TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Řešení optimalizační úlohy LASSO pomocí proximálních algoritmů

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Václav Langr**
Vedoucí práce: doc. Ing. Zbyněk Koldovský, Ph.D.

Liberec 2016

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Solution to LASSO Using Proximal Algorithms

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **Václav Langr**
Supervisor: doc. Ing. Zbyněk Koldovský, Ph.D.

Liberec 2016

Tento list nahraďte
originálem zadání.

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Chtěl bych tímto poděkovat všem, kteří mne podporovali. Především mé rodině za velikou podporu, díky které mi bylo umožněno studovat.
Zároveň bych chtěl poděkovat také vedoucímu této práce doc. Ing. Zbyňku Koldovskému, Ph.D. za velice přínosné konzultace, rady a trpělivost.

Abstrakt

Tato bakalářská práce je zaměřena na rekonstrukci řídkého vektoru z jeho komprimovaného pozorování. Pro rekonstrukci se využívá optimalizačního problému LASSO a jeho řešení pomocí proximálních algoritmů. Po vytvoření takového algoritmu, který je schopen původní signál rekonstruovat, se využívá metody Monte Carlo pro pozorování závislosti chyby řešení na parametru lambda. Pro takto získaný výpočet je zjištěna kvadratická chyba řešení LASSO od původního vektoru dat, která je následně porovnávána s teoretickou chybou.

Vypracování této bakalářské práce bylo rozděleno do několika navazujících částí. Prvním a také nejdůležitějším krokem bylo nastudování vlastností proximálních algoritmů a výpočet proximálního operátora při různých vstupních funkcích. Po takto provedené rešerši proximálních algoritmů proběhla také rešerše vlastností optimalizační úlohy LASSO a jejích variant. Následně bylo možné přistoupit k implementaci algoritmu v programovacím jazyce a vývojovém prostředí MATLAB. Při postupné implementaci byl algoritmus upraven tak, aby vždy dokonvergoval ke správnému nebo alespoň přibližnému řešení optimalizačního problému. Z tohoto důvodu byl algoritmus rozšířen o podmínky optimality, jež ukončují výpočet při dosažení poměrně přesné aproximace. Dále byl rozšířen o výpočet dynamické velikosti kroku. S takto připraveným algoritmem mohla být vytvořena metoda Monte Carlo, která generuje nekomprimovaný řídký vektor dat, měřící matice, jejichž prvky mají Gaussovo rozložení, a parametr lambda v zadaném rozsahu s logaritmickým rozdělením.

Výsledek této práce může být využit pro další zpracování. Jedním z možných použití je např. pro vytvoření nového datového formátu, ve kterém by byl uložen jen komprimovaný vektor dat případně i měřící matice, pokud by nebyla shodná pro všechny komprimace. Na straně klienta by byl tedy pouze zrekonstruován původní nekomprimovaný signál.

Klíčová slova: MATLAB, proximální algoritmus, proximální operátor, LASSO, Monte Carlo

Abstract

This bachelor thesis is focused on reconstruction of sparse vector from his compressed observation. For the reconstruction is used the LASSO problem and its solution using proximal algorithms. After creation of an algorithm that is able to restore the original signal is used Monte Carlo method for analyzing dependence of computation error on lambda parameter. Then is calculated the squared error for the found solution and the original data that is compared with the theoretical error.

Realization of this bachelor thesis was divided into several parts. The very first and the most important step was studying the properties of proximal algorithms and evaluation of proximal operator for different functions. After the research on proximal algorithms there was also research on the properties of the LASSO and its variants. After that is was possible to implement algorithm using MATLAB language and development environment. The algorithm was modified during the implementation so it always converges into correct or at least approximate solution of LASSO. Due to this reason optimality conditions were added that terminates solving if the approximation is very accurate and a computation of dynamical step size. This prepared algorithm could be used for creation of Monte Carlo method that generates uncompressed sparse vector of data, random measurement matrix with Gaussian distribution and a lambda parameter within an interval with logarithmic distribution.

The result of this thesis can be used for another usage. One of the possible usage is for example creation of a new data format. The compressed data would be saved in this format and only if the measurement matrix is not the same for all compressions it would be also saved. On the client side the original uncompressed signal would be recovered.

