset. So it is difficult to mention exact value. Specific values of m in respect to data size is visible in table . So we can say that with $p < 10$ choosing $m = 500$ is usually safe starting point.

2.3.3.2 Nested extension

C-step computation is usually very fast for small n. Problem starts with very high n say $n > 10^3$ because we need to compute OLS on H_i subset of size h which is dependent on n. And then calculate n absolute residuals.

Authors came up with solution they call Nested extension. We will describe it briefly now.

- If n is greater than limit l, we'll create subset of data samples $L\,, |L| = l$ and divide this subset into s disjunctive sets $P_1, P_2, \ldots, P_s\,, |P_i| = \frac{l}{s}\,, P_i \cap P_j = \emptyset\,, \bigcup_{i=1}^{s} P_i = L$.

- For every P_i we'll set number of starts $m_{P_i} = \frac{m}{l}$.

- Next in every P_i we'll create m_{P_i} number of initial $H_{P_{i_1}}$ subsets and iterate C-steps for two iterations.

- Then we'll choose 10 best results from each subsets and merge them together. We'll get family of sets F_{merged} containing 10 best $H_{P_{i_3}}$ subsets from each P_i.

- On each subset from F_{merged} family of subsets we'll again iterate 2 C-steps and then choose 10 best results.

- Finally we'll use these best 10 subsets and use them to iterate C-steps till convergence.

- As a result we'll choose best of those 10 converged results.

2.3.3.3 Putting all together

We've described all major parts of the algorithm FAST-LTS. One last thing we need to mention is that even though C-steps iteration usually converge under 20 steps it is appropriate to introduce two parameters *max_iteration* and *threshold* which will limit number of C-steps iterations in some rare cases when convergence is too slow. Parameter *max_iteration* denotes maximum number of iterations in final C-step iteration till convergence. Parameter *threshold* denotes stopping criterion such that $|RSS(\hat{\boldsymbol{w}}_{\boldsymbol{i}}) - RSS(\hat{\boldsymbol{w}}_{\boldsymbol{i+1}})| \leq$ *threshold* instead of $RSS_i == RSS_{i+1}$. When we put all together, we'll get

FAST-LTS algorithm which is described by following pseudocode.

Algorithm 3: FAST-LTS

Input: $\boldsymbol{X} \in \mathbb{R}^{n \times p}, \boldsymbol{y} \in \mathbb{R}^{n \times 1}, m, l, s, max\_iteration, threshold$
Output: $\hat{\boldsymbol{w}}_{\boldsymbol{final}}$, H_{final}

$\hat{\boldsymbol{w}}_{\boldsymbol{final}} \leftarrow \emptyset$;
$H_{final} \leftarrow \emptyset$;
$F_{best} \leftarrow \emptyset$;
if $n \geq l$ **then**
  $F_{merged} \leftarrow \emptyset$;
  for $i \leftarrow 0$ ***to*** s **do**
    $F_{selected} \leftarrow \emptyset$;
    for $j \leftarrow 0$ ***to*** $\frac{l}{s}$ **do**
      $F_{initial} \leftarrow Selective\ iteration(\frac{m}{l})$;
      for H_i ***in*** $F_{initial}$ **do**
        $H_i \leftarrow Iterate\ C\ step\ few\ times(H_i)$;
        $F_{selected} \leftarrow F_{selected} \cup \{H_i\}$;
      end
    end
    $F_{merged} \leftarrow F_{merged} \cup Select\ 10\ best\ subsets\ from\ F_{selected}$;
  end
  for H_i ***in*** F_{merged} **do**
    $H_i \leftarrow Iterate\ C\ step\ few\ times(H_i)$;
    $F_{best} \leftarrow F_{best} \cup \{H_i\}$;
  end
  $F_{best} \leftarrow Select\ 10\ best\ subsets\ from\ F_{best}$;
else
  $F_{initial} \leftarrow Selective\ iteration(m)$;
  $F_{best} \leftarrow Select\ 10\ best\ subsets\ from\ F_{initial}$;
end
$F_{final} \leftarrow \emptyset$;
$W_{final} \leftarrow \emptyset$;
for H_i ***in*** F_{best} **do**
  $H_i, \hat{\boldsymbol{w}}_{\boldsymbol{i}} \leftarrow$
   $Iterate\ C\ step\ till\ convergence(H_i, max\_iteration, threshold)$;
  $F_{final} \leftarrow F_{final} \cup \{H_i\}$;
  $W_{final} \leftarrow W_{final} \cup \{\hat{\boldsymbol{w}}_{\boldsymbol{i}}\}$;
end
$\hat{\boldsymbol{w}}_{\boldsymbol{final}}, H_{final} \leftarrow select\ what\ with\ best\ RSS(F_{final}, W_{final})$;
return $\hat{\boldsymbol{w}}_{\boldsymbol{final}}, H_{final}$;

2.4 Feasible solution

In this section we'll introduce feasible solution algorithm first introduced in [7]. It is based on strong necessary condition we've described at 6. The basic idea can be described as follows.

