Uniwersytet Ekonomiczny
w Krakowie

Praca licencjacka

Wybrane estymatory parametrów prostej regresji w strumieniowym przetwarzaniu danych

Zygmunt Zawadzki

Kierunek: Finanse i Rachunkowość
Specjalność: Rynki Finansowe

Nr albumu: 161509

Promotor
dr hab. Daniel Kosiorowski

Wydział Finansów

Kraków 2015

Spis treści

Wstęp

Analiza wielkich zbiorów danych jest bardzo interesującym zagadnieniem z punktu widzenia ekonomii i finansów. Gospodarka i rynki finansowe codziennie generują olbrzymie ilości informacji, których analiza może dostarczyć cennych wiadomości na temat stanu światowej ekonomii lub być wykorzystana by znaleźć okazję do inwestycji, szczególnie w kontekście tak zwanego handlu algorytmicznego, w którym decyzja o zajęciu określonej pozycji na rynku zostaje podjęta bez udziału człowieka.

Równocześnie analiza wielkich zbiorów danych stawia przed analitykiem wyzwania niespotykane w klasycznej analizie statystycznej. Dla zbiorów danych rzędu kilku tysięcy obserwacji estymacja modelu statystycznego trwa często poniżej jednej sekundy. Może się jednak okazać, że ten sam algorytm w przypadku większego zbioru, rzędu kilku milionów obserwacji, będzie zbyt złożony obliczeniowo, przez co bezużyteczny. Zagadnienia związane ze złożonością obliczeniową zostaną omówione w pierwszej części pracy, wraz z szerszym przedstawieniem problemów i potencjalnych rozwiązań w przypadku analizy wielkich zbiorów danych.

Szczególnym przypadkiem wielkich zbiorów danych o szczególnej strukturze są strumienie danych. Intuicyjnie rzecz ujmując są to ciągi obserwacji o nieokreślonej długości, często pojawiających się w nieregularnych odstępach czasu. Koncepcja strumieni danych wywodzi się z informatyki, lecz może zostać z powodzeniem zastosowane w analizie danych ekonomicznych - szczególnie cen instrumentów finansowych pochodzących z giełd światowych, które doskonale wpisują się w definicję strumieni.

Do monitorowania strumienia danych pod kątem różnych zależności można użyć klasycznego modelu regresji liniowej. Regresja pozwala określić typową zależność pomiędzy zmiennymi, a znane i powszechnie stosowane metody pozwalają formalnie ocenić określony estymator pod kątem jego jakości i przydatności w prognozowaniu i empirycznej weryfikacji tez stawianych przez teorię ekonomii. Jednak w przypadku strumieni danych wiele z założeń formułowanych dla estymatorów regresji, a szczególnie dla bardzo powszech-

nego estymatora metodą najmniejszych kwadratów, może być nie spełnione. Jednym z przykładów naruszenia założeń, jest przypadek, dynamicznie zmieniających się w czasie reżimów procesu generującego dane. Estymowana zależność użyteczna do pewnej chwili, w następnej, może nie mieć większego sensu.

W pracy wykorzystano trzy proste metody estymacji modelu regresji - dwie metody parametryczne, i jedną nieparametryczną. Pierwszą metodą była regresja najmniejszych kwadratów (MNK), jako najbardziej rozpowszechniona, odznaczająca się przy określonych założeniach bardzo dobrymi własnościami, jednak jest to metoda nieodporna na obserwacje odstające. Drugim zastosowanym algorytmem była regresja metodą największej głębi regresyjnej (NGR). W przypadku typowych założeń dla MNK, metoda NGR odznacza się gorszymi własnościami pod względem efektywności, jednak w dużym stopniu jest niewrażliwa na obserwacje odstające. Dla NGR nawet 30% obserwacji może przyjmować dowolnie skrajne wartości, mimo to wskazania modelu pozostaną sensowne. Jako metodę nieparametryczną przyjęto algorytm K-najbliższych sąsiadów (KNN). W tym przypadku na uwagę zasługiwała głównie prostota obliczeniowa - w przypadku dwuwymiarowym - jest to najszybszy algorytm użyty w pracy, Jego złożoność obliczeniowa jest liniowa względem ilości obserwacji w zbiorze uczącym.

Własności wyżej wymienionych metod zbadano przy pomocy symulacji komputerowych wykonanych w pakiecie statystycznym R.

Rozdział 1

Wielkie zbiory danych

W literaturze można znaleźć kilka definicji zbioru danych, który można określić mianem „Wielkiego zbioru danych". Na potrzeby tej pracy zostanie przyjęta definicja wzorowana na [22]. Wielki zbiór danych jest to zbiór, który charakteryzują trzy określenia:

- Rozmiar - ilość zajmowanego miejsca,
- Szybkość napływania nowych danych - tempo w jakim zbiór się rozrasta,
- Struktura zbioru - poziom komplikacji,

W dalszej części rozdziału powyższe hasła zostaną szerzej omówione, wraz z opisem podstawowym problemów z nimi związanych.

1.1. Rozmiar zbioru danych a złożoność obliczeniowa

1.1.1. Klasyfikacja wielkości zbioru

Jedną z możliwych klasyfikacji zbioru według wielkości jest klasyfikacja zaproponowana przez Hubera [13], oparta na rozmiarze zbioru w bajtach:

- malutki (tiny) - 10^2 bajtów,
- mały (small) - 10^4 bajtów,
- średni (medium) - 10^6 bajtów,
- duży (large) - 10^8 bajtów,
- ogromny (huge) - 10^{10} bajtów,
- monstrualny (monster) - 10^{12} bajtów.

Jednocześnie autor stwierdza, że w istocie klasyfikacja zbioru danych jako wielki jest mocno subiektywna i zależy przede wszystkim od postawionego zadania, umiejętności analityka i dostępnych zasobów.

1.1.2. Złożoność obliczeniowa - podstawowe definicje

Jedną z podstawowych metod oceny algorytmu jest notacja O określająca rząd wielkości ilości operacji potrzebnych do wykonania algorytmu. Dla danej funkcji $g(x)$ oznaczamy przez $O(g(x))$ zbiór funkcji takich że:

$O(g(x)) = \{f(x)\}$ istnieją dodatnie stałe c i n_0 takie, że $0 \leq f(x) \leq c \cdot g(x)$ dla wszystkich $n \geq n_0$.