Keywords: MATLAB, Proximal Algorithm, Proximal Operator, LASSO, Monte Carlo

Obsah

Seznam obrázků

Seznam vzorců

1 Úvod

V dnešním světě, kdy získáváme stále více důležitých a velkých dat je stále důležitější nějakým způsobem získaná data komprimovat. Z tohoto důvodu se tato práce zabývá komprimací dat a následnou rekonstrukcí u koncového uživatele. Bude pro to použito nového přístupu pomocí proximálních algoritmů, které jsou rychlé a jednoduché na implementaci.

Základní měření odpovídá funkci, viz Vzorec 1, kde jsou původní data $x_0 \in \mathbb{C}^n$ transformovány měřící maticí $A \in \mathbb{C}^{m \times n}$, kde m je počet řádků a n počet sloupců matice a v poslední řadě je nutné uvažovat i náhodná data $z \in \mathbb{C}^m$, v našem případě toto reprezentuje náhodný šum. Takto jsou získaná nová data $y \in \mathbb{C}^m$. V obecných případech platí, že $m \geq n$, ale takto nedochází k žádné komprimaci dat případně by i vzrostl počet dat. Následná rekonstrukce by se tak redukovala pouze na spočtení inverzní, v případě obdélníkové matice pseudoinverzní, matice A^{-1} a spočtení původních dat x_0. Tato bakalářská práce se zabývá případy, kdy $m < n$ a nelze tedy použít pseudoinverzní matici.

$$y = A \times x_0 + z \tag{Vzorec 1}$$

Dle pravidel lineární algebry tak existuje nekonečně mnoho řešení a je tak nemožné v tomto případě zrekonstruovat původní data. Nicméně ve skutečnosti lze data zrekonstruovat a to za podmínky, že původní vektor dat x_0 je řídký, tedy většina prvků z x_0 je rovno 0. Právě z tohoto důvodu jsou kompresní algoritmy, například pro kompresi formátu JPEG, kdy se ukládají pouze největší koeficienty diskrétní kosinové transformace, tak efektivní. Toto ale není jedinná podmínka, nutná k úspěšné rekonstrukci. Další podmínkou je správně zvolená měřící matice A. Například, při volbě diagonální jednotkové matice, by komprimovaná data y byla z velké části nulová. Z tohoto důvodu by pak bylo nemožné rekonstruovat původní data. Jelikož je toto téma velice zkoumané v dnešní době, vzniklo mnoho výzkumů, které se problematikou tohoto problému zabývají velice podrobně. Právě tyto výzkumy ukazují, že aby došlo k jisté rekonstrukci, je potřeba využít měřící matice o náhodných prvcích z normovaného normálního rozdělení, tj. normální rozdělení s nulovou střední hodnoutou a jednotkovým rozptylem.

2 Optimalizační problémy

Problém rekonstrukce původních dat z komprimovaného pozorování se vyskytuje v nejrůznějších oborech. Mezi tyto obory patří strojové učení, zpracování signálů a další. Jeden z možných přístupů je využít tzv. optimalizační problémy. Optimalizační problém je takový problém, kdy hledáme nejlepší možné řešení x ze všech možných a to tak, aby $f(x)$ bylo maximální nebo minimální, podle zadané úlohy. Hledáme tedy globální maximum, respektive minimum, funkce, kterou se snažíme vyřešit.

Veškeré další řešení je založeno na tom, že je lze zapsat v neomezeném tvaru. V tomto tvaru se minimalizuje obecný vstup x přes součet m konvexních funkcích $f_0 \dots f_m$, které náleží množině $\mathbb{R}^n$. Takto zapsaný problém by byl definován jako:

$$\min_{x \in \mathbb{R}^n} f_0(x) + \ldots + f_m(x) \qquad \text{(Vzorec 2)}$$

Nicméně, jelikož lze uvažovat jakoukoliv výše zmíněnou funkci, naskýtá se tu typický problém. Některé funkce nejsou derivovatelné. Z tohoto důvodu se nadále bude na funkce a jejich řešení nahlížet jako na samostatný problém.