Let's consider that we have some $\boldsymbol{m} \in Q^{(n,h)}$. It'll be convenient when we'll mark in the following text $O_m = \{i \in \{1, 2, \ldots, n\}; w_i = 1\}$ and $Z_m = \{j \in \{1, 2, \ldots, n\}; w_j = 0\}$ thus sets of indexes of positions where is 0 respectively 1 in vector $\boldsymbol{m}$. We can think about it as indexes of observations in h set and g set respectively. Then we can mark $\boldsymbol{m}^{(i,j)}$ as a vector which is constructed by swap of its ith and jth element where $i \in O$ and $j \in Z$. Such vector correspond to vector $\boldsymbol{m_{swap}}$ which we marked at 6.

With this in mind we can mark let's mark

$$\Delta S_{i,j}^{(\boldsymbol{m})} = \mathrm{J}(\boldsymbol{m}^{(i,j)}) - \mathrm{J}(\boldsymbol{m}) \tag{2.6}$$

thus change of the LTS objective function by swapping one observation from h subset with another from g subset. To calculate this we can obviously first calculate $\mathrm{J}(\boldsymbol{m})$ and $\mathrm{J}(\boldsymbol{m}^{(i,j)})$ and finally subtract both results. Although it is a option, it it computationally hard. So the question is if there is easier way of calculating $\Delta S_{i,j}^{(\boldsymbol{m})}$ and the answer is positive.

Let's mark $M = diag(\boldsymbol{m}$ and $M^(i,j) = diag(\boldsymbol{m}^{(i,j)})$ and also $\boldsymbol{H} = (\boldsymbol{X}^T\boldsymbol{M}^T\boldsymbol{X})$ For now let's assume that we already calculated $\boldsymbol{H}^{-1}$ and also $\hat{\boldsymbol{w}} = \boldsymbol{H}^{-1}\boldsymbol{X}^T\boldsymbol{M}\boldsymbol{y}$ we now want to calculate $\Delta S_{i,j}^{(\boldsymbol{m})}$ Let's mark vector of residuals $\boldsymbol{r}^{(m)} = \boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{w}}$ and also $d_{r,s} = \boldsymbol{x_r}\boldsymbol{H}^{-1}\boldsymbol{x_s}$ then by equation from [7] we get

$$\Delta S_{i,j}^{(\boldsymbol{m})} = \frac{(r_j^{(m)})^2(1 - d_{i,i}) - (r_i^{(m)})^2(1 + d_{j,j}) + 2r_i^{(m)}r_j^{(m)}d_{j,j}}{(1 - d_{i,i})(1 + d_{j,j}) + d_{i,j}^2}. \tag{2.7}$$

Let's now describe core of the algorithm. It's a similar to the FAST-LTS algorithm in the sense of iterative refinement of h subset. So let's assume that we have some vector $\boldsymbol{m} \in Q^{(n,h)}$. No we'll compute $\Delta S_{i,j}^{(\boldsymbol{m})}$ for all $i \in O$ and $j \in Z$. This may lead to several different outcomes

1. all $S_{i,j}^{(\boldsymbol{m})}$ are non-negative
2. one $S_{i,j}^{(\boldsymbol{m})}$ is positive
3. multiple $S_{i,j}^{(\boldsymbol{m})}$ are positive

In the first case, all $\mathrm{J}(\boldsymbol{m}^{(i,j)})$ are higher than $\mathrm{J}(\boldsymbol{m})$ so none swap will lead to an improvement. That also means that strong necessary condition is satisfied and the algorithm ends.

In the second and third case strong necessary condition is not satisfied and we make the swap. In second case it's easy which one to choose, because we have only one. in the third case we have couple of options again.

1. use the first swap that leads to the improvement (so don't event try to find different swap)
2. from all possible swaps choose one that has highest improvement value $\mathrm{J}(\boldsymbol{m}^{(i,j)})$
3. use the first swap that has improvement higher than some given threshold

We'll discuss those options later and let's now assume that we'll use case number two. So if there positive $S_{i,j}^{(\boldsymbol{m})}$ we'll make the swap and repeat the process again. Algorithm ends when there is no possible improvement i.e. when all $S_{i,j}^{(\boldsymbol{m})}$ are non-negative. Number of iterations needed till algorithm stops is usually quite low, but for practical usage it's still convenient to use some parameter $max\_iteration$ after which algorithm will stop without finding h subset satisfying strong necessary condition. there is a decision to be made in considering the swaps âĂŞ should one 1. Accept the first swap that leads to an improvement in the criterion, 2. Search for the swap that leads to the greatest improvement, or 3. Search for a while, stopping at the first subset has been found that gives at least some minimum threshold of improvement. In complexity terms, there is no difference between these three approaches (since in all of them to establish that a trial solution is feasible you need to evaluate all possible swaps), but in practical terms 3 is a clear winner since it leads to many fewer inner iterations than 1, and mostly much faster inner iterations than 2. However the overall complexity of O(n2p) or O(n3p) remains, and means that the feasible solution algorithms based on the case-swap necessary condition cannot be used for very large data sets.

put sem part of the pseudocode describing core iteration?