Głównymi przewagami notacji O nad zwykłym pomiarem czasu wykonywania się określonych algorytmów, jest możliwość porównania ich niezależnie od platformy sprzętowej i użytego języka programowania oraz ocena czy dany algorytm w ogóle skończy się w sensownym czasie. Jedną z wad takiej notacji jest fakt, że daje ona jedynie rząd wielkości dla przypadku pesymistycznego - najgorszego z możliwych - jednak nie jest to problem znaczący w kontekście analizy wielkich zbiorów danych. Więcej na temat notacji asymptotycznych można znaleźć w książce [5]. W tabeli 1.1 przedstawiono przykładowe ilości operacji jakie należy wykonać dla różnych klas złożoności algorytmów.

Tablica 1.1: Przybliżona ilość operacji do wykonania dla różnych klas złożoności

Funkcja \ N	10	100	1000	10^6	10^9
$\log_2 N$	4	7	10	20	30
N	10	100	1000	10^6	10^9
$N \cdot \log_2 N$	34	66	59966	19931569	$3 \cdot 10^{10}$
$N^{3/2}$	32	1000	$3.16 \cdot 10^4$	10^9	$3.16 \cdot 10^{13}$
2^N	1024	$1.26 \cdot 10^{30}$	$1.07 \cdot 10^{301}$	$> 10^{301030}$	$> 10^{301029996}$

Dla porównania - liczba protonów we wszechświecie ma 79 cyfr; liczba nanosekund od Wielkiego Wybuchu ma 27 cyfr.

1.1.3. Problem złożoności obliczeniowej w przypadku wielkich zbiorów danych

Głównym problemem w przypadku *wielkich zbiorów danych* jest odpowiedź na pytanie czy algorytm użyty do analizy zakończy działanie w sensownym czasie, przy czym definicja „sensownego czasu" zależy od badacza - w pewnych przypadkach może on oczekiwać by algorytm wykonał się w ciągu ułamka sekundy (w przypadku handlu wysokiej częstotliwości [2]), minutach (obrazowanie medyczne, analizy ekonomiczne), lub miesiącach (symulacje fizyczne).

Do analizy wielkich zbiorów danych powinno stosować się algorytmy o złożoności obliczeniowej mniejszej lub równej $O(n^{\frac{3}{2}})$ [13] - w przypadku większej złożoności czas oczekiwania na zakończenie algorytmu może okazać się zbyt duży.

Przykład: Do analizy zbioru danych o rozmiarze $n = 10^{10}$ użyto algorytmu o złożoności obliczeniowej rzędu $O(n^2)$. W takim przypadku do zakończenia działania algorytmu będzie potrzebnych około 10^{20} operacji. Przyjmując, że aktualnie najszybszy dostępny superkomputer ma wydajność na poziomie $17,59 PFLOPS$[1][2], czas potrzebny na wykonanie takiego algorytmu na superkomputerze wyniesie około półtorej godziny. Dla procesora domowego o wydajności rzędu 100 GFlops [3], obliczenia trwałyby około 32 lata - co jest wielkością z pewnością nieakceptowalną.

1.1.4. Techniki radzenia sobie ze złożonością obliczeniową

W tej części zostaną omówione trzy podstawowe sposoby radzenia sobie z rozmiarem problemu:

- analiza oparta na losowo wybranym podzbiorze obserwacji,
- stosowanie algorytmów heurystycznych i aproksymacyjnych,
- programowanie równoległe i rozproszone.

Analiza oparta na losowo wybranym podzbiorze obserwacji

W pewnych przypadkach do analizy nie jest potrzebny cały posiadany zbiór danych, gdyż analizę można oprzeć na wybranym podzbiorze, mniejszym o kilka rzędów wielkości. Reszta zbioru danych może być na w takim przypadku użyteczna jedynie do badania poprawności modelu.

Stosowanie algorytmów heurystycznych i aproksymacyjnych

Może się okazać, że analiza problemu o zadanym rozmiarze, przy użyciu określonego algorytmu jest po prostu niemożliwa - w takim przypadku najprostszym podejściem będzie zmiana używanego algorytmu. Nowy algorytm może działać na zasadzie przeszukiwania tylko podzbioru możliwych rozwiązań (np. algorytm sprawdzania prymarności [11] lub podejście omówione powyżej operowania na podzbiorze możliwych rozwiązań) lub niezależnie od wyniku kończyć działanie po określonej liczbie kroków albo osiągnięciu określonej dokładności (np. metody optymalizacji [20]).

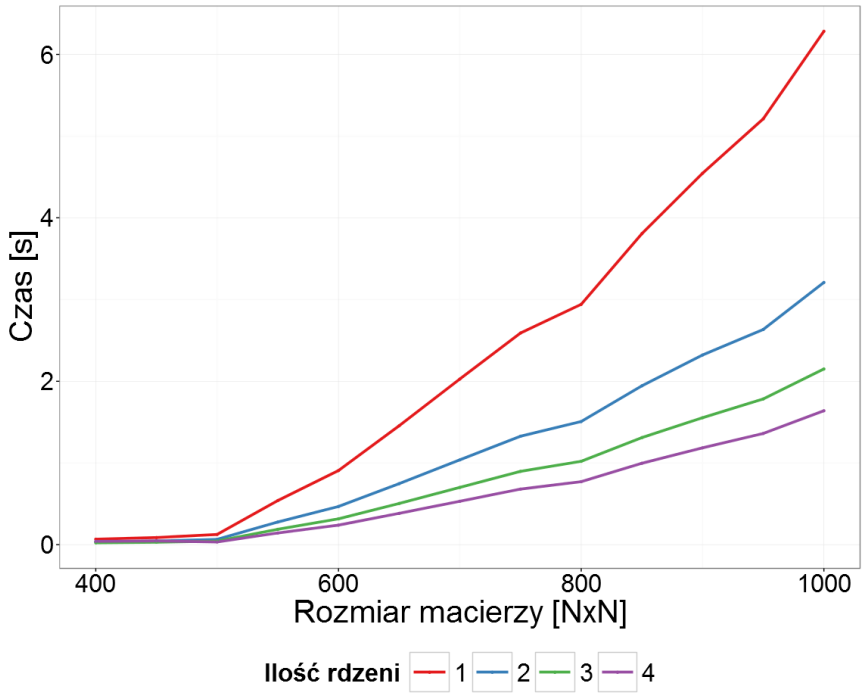
1. FLOPS - (ang. FLoating point Operations Per Second) - liczba operacji zmiennoprzecinkowych na sekundę. Jest to jednostka określająca moc obliczeniową komputera.
2. PFLOPS - $10^{15} FLOPS$
3. Wydajność procesora w komputerze autora

Programowanie równoległe i rozproszone

Jednym z głównych sposobów radzenia sobie z rozmiarem problemu jest zastosowanie metod programowania równoległego, bądź rozproszonego. W takim przypadku główne zadanie dzielone jest na mniejsze części, które są rozwiązywane niezależnie od siebie - na końcu ich wyniki są łączone dając ostateczny rezultat.