2.1 Optimalizační problém LASSO

Jednou z variant jak řešit zadání této bakalářské práce je využít minimalizace a to tak, že se využije neomezeného tvaru optimalizačního problému pouze s funkcemi f_0 a f_1. Jedna z těchto funkcí se nahradí konkrétním předpisem a vznikne tak Vzorec 3. Funkce $f_1(x)$ ve vzorci je libovolná a volí se podle toho, jaké vlastnosti mají vstupní data.

$$\underset{x}{\operatorname{argmin}} \left\{ \|y - A \times x\|_2^2 + \lambda \times f_1(x) \right\} \tag{Vzorec 3}$$

V případě řídkého vektoru se volí funkce $\|x\|_1$ a vzniká tak vztah popsaný níže. Tento nový vztah zároveň řeší dva problémy a to rekonstrukci původních dat x_0 a jejich odšumnění.

$$\underset{x}{\operatorname{argmin}} \left\{ \|y - A \times x\|_2^2 + \lambda \times \|x\|_1 \right\} \tag{Vzorec 4}$$

Jelikož obě funkce optimalizačního problému LASSO obsahují funkci l normy, musí být zmíněna i její definice, která je popsána Vzorec 5. V případě $l1$ normy tak získáme pouze sumu absolutních hodnot a vzniká tak problém, jak bylo naznačeno v kapitole 2. Jak je patrné z definice absolutní hodnoty, lze sestrojit nekonečně mnoho tečen v bodě se souřadnicemi $[0, 0]$ a nelze tak spočítat její derivaci, což je velice důležité pro pozdější zpracování.

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \tag{Vzorec 5}$$

3 Proximální algoritmy

Jednou z možností, jak vyřešit problém zmíněný v předchozí kapitole, je využít proximální algoritmy a proximální operátory příslušející zadaným funkcím. Proximimální algoritmus je něco, co lze zapsat vztahem:

$$x_{n+1} = prox_{\lambda \times f}(x_n) \qquad \text{(Vzorec 6)}$$

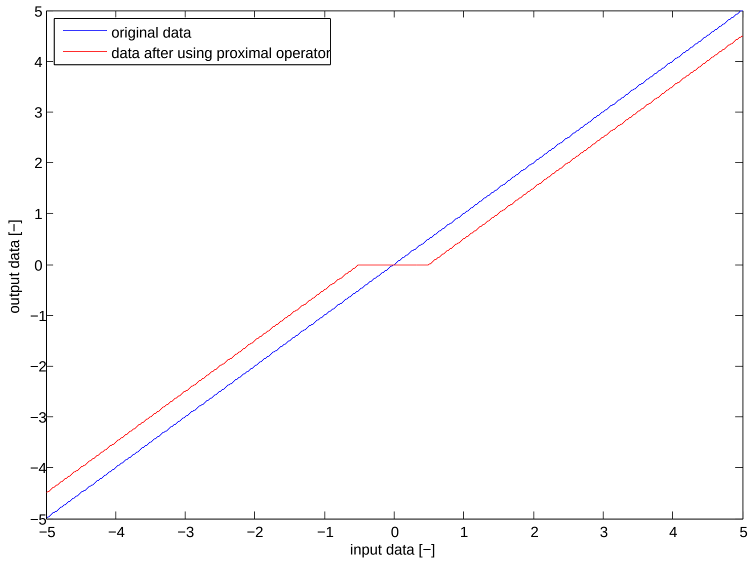
V tomto vzorci je f uzavřená konvexní funkce, která splňuje $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$. Jak je patrné, proximální algoritmy jsou velice výhodné, pouze pokud je výpočet proximálního operátora efektivní a velice rychlý na výpočet. Pokud by nebylo splněno toto kritérium, tak by se mnoho času strávilo vyhodnocováním proximálního operátoru, které se musí provádět v každé iteraci algoritmu. Další obrovskou výhodou těchto algoritmů je také fakt, že byly navrženy pro co nejobecnější využití a lze je tak využít v nejrůznějších problémech.

3.1 Proximální operátor

Jak již bylo naznačeno v úvodu této kapitoly, bude využito proximálního operátoru pro rekonstrukci dat. Proximální operátory jsou velice důležitou součástí a to proto, že nahrazují funkce, kterou jsou obtížně řešitelné nebo by jejich výpočet byl velice zdlouhavý. Volbu tohoto operátoru určují vstupní funkce z optimalizačního problému, resp. vlastnosti vstupních dat.

Jelikož se v konkrétním předpisu úlohy LASSO, viz Vzorec 4, objevuje funkce $\|x\|_1$, byl zvolen proximální operátor nazývaný "měkké prahování", dle *odkaz*. Jedná se o velice jednoduchou funkci, kdy se vstupní data porovnávají s parametrem a to dle vztahu Vzorec 7. Průběh této funkce je také zobrazen grafem 3.1.

