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 8, 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 8, 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

Introduction **1**

1 Linear Regression **3**
1.1 Description . . . 3
1.2 Computation . . . 3
1.3 Downfalls . . . 3

2 The Least trimmed squares **5**

3 Algorithms **7**
3.1 FAST-LTS . . . 7
3.2 Exact algorithm . . . 14
3.3 Feasible solution . . . 14
3.4 MMEA . . . 14
3.5 Branch and bound . . . 14
3.6 Adding row . . . 14

4 Experiments **15**
4.1 Data . . . 15
4.2 Results . . . 15
4.3 Outlier detection . . . 15

Conclusion **17**

Bibliography **19**

A Datasets **21**

B Contents of enclosed CD **23**

List of Figures

Introduction

Chapter 1

Linear Regression

1.1 Description

1.2 Computation

1.3 Downfalls

Chapter 2

The Least trimmed squares

2.0.1 Objective function

2.0.1.1 Problems

Chapter 3

Algorithms

3.1 FAST-LTS

In this section we will introduce FAST-LTS algorithm[1]. 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.

3.1.1 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 1. Consider dataset consisting of $\boldsymbol{x_1}, \boldsymbol{x_2} \ldots, \boldsymbol{x_n}$ explanatory variables where $\boldsymbol{x_i} \in \mathbb{R}^p, \forall \boldsymbol{x_i} = (x_1^i, x_2^i, \ldots, x_p^i)$ 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

$$RSS(\hat{\boldsymbol{w}}_{\boldsymbol{1}}) \leq RSS(\hat{\boldsymbol{w}}_{\boldsymbol{0}}) \tag{3.1}$$

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

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

□

Corollary 2. 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.

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

Lemma 4. 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 $\hat{\boldsymbol{w}} = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{Y}$. 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 $\boldsymbol{A} = \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X} \sim \mathcal{O}(p^2 n)$ and $\boldsymbol{B} = \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{Y} \sim \mathcal{O}(pn)$ 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 $\mathcal{O}(p^2 n + p^3)$.

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

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. reference: ***TODO*** 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 $\mathcal{O}(p^2n + p^3)$ 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 $\hat{\boldsymbol{w}}_{\boldsymbol{1}}, \hat{\boldsymbol{w}}_{\boldsymbol{2}} \ldots$ 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 5. 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}

$\hat{\boldsymbol{w}}_{\boldsymbol{new}} \leftarrow \emptyset$;
$H_{new} \leftarrow \emptyset$;
$RSS_{new} \leftarrow \infty$;
while $True$ **do**
  $RSS_{old} \leftarrow RSS(\hat{\boldsymbol{w}}_{\boldsymbol{old}})$;
  $\hat{\boldsymbol{w}}_{\boldsymbol{new}}, H_{new} \leftarrow \boldsymbol{X}, \boldsymbol{y}, \hat{\boldsymbol{w}}_{\boldsymbol{old}}$;
  $RSS_{new} \leftarrow RSS(\hat{\boldsymbol{w}}_{\boldsymbol{new}})$;
  if $RSS_{old} == RSS_{new}$ **then**
    break
  end
  $\hat{\boldsymbol{w}}_{\boldsymbol{old}} \leftarrow \hat{\boldsymbol{w}}_{\boldsymbol{new}}$
end
return $\hat{\boldsymbol{w}}_{\boldsymbol{new}}$, H_{new};

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. ***TODO*** nejaky hezky grafik ktery to ukazuje.... 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. [1]

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.

3.1.2 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.

3.1.2.1 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 6. 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.

3.1.2.2 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 $\hat{\boldsymbol{w}}_{\mathbf{0}} = OLS(J)$ 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 7. 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. □

*Note that there are other possibilities of choosing H_1 subset other than these presented in [1]. We'll properly discuss them in chapter **TODO**.*

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.

3.1.3 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.

3.1.3.1 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 $RSS(\hat{\boldsymbol{w}}_{\mathbf{3}})$ or $RSS(\hat{\boldsymbol{w}}_{\mathbf{4}})$ 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.

We can choose m with respect to observation 7. 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 ***TODO****. So we can say that with $p < 10$ choosing $m = 500$ is usually safe starting point.*

3.1.3.2 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.

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

3.2 Exact algorithm

3.3 Feasible solution

3.4 MMEA

3.5 Branch and bound

3.6 Adding row

Chapter 4

Experiments

4.1 Data

4.2 Results

4.3 Outlier detection

Conclusion

Bibliography

[1] 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.

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