Poniżej zamieszczono przykład obrazujący zastosowanie powyższego podejścia.

Dane są macierz A rozmiaru n na p i macierz B rozmiaru p na m. Jeżeli $C = AB$ a $c_{i,j}$ jest elementem macierzy C znajdującym się na pozycji (i,j) to $c_{i,j} = \sum_{k=1}^{p} a_{i,k} \cdot b_{k,j}$. W podstawowym algorytmie wpierw obliczone zostałoby $c_{1,1}$, następnie $c_{1,2}$ itd. Złożoność obliczeniowa algorytmu wyniesie $O(n \cdot m \cdot p)$. Łatwo jednak zauważyć, że obliczanie elementu $c_{i,j}$ w żaden sposób nie zależy od obliczania innych elementów macierzy C. Stosując programowanie równoległe czas wykonywania algorytmu możemy skrócić proporcjonalnie do ilości posiadanych rdzeni procesora, przydzielając każdemu z procesorów po części zadań - teoretycznie posiadając $n \cdot m$ procesorów czas wykonywania będzie proporcjonalny do p. Na rysunku 1.1 zaprezentowano czas wykonywania się mnożenia macierzowego w zależności od użytej liczby rdzeni procesora.



Rysunek 1.1: Porównanie czasu wykonywania się algorytmu mnożenia macierzy dla różnej ilości rdzeni (procesor Intel Core I7 3770K).

Źródło: Obliczenia własne

W przypadku obliczeń równoległych istnieje jednak górne ograniczenie określające maksymalny stopień przyspieszania, który można uzyskać używając wielu procesorów. Niech $S(p,n)$ oznacza krotność przyspieszenia algorytmu równoległego (np. 1 oznacza brak przyspieszenia, 2 - algorytm równoległy będzie maksymalnie dwa razy szybszy), dla problemu wielkości n, przy użyciu p procesorów, a s udział sekwencyjnej części w całym programie, wtedy:

$$S(p,n) \leq \frac{1}{s+(1-s)/p}. \tag{1.1}$$

Powyższy wzór nosi nazwę prawa Amdhala i służy do wyznaczania górnego ograniczenia przyspieszenia jako funkcji s i p przy ustalonym n. Więcej informacji na temat oceny algorytmów równoległych można znaleźć w [6]. Również w przypadku niektórych algorytmów zastosowanie technik programowania równoległego w ogóle nie jest możliwe - taka sytuacja zachodzi w przypadku, gdy kolejny krok algorytmu zależy bezpośrednio od kroku poprzedniego.

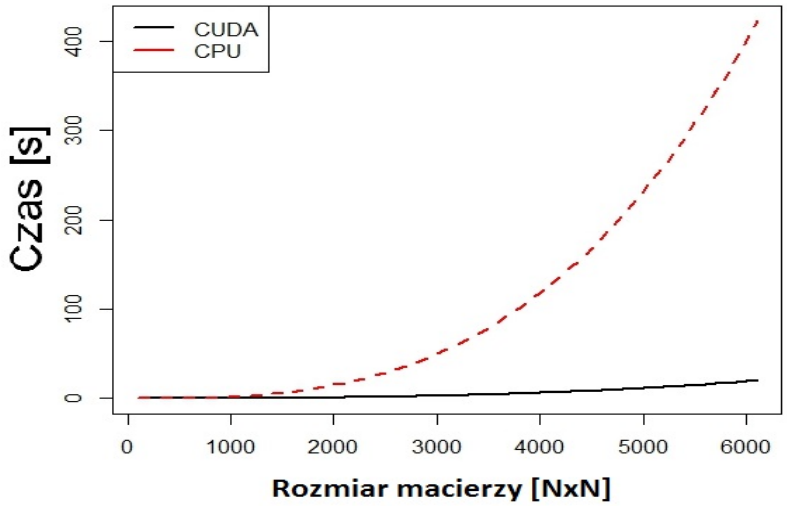
Dodatkowym problemem związanym z zastosowaniem programów równoległych i rozproszonych może okazać się trudność z ich implementacją. Jednak w tym zakresie prowadzone są prace w celu umożliwienia wykorzystania architektury wieloprocesorowej przy jak najmniejszym wysiłku ze strony programisty. W efekcie tych prac powstała w firmie Google platforma MapReduce [7], ułatwiająca programowanie klastrów komputerów na wysokim poziomie abstrakcji, to znaczy pewne zadania, takie jak podział problemu na podproblemy, rozkład zadań pomiędzy węzły, synchronizacja wyników, czy podejmowanie działań na wypadek awarii któregoś z węzłów w trakcie obliczeń, są rozwiązane na poziomie tej platformy, przez co ich implementacja nie należy do zadań piszącego algorytm, co z kolei znacznie ułatwia i przyspiesza proces tworzenia potrzebnej aplikacji.

Istnieje otwarta (oparta na licencji open source) implementacja platformy MapReduce nosząca nazwę Apache Hadoop. Również w przypadku pakietu statystycznego R istnieje i jest rozwijanych kilka pakietów implementujących to podejście, między innymi RHadoop, RHIPE [9].

Na uwagę zasługuje fakt, iż w ostatnim czasie dostęp do klastrów obliczeniowych, został znacząco ułatwiony, między innymi dzięki możliwości wynajęcia określonej mocy obliczeniowej na pewien okres czasu (na przykład wynajęcie klastra złożonego z 100 węzłów na 10 godzin potrzebnych do wykonania potrzebnej analizy). Rozwiązanie to jest szczególnie interesujące w przypadku małych i średnich przedsiębiorstw, dla których potrzeba wykorzystania dużej mocy obliczeniowej może być sporadyczna, a koszty utrzymania własnej infrastruktury zbyt duże.

Warto również zwrócić uwagę na rozwijający się dynamicznie sektor obliczeń pro-

wadzonych na GPU[4]. Rysunek 1.2 pokazuje czas wykonywania się algorytmu mnożenia macierzy na karcie graficznej NVidia GT640 z zastosowaniem technologi CUDA.



Rysunek 1.2: Porównanie czasu wykonywania się mnożenia macierzowego dla implementacji na procesorze i karcie graficznej.