Observation 17. Time complexity of algorithm C-step 1 is the same as time complexity as OLS. Thus $O(p^2 n)$

create better proof. And take into account both versions - directly vs. using decomposition

reference or define quick select

Proof. In C-step we must compute n absolute residuals. Computation of one absolute residual consists of matrix multiplication of shapes $1 \times p$ and $p \times 1$ that gives us $\mathcal{O}(p)$. Rest is in constant time. So time of computation n residuals is $\mathcal{O}(np)$. Next we must select set of h smallest residuals which can be done in $\mathcal{O}(n)$ using modification of algorithm QuickSelect. Finally we must compute $\hat{w}$ OLS estimate on h subset of data. Because h is linearly dependent on n, we can say that it is $\mathcal{O}(p^2 n + p^3)$ which is asymptotically dominant against previous steps which are $\mathcal{O}(np + n)$. □

As we stated above, repeating algorithm C-step will lead to sequence of $\hat{\boldsymbol{w}}_{\mathbf{1}}, \hat{\boldsymbol{w}}_{\mathbf{2}} \ldots$ on subsets $H_1, H_2 \ldots$ with corresponding residual sum of squares

$RSS(\hat{\boldsymbol{w}}_1) \geq RSS(\hat{\boldsymbol{w}}_2) \geq \ldots$. One could ask if this sequence will converge, so that $RSS(\hat{\boldsymbol{w}}_i) == RSS(\hat{\boldsymbol{w}}_{i+1})$. Answer to this question will be presented by the following theorem.

Theorem 18. Sequence of C-step will converge to $\hat{\boldsymbol{w}}_m$ after maximum of $m = \binom{n}{h}$ so that $RSS(\hat{\boldsymbol{w}}_m) == RSS(\hat{\boldsymbol{w}}_n), \forall n \geq m$ where n is number of data samples and h is size of subset H_i.

Proof. Since $RSS(\hat{\boldsymbol{w}}_i)$ is non-negative and $RSS(\hat{\boldsymbol{w}}_i) \leq RSS(\hat{\boldsymbol{w}}_{i+i})$ the sequence will converge. $\hat{\boldsymbol{w}}_i$ is computed out of subset $H_i \subset \{1, 2, \ldots, n\}$. When there is finite number of subsets of size h out of n samples, namely $\binom{n}{h}$, the sequence will converge at the latest after this number of steps. □

One run of this iteration process will lead to some local optima i.e. set satisfying strong necessary condition. In [7] they refer to to this set as to *feasible set*. This because it is not global optima the algorithm needs to be run multiple times say N times. As a final solution is taken such h subset with the smallest residual sum of squares.

REWRITE UP FROM THIS TO MATCH FEASIBLE SOLUTION

We didn't yet mention how to create initial h subset respectively initial $\boldsymbol{m}$. We already had this discussion when describing FAST-LTS algorithm. The [7] describes only random starting h subsets, but using p subsets instead may lead to improvement . More importantly as we already suggested h subset satisfying weak necessary condition don't need to satisfy strong necessary condition so passing such a h subset as input to this algorithm is another option and we'll discuss it in detail later. For that reasons we'll now describe feasible algorithm with pseudocode and we'll assume that we already have

experiment with this and reffer here

some function that generates for us h subsets e.g. random one.

Algorithm 4: Feasible solution

Input: $\boldsymbol{X} \in \mathbb{R}^{n \times p}, \boldsymbol{y} \in \mathbb{R}^{n \times 1}, max\_iteration, N$
Output: $\hat{\boldsymbol{w}}_{\boldsymbol{final}}$, H_{final}