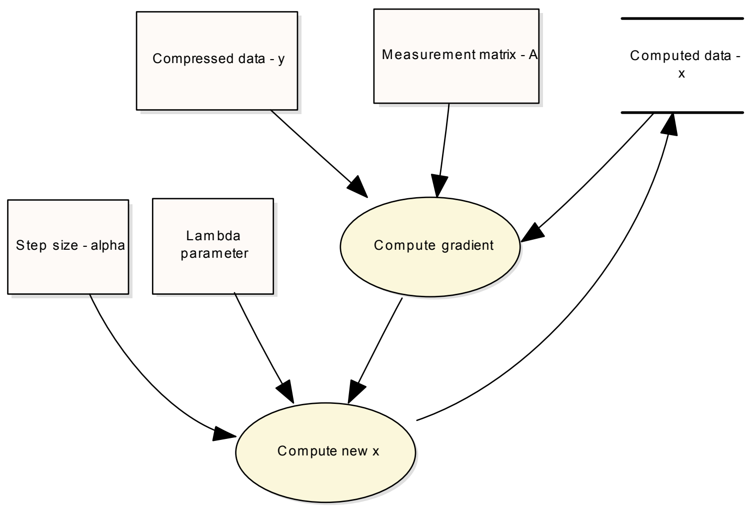
$$soft(x, \lambda) = \begin{cases} 0 & x \leq \lambda \\ x - \lambda & x > 0 \\ x + \lambda & x < 0 \end{cases} \qquad \text{(Vzorec 7)}$$



Obrázek 3.1: Průběh zvoleného proximálního operátoru pro $\lambda = 0.5$

4 Dopředno-zpětný algoritmus

Jako hlavní algoritmus, na který se tato práce zaměřuje, je dopředo-zpětný algoritmus. Jeho základní varianta, popsaná obrázkem 4.1, používá pevnou délku kroku. Je to iterační algoritmus, který získá na vstupu komprimovaná data y, měřící matici A, velikost kroku α a kompenzační parametr λ, který si volí uživatel.



Obrázek 4.1: Schéma dopředno-zpětného algoritmu

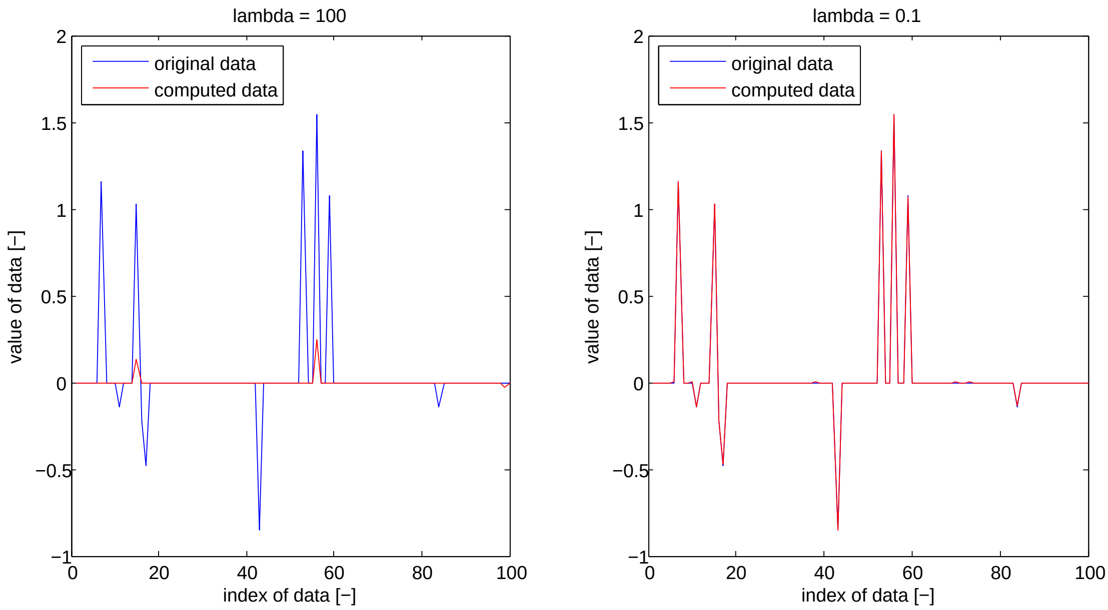
Za pomoci těchto údajů se v každém kroku počítá gradient diferenciovatelné části optimalizační úlohy LASSO, v našem případě je to $|| \, y - A \times x \, ||_2^2$, jehož předpis funkce je Vzorec 8.