Źródło: Obliczenia własne

Zdaniem autora programowanie kart graficznych w obecnym stadium rozwoju jest zbyt trudne i czasochłonne z perspektywy statystyka czy analityka, gdyż w czasie budowania programu należy mieć na uwadze bardzo niskopoziomowe zagadnienia związane z zarządzaniem pamięcią, przesyłaniem danych z karty graficznej (w większości przypadków pamięć operacyjna nie jest współdzielona pomiędzy procesorem i kartą graficzną), równomiernym obciążeniem procesorów strumieniowych itd. Jednak również w tym zakresie prowadzone są prace nad stworzeniem odpowiednich narzędzi ułatwiających korzystanie z obliczeń na procesorach GPU - przykładem może być dodatek do środowiska MatLab firmy MathWorks[23] jak i pakiety do R - np „gputools" i „Rpgu".

Należy również zwrócić uwagę, iż w obecnych superkomputerach spora część mocy obliczeniowej pochodzi właśnie z układów GPU. Więcej na temat obliczeń na kartach graficznych można znaleźć w [26][16].

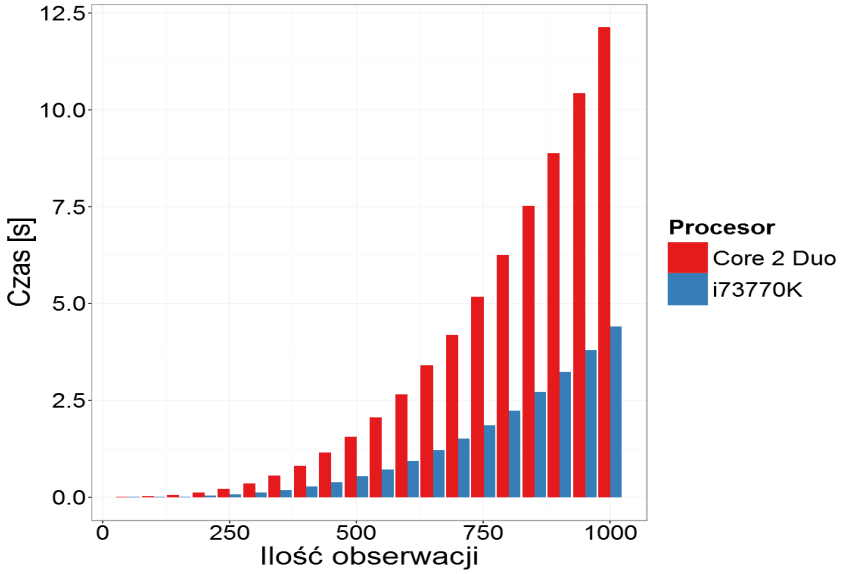
1.1.5. Rozwój technologii a wielkie zbiory danych

Na zakończenie paragrafu związanego z rozmiarem problemu zostanie krótko omówiony problem zasygnalizowany przez Petera Hubera w [13]:

„Czy zwiększenie wydajności komputerów zwiększyło moc czy wygodę?"

4. GPU - (ang. Graphical Procesor Unit) - karta graficzna komputera

Odpowiedź na to pytanie nie mieści się w zakresie tej pracy. Pewne wyobrażenie o problemie może dać symulacja przygotowana przez autora, porównująca czas wykonywania się naiwnego algorytmu obliczania głębi regresyjnej 2d dla procesorów domowych pochodzących z roku 2008 (Intel Core 2 duo) i roku 2012 (Intel Core I7 3770K). Jej wyniki zaprezentowano na rysunku 1.3.



Rysunek 1.3: Porównanie czasu wykonania naiwnego algorytmu obliczania głębi regresyjnej 2d dla procesorów Intel Core I7 3770K (rok 2012) i Intel Core 2 duo (rok 2008).

Źródło: Obliczenia własne

W tym przypadku uzyskano około trzykrotne przyspieszenie. Po upływie 4 lat w tym samym czasie, dla tego algorytmu jest się w stanie rozwiązać problem większy o około 50%.

1.2. Szybkość napływania nowych danych

Drugą cechą charakteryzującą wielki zbiór danych jest szybkość napływania nowych obserwacji, co jednocześnie wiąże się z tempem w jakim zbiór się powiększa. Najprostszym przykładem zbioru powiększającego się w dynamiczny sposób są dane giełdowe, gdzie ceny transakcyjne i informacje o zmianach na książce zleceń pojawiają się w krótkich interwałach, często wielokrotnie w ciągu sekundy. Problemy i sposoby radzenia sobie w przypadku danych o dużej częstotliwości zostaną szerzej omówione w dalszej części pracy dotyczącej zagadnienia strumieni danych.

1.3. Struktura zbioru

Złożoność struktury zbioru można rozumieć dwojako. Po pierwsze - złożoność pod kątem informacji, których dotyczą dane znajdujące się w zbiorze. Przykładem może być zbiór danych ze spisu powszechnego, w którym znajdują się informacje dotyczące wielu aspektów gospodarki, demografii itd. Po drugie - pod względem sposobu zapisu danych - czy w łatwy sposób można wydobyć interesujące dane?

Pierwszy przypadek traktowania złożoności struktury zbioru pod kątem ilości informacji w nim zawartych wydaje się być raczej powodem dla którego analizuje się wielkie zbiory danych, niż problemem z którym należy się zmierzyć. W takim przypadku użyte narzędzia i napotkane trudności zależeć będą od informacji, które badacz chce uzyskać na podstawie danego zbioru. Z faktu, iż jest to temat zbyt obszerny autor ogranicza się do przedstawienia problemów związanych z drugim sposobem rozumienia złożoności zbioru jako sposobu jego magazynowania i postaci danych jakie się w nim znajdują.

Trudność związana z magazynowaniem wielkich zbiorów danych nie zależy jedynie od jego fizycznego rozmiaru (rozumianego w kontekście przestrzeni dyskowej), ale również od innych parametrów które chcemy uzyskać, na przykład:

- zachowanie określonych relacji, bądź hierarchii w danych,
- wysoka dostępność danych,
- wysoka przepustowość bazy danych.