$\hat{\boldsymbol{w}}_{\boldsymbol{final}} \leftarrow \emptyset$;
$H_{final} \leftarrow \emptyset$;
$Results \leftarrow \emptyset$;
for $k \leftarrow 0$ ***to*** N **do**
  $\boldsymbol{m} \leftarrow generate\_intial\_subset()$; // e.g. random $\boldsymbol{m} \in Q^{(n,h)}$
  while $True$ **do**
    $best_i \leftarrow 0$;
    $best_j \leftarrow 0$;
    $best_{delta} \leftarrow 0$;
    for $i \in O_m$ **do**
      for $j \in Z_m$ **do**
        $delta \leftarrow calculate\Delta S_{i,j}^{(\boldsymbol{m})}$;
        if $delta > best_{delta}$ **then**
          $best_{delta} \leftarrow delta$;
          $best_i \leftarrow i$;
          $best_j \leftarrow j$;
        end
      end
    end
    if $best_{delta} > 0$ **then**
      $\boldsymbol{m} \leftarrow \boldsymbol{m}^{(i,j)}$;
    end
    else
      $H \leftarrow h$ *subset corresponding to* $\boldsymbol{m}$;
      $\boldsymbol{M} \leftarrow diag(\boldsymbol{m})$;
      $\hat{w} \leftarrow \mathrm{OF}^{(OLS, \boldsymbol{MX}, \boldsymbol{My})}$;
      $Results \leftarrow Results \cup \{H, \hat{w}\}$;
      break;
    end
  end
end
$H_{final}, \hat{\boldsymbol{w}}_{\boldsymbol{final}} \leftarrow$ select best from $Results$ based on smallest RSS;
return $\hat{\boldsymbol{w}}_{\boldsymbol{final}}$, H_{final};

2.5 Combined algorithm

Here we'll describe what we've indicated above and that is combination of both previous algorithms. Two options. z

1. Let fast-lts converge and then run FSA and let it converge
2. Let fast-lts converge make one step of FSA and try fast-LTS again and iterate between this

Second option would be faster ? Let's think about it.. etc etc..

2.6 Improved FSA

2.6.1 Bounding-FSA aka. Modified Optimum exchange (MOEA)

2.6.2 Min-Max-FSA aka. Min Max exchange (MMEA)

make CHAPTER approximate ALGORITHMS AND chapter EXACT ALGORITHMS?

Chapter 3

Exact algorithms

Chapter 4

Experiments

4.1 Data

4.2 Results

4.3 Outlier detection

OF(ols, x,y)(w) $\text{OF}^{(OLS,\boldsymbol{X},\boldsymbol{y})}(\boldsymbol{w})$
w hat (OLS, n) $\hat{\boldsymbol{w}}^{(\boldsymbol{OLS},\boldsymbol{n})}$
OF (lts, x, y) (w) ¡- NOT USED $\text{OF}^{(LTS,\boldsymbol{X},\boldsymbol{y})}(\boldsymbol{w})$
OF lts J(w) $\text{J}(\boldsymbol{m})$
w hat (LTS, h, n) $\hat{\boldsymbol{w}}^{(\boldsymbol{LTS},\boldsymbol{h},\boldsymbol{n})}$
VEC m $\boldsymbol{m}_{LTS} \in Q^{(n,h)}$ $\boldsymbol{m} \in Q^{(n,h)}$ O Z
M = diag(m) MX MY $\text{OF}^{(OLS,\boldsymbol{MX},\boldsymbol{My})}$
h subset g subset
J J
RSS $RSS = \sum_{i=1}^{n} r_i^2 = (y_i - \boldsymbol{w}^T \boldsymbol{x_i})^2$

Conclusion

Bibliography

[1] Rousseeuw, P.; C. van Zomeren, B. Unmasking Multivariate Outliers and Leverage Points. *Journal of The American Statistical Association - J AMER STATIST ASSN*, volume 85, 06 1990: pp. 633–639, doi: 10.1080/01621459.1990.10474920.

[2] Massart, D. L.; Kaufman, L.; et al. Least median of squares: a robust method for outlier and model error detection in regression and calibration. *Analytica Chimica Acta*, volume 187, 1986: pp. 171–179.

[3] Bai, E.-W. A random least-trimmed-squares identification algorithm. *Automatica*, volume 39, no. 9, 2003: pp. 1651–1659.

[4] Rousseeuw, P. J.; Leroy, A. M. *Robust regression and outlier detection*. John wiley & sons, 1987.

[5] Hawkins, D. M.; Olive, D. J. Improved feasible solution algorithms for high breakdown estimation. *Computational statistics & data analysis*, volume 30, no. 1, 1999: pp. 1–11.

[6] Rousseeuw, P. J.; Driessen, K. V. An Algorithm for Positive-Breakdown Regression Based on Concentration Steps. In *Data Analysis: Scientific Modeling and Practical Application*, edited by M. S. W. Gaul, O. Opitz, Springer-Verlag Berlin Heidelberg, 2000, pp. 335–346.

[7] Hawkins, D. M. The feasible solution algorithm for least trimmed squares regression. *Computational statistics & data analysis*, volume 17, no. 2, 1994: pp. 185–196.

Appendix A

Datasets

GUI Graphical user interface

XML Extensible markup language

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `exe` ..... the directory with executables
- `src` ..... the directory of source codes
  - `wbdcm` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format
  - `thesis.ps` ..... the thesis text in PS format