$$\partial f(x) = -2 \times A^T \times (y - A \times x) \qquad \text{(Vzorec 8)}$$

Poté se již spočítá nový vektor dat x za pomoci proximálního operátoru, jehož průběh byl znázorněn v předchozí kapitole, dle vzorce viz Vzorec 9.

$$x_{n+1} = prox_{\lambda \times \alpha_n}(x_n - \alpha_n \times \partial f(x_n)) \qquad \text{(Vzorec 9)}$$

Takto provedený výpočet nám dává řešení, které postupně konverguje k optimálnímu bodu, pokud byl vhodně zvolen kompenzační parametr λ. V případě neoptimálně zvolené parametru λ je algoritmus daleko rychlejší než v případě s optimálním parametrem, ale nedochází k úplně rekonstrukci dat a získali bychom tak nepřesná data.

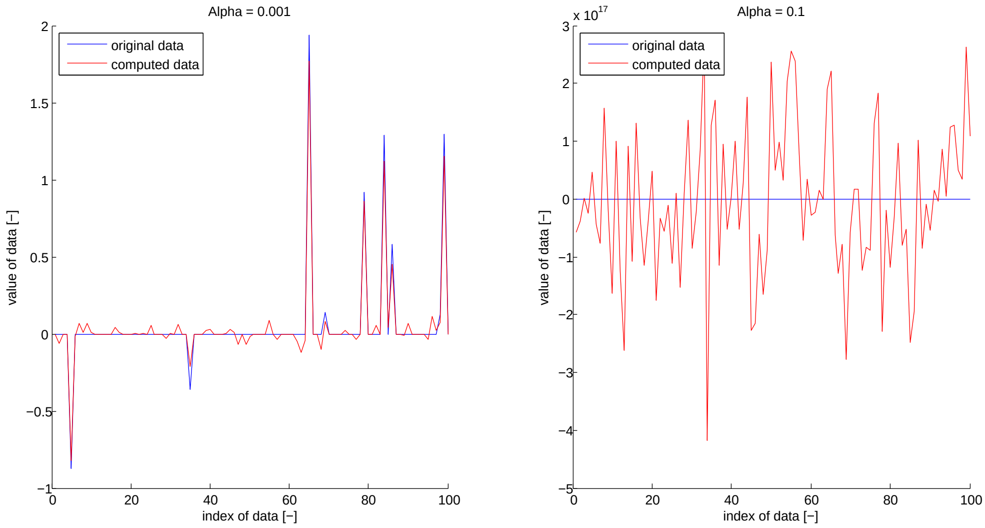


Obrázek 4.2: Ukázka průběhu algoritmu při nevhodně zvoleném parametru λ

Jak lze vidět z těchto grafů, je parametr λ je velice významný. V prvním případě, kdy byl parametr zvolen extrémně nevhodně, algoritmus nedokázal najít přesné řešení. Dokázal najít pouze část původních dat s redukovanou hodnotou a dokonce pro jeden prvek nezkovergoval vůbec k nule. Ve druhém případě již bylo nalezeno daleko přesnější řešení a nedošlo k chybnému nalezení jako v předchozím případě.

4.1 Délka kroku algoritmu

Důležitým parametrem algoritmu, jak je patrné z jeho průběhu v kapitole 4, je délka kroku. Pro pochopení fungování algoritmu byl nejprve implementován s pevnou délkou kroku. Nicméně, toto není zcela optimální, jak bude zřejmé z následujících grafů.