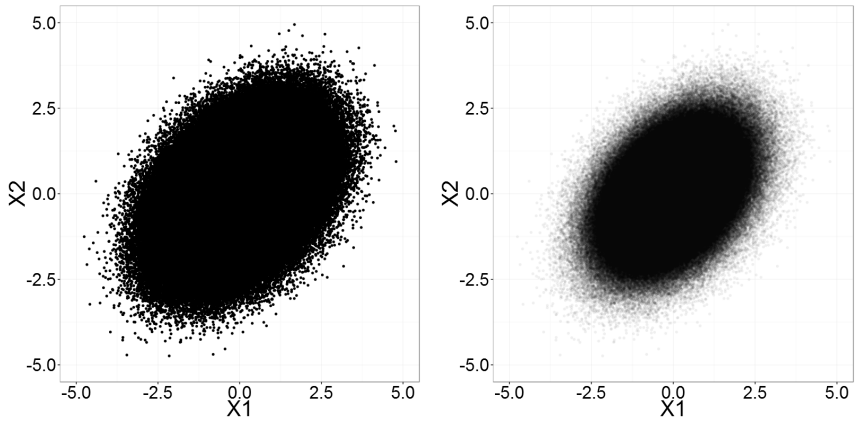
W przypadku próby zachowania określonych relacji pomiędzy danymi naturalnym rozwiązaniem jest zastosowanie relacyjnego modelu bazy danych. W takim podejściu dostęp do interesujących danych jest w znacznym stopniu ułatwiony, odbywa się to jednak kosztem innych właściwości, takich jak przepustowość i czas dostępu, które w innych zagadnieniach mogą mieć bardziej kluczowe znaczenie. Przykładowo firma Google opracowała własną, nierelacyjną bazę danych BigTable[3], która pozwala na analizę 20 petabajtów $2 \cdot 10^{16}$ danych dziennie[27]. Rozmiar ten w znaczący sposób wykracza poza przedstawioną skalę klasyfikującą zbiór danych według rozmiaru.

1.3.1. Wizualizacja wielkiego zbioru danych

W przypadku klasycznej analizy statystycznej badacz ma dostęp do szerokiego spektrum metod wizualizacji danych, jednak dla wielkich zbiorów danych wizualizacja może być utrudniona. Pierwszym problemem jest sam rozmiar zbioru, przez który interesujące zależności (np. obszary podwyższonej gęstości) mogą być niewidoczne na wykresie.

Na rysunku 1.4 została zaprezentowana jedna z możliwych metod pozwalającą lepiej przedstawić na wykresie punktowym zbiór danych o liczebności 10^6 obserwacji (w przykładzie są to dane wylosowane z dwuwymiarowego rozkładu normalnego). Polega ona na

wypełnieniu punktu na wykresie tylko w określonym stopniu tak, by dopiero określona z góry ilość punktów nakładających się na siebie dawała pełne wypełnienie.



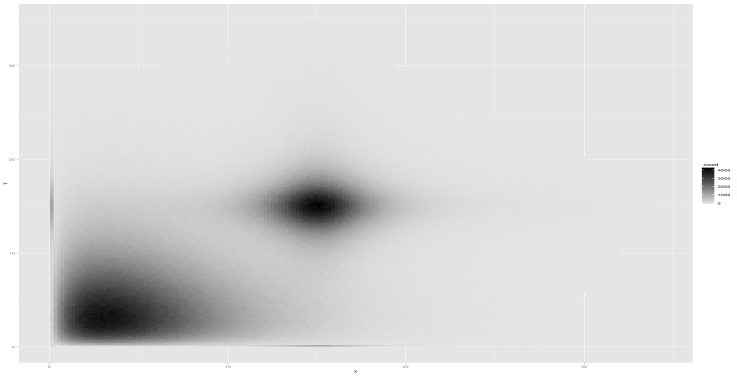
(a) Bez zastosowania wyżej omawianej metody.

(b) Z zastosowaniem wyżej omówionej metody - obszar o mniejszej gęstości na obrzeżach zbioru jest lepiej widoczny.

Rysunek 1.4: Prosta wizualizacja zbioru danych zawierającego 10^6 *obserwacji*

Źródło: Obliczenia własne - R Project

Kolejnym sposobem (będącym jednocześnie rozwinięciem metody zaprezentowanej powyżej) jest podział przestrzeni na komórki i zliczenie obserwacji należących do poszczególnych komórek i na tej podstawie określenie poziomu wypełnienia punktu na wykresie, będącego reprezentacją pojedynczej komórki. Powyższą metodę zastosowano w pakiecie "bigvis"[29] środowiska R. Rysunek 1.5 utworzono przy zastosowaniu tego pakietu.



Rysunek 1.5: Wizualizacja przykładowego zbioru danych zawierającego 10^8 *obserwacji przy użyciu pakietu „bigvis" środowiska R*

Źródło: Obliczenia własne - R Project, pakiet bigvis

Więcej na temat wizualizacji wielkich zbiorów danych można znaleźć w: [14] - wizualizacja struktury sieciowej, [4] - sposoby przedstawiania wielkich zbiorów danych w programie SAS, [12] - prezentacja danych tekstowych.

Rozdział 2

Regresja w zagadnieniu strumieni danych

Szczególnym zagadnieniem związanym z analizą wielkich zbiorów danych są strumienie danych. Intuicyjnie „strumień danych" można określić jako ciąg obserwacji o nieokreślonej długości. Różnica pomiędzy analizą strumienia, a tradycyjnie rozumianą analizą procesu stochastycznego polega na fakcie, iż w przypadku procesu zakłada się ustalony przedział czasowy [0,T] i na tej podstawie dokonywane są obliczenia. Natomiast w przypadku strumienia danych taki przedział nie jest ustalony, a pojawienie się nowych obserwacji oznacza nową analizę. Analizę strumienia danych można określić jako sekwencję analiz procesu stochastycznego[17].

Strumienie danych mają bardzo szerokie zastosowanie praktyczne związane z między innymi z monitorowaniem pracy różnorodnych urządzeń, czy zjawisk (np. w takim przypadku strumieniem mogą wskazania jakiegoś detektora), jak również w przypadku rynków finansowych - analiza danych giełdowych pod kątem wyszukiwania okazji inwestycyjnych. Szczególnie ostatnie zagadnienie związane jest z zainteresowaniami autora i zostanie rozwinięte w dalszej części pracy.

Analiza strumieni danych wiąże się z wieloma wyzwaniami zarówno z punktu widzenia statystyki i informatyki. W pierwszej kolejności zostaną omówione zagadnienia związane z aspektami informatycznymi.

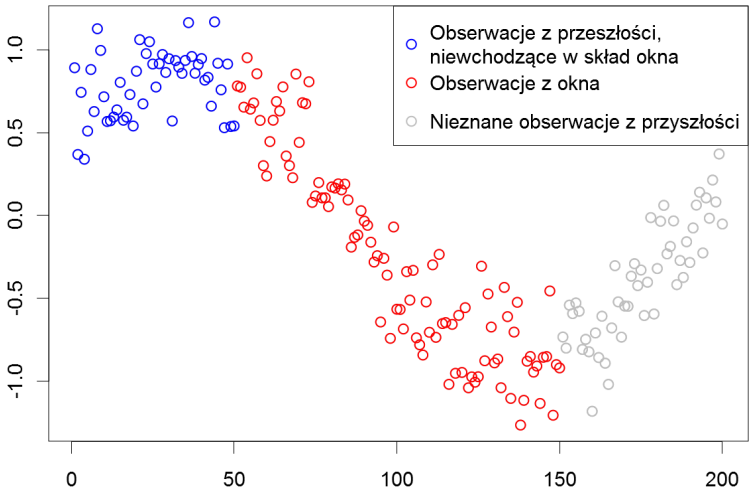
2.0.2. Okno danych - ilość używanych obserwacji do analizy strumienia w określonym momencie czasowym

Analizę w przypadku strumieni danych prowadzi się o stale aktualizowaną próbę, najczęściej w oparciu o okno ustalonej długości.

Niech $\{X_t, t \in \mathbb{N}\} = (X_1, X_2, ...)$ oznacza strumień danych, gdzie X_k to k-ta obserwacja pochodząca ze strumienia. Okno $W_{i,n}$ w analizie strumieni danych zdefiniowane jest jako ciąg obserwacji X_k kończących się w X_i, o długości n: $W_{i,n} = (X_{i-n+1}, ..., X_i)$.

Rysunek 2.1 przedstawia przykładowe okno danych. Obserwacje zaznaczone na czer-

wono, wchodzą w skład okna i na ich podstawie budowany jest model. Kolorem niebieskim przedstawiono obserwacje z przeszłości które nie będą już używane w analizie, szare obserwacje to nieznane w danej chwili wartości strumienia z przyszłości.



Rysunek 2.1: Przykładowe okno danych

Źródło: Obliczenia własne

Długość używanego okna zależy przede wszystkim od badacza i cech procesu które stara się uchwycić. Im dłuższe okno, tym zależności bardziej długoterminowe. Równocześnie można monitorować kilka okien różnej długości, o różnej częstotliwości zbierania danych.

2.0.3. Szybkość pojawiania się nowych danych

Problem związany z szybkością pojawiania się nowych danych został już zasygnalizowany w poprzednim rozdziale, związanym bezpośrednio z wielkimi zbiorami danych - w tej sekcji zostanie on szerzej omówiony.

W przypadku strumieni danych wraz z pojawieniem się nowej informacji używany model powinien zostać zaktualizowany, tak by ująć nową wiedzę. Może jednak zaistnieć sytuacja w której czas potrzebny na przebudowanie modelu znacząco przekracza okres do pojawienia się nowej obserwacji. Najprostszym rozwiązaniem w takiej sytuacji jest zbieranie danych w pakiety określonej wielkości (lub z określonego okresu) i w tej formie przekazywanie ich do modelu. W tym przypadku należy się jednak liczyć z tym, iż decyzje nie są podejmowane w oparciu o najnowszą informację - w wielu zastosowaniach takie podejście jest jednak wystarczające, a nawet pożądane. W przypadku strategii giełdowych w pewnych przypadkach pożądane jest zmniejszenie częstotliwości otrzymywanych danych, tak by zmniejszyć szum związany efektami mikrostruktury rynku[8].

Bardzo ciekawym rozwiązaniem z punktu widzenia zarówno praktycznego, jak i algorytmicznego jest użycie, lub zaprojektowanie algorytmu którego złożoność obliczeniowa w momencie jego aktualizacji wynosi $O(1)$ (stała). W tym przypadku na początkowym oknie $W_{i,n}$ można użyć algorytmu, o stosunkowo dużej złożoności obliczeniowej (np. $O(n^3)$), a dla kolejnych okien $W_{i+k,n}$ szybko aktualizować parametry zbudowanego modelu. Natomiast sytuacja w której dla każdego okna należy wykonać tę samą ilość operacji może okazać się nieakceptowalna, gdyż czas potrzebny na ponowną estymację parametrów modelu, może w znaczący sposób przekraczać okres w którym otrzymane wyniki mają sens (np. otrzymanie prognozy pogody na okres 4 najbliższych dni, dopiero po 6 dniach obliczeń, por. [15]).

2.1. Metoda najmniejszych kwadratów

Estymator regresji liniowej metodą najmniejszych kwadratów ma bardzo długą historię, jego pierwsze opisy dokonane przez Carla Fredricha Gaussa i Adriena-Marie Legendre pochodzą z końca XVIII wieku, jednak ciągle jest bardzo szeroko stosowany.

Modelu w przypadku regresji liniowej ma postać:

$$\mathbf{y} = \theta\mathbf{X} + \epsilon, \tag{2.1}$$

gdzie, $\mathbf{y}$ to wektor zmiennej objaśnianej, $\mathbf{X}$ jest macierzą zmiennych objaśniających, θ to wektor parametrów, natomiast ϵ jest to wektor odzwierciedlający wpływ czynników przypadkowych lub nieuwzględnionych w zbiorze zmiennych objaśnianych.

Dana jest próba $\mathbf{Z_N} = \{(\mathbf{x_1}, y_1), ..., (\mathbf{x_N}, y_N)\} \subset \mathbb{R}^{p+1}$, gdzie $\mathbf{x_i} = (x_{1,i}, ...x_{p,i})$ oznacza i-ty wektor zmiennych objaśniających o liczbie współrzędnych p, natomiast y_i i-tą zmienną objaśnianą, N jest to ilość obserwacji w próbie. Estymator MNK zostaje wyznaczony poprzez minimalizację kwadratów reszt 2.2.

$$\hat{\theta}_{MNK} = argmin(\sum_{i=1}^{N}(y_i - \theta\mathbf{x_i})^2). \tag{2.2}$$

Przy założeniach, że zmienne objaśniające są liniowo niezależne, estymator MNK można uzyskać rozwiązując układ równań:

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}., \tag{2.3}$$

gdzie $\mathbf{X}^T$ oznacza transpozycję macierzy obserwacji objaśniających, $(\mathbf{X}^T\mathbf{X})^{-1}$ jest macierzą odwrotną do macierzy $\mathbf{X}^T\mathbf{X}$.