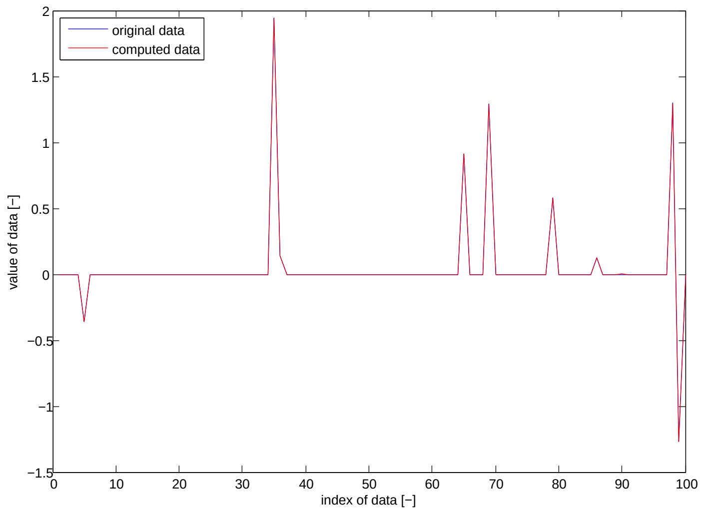
Obrázek 4.3: Ukázka průběhu algoritmu při různé volbě délky kroku

Jak lze vidět na těchto grafech, tak délka kroku má obrovský dopad, dokonce větší než kompenzační parametr λ. Na levém grafu, kde byla použita rozumná velikost kroku, je vidět, že algoritmus konverguje ke správnému řešení, vzniká tak ale mnoho nepřesností, které se nepodařilo odstranit do konce algoritmu, v tomto případě bylo nastaveno maximálně 5000 iterací. V druhém případě, kde byla použita stejná data jako v prvním grafu, lze vidět, že je algoritmus velice citlivý. Proběhlo pouze deset iterací a řešení velice rychle diverguje k nekonečnu. Z tohoto důvodu byl původní algoritmus rozšířen o dynamický výpočet velikosti kroku.

Dynamická volba kroku využívá přístupu z článku *odkaz*, kdy se velikost kroku vypočte pomocí aktuálně nalezených x_n, předchozích dat x_{n-1} a měřící matice A a to dle vztahu Vzorec 10.

$$alpha = \frac{\|A \times (x_n - x_{n-1})\|}{\|A^T \times A \times (x_n - x_{n-1})\|} \qquad \text{(Vzorec 10)}$$

Takto upravený algoritmus, již korektně rekonstruuje původní data, jak lze také vidět na grafu 4.4 a lze tak přistoupit k další části řešení této bakalářské práce.



Obrázek 4.4: Ukázka průběhu algoritmu s dynamickou délkou kroku

4.2 Podmínky optimality

Takto připravený algoritmus byl dále rozšířen o podmínky optimality, které zaručují ukončení algoritmu když změna rekonstruovaných dat je dostatečně malá. V případě, že by algoritmus toto neobsahoval v lepších situacích by pouze nedošlo k zastavení algoritmu a bylo by plýtváno pouze výpočetním časem uživatele. V horších případech by však mohlo dojít i k divergenci algoritmu z jakéhokoliv aktuálního bodu a to kvůli dynamické volbě kroku. Dynamická volba kroku je založena na výpočtu rozdílu aktuálního a předchozího kroku algoritmu. Pokud by tato změna byla dostatečně malá, vlivem nepřesnosti počítače by tak vznikla nulová data a právě kvůli tomu by se z původního výpočtu, viz Vzorec 10, stalo $0/0$. Právě kvůli těmto případům bylo využito článku [2]. Byly tedy vytvořeny další dvě funkce pro tyto podmínky. První z nich se zabývá pouze nenulovými prvky s indexy i a má předpis:

$$A_i^T * (A * x_n - y) + \lambda * sign((x_n)_i) < \epsilon \qquad \text{(Vzorec 11)}$$

Výsledek levé strany tohoto vzorce je porovnáván s uživatelem zadanou hodnotou ϵ, což je velikost odchylky chyby. Tento vzorec byl upraven pro toto konkrétní řešení, jelikož původní článek uvažoval i různé parametry λ pro každý prvek původních dat x. Jelikož první z těchto vzorců používal pouze nenulové hodnoty, bylo využito i druhé varianty vzorce pro nulové hodnoty a to ve tvaru:

$$\left| A_i^T * (A * x_n - y) \right| < \lambda \qquad \text{(Vzorec 12)}$$

4.3 Konečná verze algoritmu

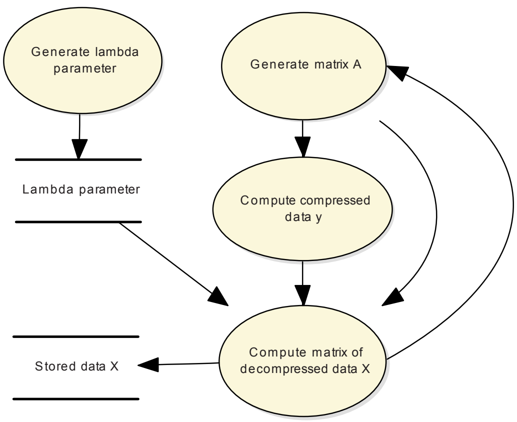
Po vypracování těchto kroků mohl být původní algoritmus, jehož diagram byl znázorněn v kapitole 4, upraven do finální podoby. Po přídání těchto funkčních bloků vypadá jeho diagram jak je zobrazeno v přílohách této práce. Tato podoba je navržena tak, aby případná změna byla snadno proveditelná a šlo tak algoritmus dále rozšiřovat pro další účely. Zároveň se tímto rozšířením dosáhlo toho, že algoritmus vždy konverguje ke správné rekonstrukci. V případě divergence se okamžitě zastaví.

5 Analýza chyby řešení

Aby bylo možné pozorovat závislost parametru λ a kvadratické chyby nalezeného řešení optimalizačního úlohy LASSO, které bylo nalezeno pomocí algoritmu navrženého v kapitole 4, je nutné vytvořit simulaci. Takovéto simulace vycházejí z toho, že se opakovaně provadí vybraný algoritmus na větším množství vygenerovaných, které jsou na sobě nezávislé.

5.1 Realizace simulace

Jednou z možných simulací je metoda Monte Carlo. Tato simulace je založená, jak již bylo zmíněno, na opakovaném generování náhodných dat, na kterých se provádí daný algoritmus a jsou takto získávány průběžné výsledky. Následně jsou tato data zprůměrována a průměr nám poté dává empirické pozorování závislosti kvadratické chyby a parametru λ.

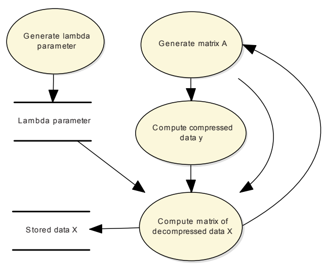


Obrázek 5.1: Simulace Monte Carlo

Protože jde pouze o opakované provádění testovaného algoritmu, je tato simulace velice jednoduchá na realizaci. Víceméně základní algoritmus lze popsat jednoduchým grafem 5.1. Nicméně jákokoliv takováto simulace má obrovskou nevýhodu a to je čas. Při větších datech může výpočet trvat i několik dní. V případě této konkrétní simulace se sto prvky λ a tisíci opakování trvala tato simulace přes jedenáct hodin. Pro porovnání byl algoritmus upraven pro $\|\cdot\|_2$ LASSO a takováto simulace již trvala přes tři dny. Jelikož je vše realizováno pomocí vývojového prostředí a jazyku MATLAB, jednou z možností bylo vytvořit knihovní funkce v jazyku C++, aby se docílilo zrychlení algoritmu a využít této knihovny funkcí místo skriptů v MATLABu. Nicméně toto by bylo velice zdlouhavé řešení a špatně testovatelné kvůli možným chybám. Z tohoto důvodu byla nalezena další možnost a tak byla simulace rozšířena o další blok, kterým se docílí toho, že výsledky nebudou ztraceny při nečekaných událostech.

5.2 Zálohování výsledků

Onou nalezenou možností, jak je zmíněno v předchozí kapitole, je průběžné zálohování výsledků. První varianta této funkce byla realizována pomocí jednoduché tabulky, kam byly ukládány pouze výsledky. Nicméně s takto zálohovanými výsledky již nebylo možné pokračovat v simulaci. Z tohoto důvodu byla vytvořena struktura v MATLABu:



Obrázek 5.2: Struktura dat

Takto připravené struktura je již připraveno pro ukládání a načítání. Z tohoto důvodu bylo využito funkcí „save“ a „load“. Obě tyto funkce jsou navíc podmíněny a to tak, že v případě funkce „save“ je do konce simulace ukládána i dočasná iterační proměnná. Jak je již patrné právě tato proměnná by měla být také obnovena při načítání dat.

6 Výsledky

7 Závěr

Seznam použité literatury

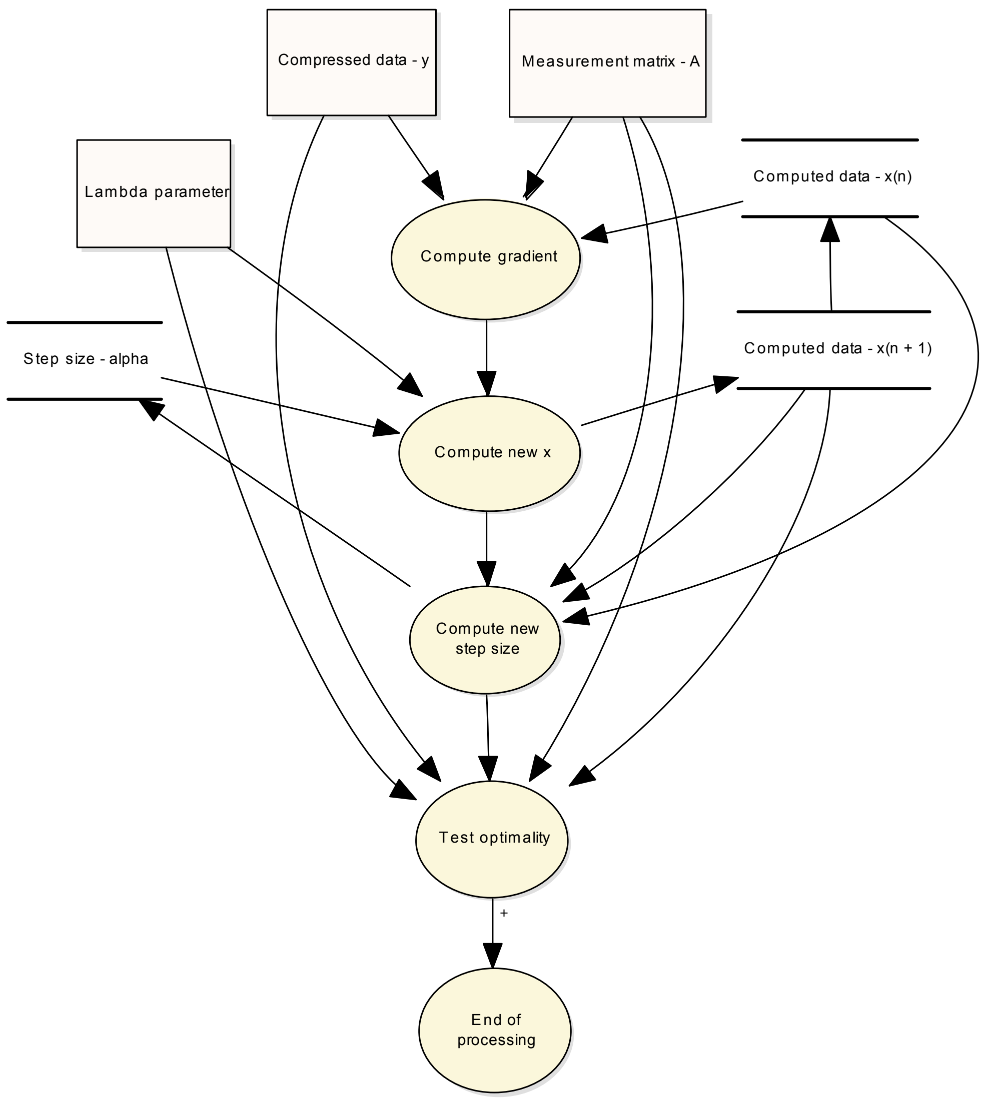
[1] BOYD, Stephen P a Lieven VANDENBERGHE. Convex optimization. Seventh printing with corrections. New York: Cambridge University Press, 2004. ISBN 05-218-3378-7.

[2] Z. Koldovský and P. Tichavský, "A Homotopy Recursive-in-Model-Order Algorithm for Weighted Lasso,"Proc. of the 41st IEEE International Conference on Audio, Speech, and Signal Processing (ICASSP 2014), Florence, Italy, pp. 4151-4155, May 2014.

Přílohy

Příloha A: Obsah přiloženého CD

Příloha B: Diagram finálního algoritmu



Obrázek 1: Diagram finálního algoritmu