Dodatkowo przy założeniach, że:

- $E(\epsilon) = \mathbf{0}$,
- $D^2(\epsilon) = E(\epsilon\epsilon^T) = \sigma^2\mathbf{I}$ i $\sigma^2 < \infty$,

- $\epsilon_i \sim N(0,\sigma)$,

estymator MNK jest klasy BLUE[1]. $E(\epsilon)$ oznacza wartość oczekiwaną ϵ, $D^2(\epsilon)$, to macierz kowariancji, σ to określona liczba, natomiast $\mathbf{I}$ to macierz jednostkowa.

2.1.1. Złożoność obliczeniowa

W praktyce z powodów stabilności numerycznej do wyznaczania wektora θ nie korzysta się z równania 2.3. Szukany wektor znajduje się poprzez rozwiązanie układu równań normalnych ze względu na θ:

$$\mathbf{X}^T\mathbf{X}\theta = \mathbf{X}^T\mathbf{y}, \tag{2.4}$$

przy pomocy różnych algorytmów dekompozycji macierzy $\mathbf{X}^T\mathbf{X}$. Wykorzystuje się do tego między innymi dekompozycję Choleskiego lub QR.

W przypadku wykorzystania dekompozycji Choleskiego złożoność obliczeniowa algorytmu regresji liniowej wyniesie $O(Np^2+p^3)$, gdzie N to ilość obserwacji, a p to liczba zmiennych objaśniających.

Regresja liniowa posiada również bardzo dobre własności pod kątem użycia w przypadku strumieni danych i regresji na oknie danych, gdyż pozwala na aktualizację modelu w czasie proporcjonalnym do ilości zmiennych objaśnianych i nie wymaga przetrzymywania danych historycznych.

Poniżej prezentowany jest opis algorytmu szybkiej regresji "IMSR"[21], charakteryzującego się powyższymi własnościami.

Załóżmy, że mamy dane pochodzące z dwóch okresów - okresu do $[0,T]$ i z okresu od $[T+1,T+k]$. Dane z pierwszego okresu dotyczące zmiennych objaśniających znajdują się w macierzy $\mathbf{X_T}$ i reprezentują one wiedzę posiadaną dotychczas, natomiast dane z okresu drugiego znajdują się w macierzy $\mathbf{X_K}$ (reprezentują one napływające dane). Cały zbiór danych składający się z $\mathbf{X_T}$ i $\mathbf{X_K}$ znajduje się w $\mathbf{X}$ (reprezentuje on zbiór danych w momencie jego aktualizacji w chwili $T+k$. Łatwo można pokazać, że:

$$\mathbf{X'X} = \mathbf{X}'_T\mathbf{X}_T + \mathbf{X}'_K\mathbf{X}_K. \tag{2.5}$$

Podobnie jeżeli $\mathbf{y_T}$ będzie zawierać zmienną objaśnianą z pierwszego okresu, $\mathbf{y_K}$ dane napływające, a $\mathbf{y}$ dane połączone, wtedy:

$$\mathbf{X}^T\mathbf{y} = \mathbf{X}_T^T\mathbf{y_T} + \mathbf{X}_K^T\mathbf{y_K}. \tag{2.6}$$

W przypadku aktualizacji modelu nie trzeba wykonywać w każdym kroku kosztownej obliczeniowo operacji mnożenia macierzy $\mathbf{X'X}$ - wystarczy obliczyć ją jeden raz, a następnie aktualizować o iloczyn $\mathbf{X}_K^T\mathbf{X}_K$. Podobnie w przypadku iloczynu macierzy $\mathbf{X}^T\mathbf{y}$.

1. ang. best linear unbiased estimator - najlepszy nieobciążony liniowy estymator

Drugą zaletą tego podejścia jest fakt, iż nie jest konieczne przetrzymywanie wszystkich informacji historycznych.

W przypadku regresji na oknie należy zachować macierze którymi aktualizuje się $\mathbf{X}^T\mathbf{X}$ i $\mathbf{X}^T y$, by po określonej liczbie kroków odjąć je w celu usunięcia zawartej w nich informacji. Ilość macierzy które musimy aktualnie przechowywać jest równa długości okna.

Złożoność obliczeniowa w takim przypadku wynosi $O(Kp^2+p^3)$, gdzie K oznacza ilość obserwacji przekazanych do aktualizacji modelu, przy czym $K < N$.

2.2. Estymator największej głębi regresyjnej i jego własności

Metoda majwiększej głębi regresyjnej (NGR) cechuje się bardzo dobrymi własnościami pod względem odporności, to jest niewrażliwości na obserwacje odstające, bądź nakładanie się lub mieszanie mechanizmów generujących dane[18].

Dany jest zbiór obserwacji $\mathbf{Z_N} = \{(\mathbf{x_1}, y_1), ..., (\mathbf{x_N}, y_N)\} \subset \mathbb{R}^{p+1}$, gdzie $\mathbf{x_i} = (x_{1,i}, ...x_{p,i})$ oznacza i-ty wektor zmiennych objaśniających o liczbie współrzędnych p, natomiast y_i i-tą zmienną objaśnianą. N jest to ilość obserwacji w próbie. Celem jest dopasowanie hiperpłaszczyzny afinicznej w $\mathbb{R}^{p+1}$, która $g((\mathbf{x_i}, 1)\beta^T) = \beta_1 x_{i,1} + ... + \beta_p x_{i,p} + \beta_{p+1}$, gdzie $\beta = (\beta_1, ..., \beta_{p+1})$

By zdefiniować estymator NGR należy wpierw wprowadzić pojęcie niedopasowania:

Definicja 1. *Wektor $\beta = (\beta_1, ..., \beta_{p+1}) \in \mathbb{R}^{p+1}$ nazwiemy niedopasowaniem do $\mathbf{Z_n}$ jeżeli istnieje hiperpłaszczyzna afiniczna V w przestrzeni X, taka, że żadne $\mathbf{x_i}$ nie należy do V i reszta $r_i(\beta) = y_i - g((\mathbf{x_i}, 1)\beta^T > 0$ dla wszystkich x_i leżących w jednej z jej otwartych półprzestrzeni, oraz $r_i(\beta) = y_i - g((x_i, 1)\beta^T < 0$ dla wszystkich x_i w drugiej otwartej półprzestrzeni.*

Definicja 2. *Głębia regresyjna $rdepth(\beta, \mathbf{Z_n})$ dopasowania $\beta = (\beta_1, ..., \beta_{p+1}) \in \mathbb{R}^p$ względem zbioru danych $\mathbf{Z_n} \subset \mathbb{R}^{p+1}$ jest najmniejszą liczbą obserwacji, które należy usunąć aby sprawić by wektor β był niedopasowaniem w sensie definicji 1.*

Jako estymator największej głębi regresyjnej definiuje się β takie, że:

$$T_r^*(\mathbf{Z_n}) = max\{rdepth(\beta, \mathbf{Z_n})\}. \tag{2.7}$$

2.2.1. Punkt załamania próby skończonej

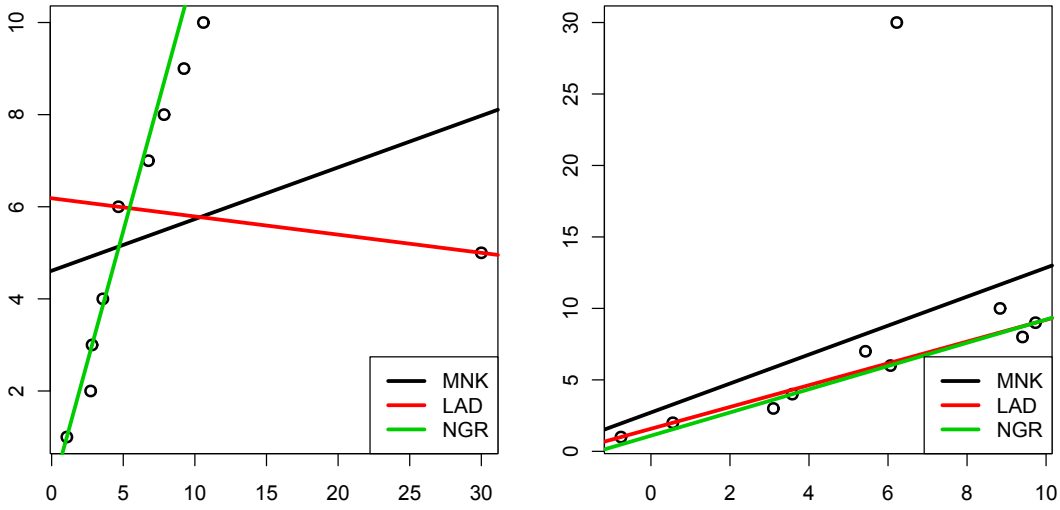
Jako punkt załamania próby skończonej definiuje się frakcję obserwacji która sprawia, że estymator staje się bezużyteczny, na przykład jego obciążenie jest zbyt duże.

Punkt załamania próby skończonej (BP - break point) dla estymatora NGR, przy założeniu, że $argmaxrdepth(\beta, \mathbf{Z_n}) \geq \frac{n}{p+1}$ i x_i są w ogólnej pozycji[2] wynosi $\frac{1}{p+1}$ w przypadku

2. Zbiór punktów znajduje się z ogólnej pozycji, gdy nie zachodzi w nim przypadek by co najmniej trzy punkty leżały na jednej prostej.

gdy dane są osobliwe w jakiś sposób. Można pokazać, że gdy dane pochodzą z założonego modelu wartość BP zbiega z prawdopodobieństwem 1 do wartości $\frac{1}{3}$.

Warto nadmienić, że BP dla estymatora LAD [3], uznawanego pierwotnie za metodę odporną, będącego pewną alternatywą dla NGR wynosi 0 [24]. Estymator ten chroni przed odstawaniem w zakresie zmiennej objaśnianej, jednak nie chroni przed odstawaniem w zakresie zmiennej objaśniającej[4]. Na wykresach 2.2a i 2.2b zaprezentowano porównanie dla oszacowań MNK, LAD i NGR w przypadku występowania obserwacji odstających.



(a) Obserwacja odstająca ze względu na zmienną objaśniającą.

(b) Obserwacja odstająca ze względu na zmienną objaśnianą.

Rysunek 2.2: Oszacowanie MNK, NGR i LAD w przypadku występowania obserwacji odstających

Źródło: Obliczenia własne - R Project

W obu przypadkach estymator MNK „załamuje się", jego oszacowanie zależy w znacznym stopniu od obserwacji odstającej, natomiast LAD jak zostało wspomniane wyżej jest odporny na obserwację odstającą w zmiennej objaśnianej. W drugim przypadku jego oszacowanie jest zaburzone. Natomiast estymator NGR jest odporny na oba przypadki odstawania.

2.2.2. Złożoność obliczeniowa metody największej głębi regresyjnej

W przypadku dwuwymiarowym złożoność obliczeniowa najlepszego algorytmu wyznaczania dopasowania metodą NGR wynosi $O(n \log n)$[1], w przypadku wyższych wymiarów

3. ang. Least Absolute Deviations - metoda najmniejszych odchyleń absolutnych.
4. ang. leverage points

wyznaczenie dokładnego dopasowania wymaga $O(n^{2p-1}\log n)$ operacji, gdzie p to ilość wymiarów. Taka złożoność obliczeniowa w zasadzie dyskwalifikuje algorytm w przypadku większych zbiorów danych - dla przykładowego zbioru danych mającego 10^4 obserwacji w 5 wymiarach, oszacowanie parametrów modelu regresji liniowej metodą NGR wymaga wykonania aż 10^{36} operacji, co przekracza możliwości obliczeniowe znanych komputerów.

Przy pomocy algorytmu przybliżonego MEEDSWEEP[1] oszacowanie NGR można uzyskać w czasie $O(p^2N + hpN + pn\log N)$, gdzie h oznacza liczbę iteracji algorytmu. Złożoność algorytmu MEEDSWEEP nie odbiega w znaczącym stopniu od złożoności algorytmu MNK.

W symulacjach przedstawionych w dalszej części pracy wykorzystano implementację algorytmu największej głębi regresyjnej, pochodzącą z pakietu „depthproc" środowiska R[19].

2.3. Regresja K-najbliższych sąsiadów

Estymator regresji K-najbliższych sąsiadów (KNN[5]) [10] jest metodą nieparametryczną, to znaczy, że nie czyni się założeń odnośnie modelu generującego dane. Jego główną zaletą jest prostota konstrukcji i niska złożoność obliczeniowa.

Dana jest próba $\mathbf{Z_N} = \{(\mathbf{x_1}, y_1), ..., (\mathbf{x_N}, y_N)\} \subset \mathbb{R}^{p+1}$, gdzie $\mathbf{x_i} = (x_{1,i}, ...x_{p,i})$ oznacza i-ty wektor zmiennych objaśniających o liczbie współrzędnych p, natomiast y_i i-tą zmienną objaśnianą, N jest to ilość obserwacji w próbie.

Prognozę wartości zmiennej objaśnianej $\mathbf{y_j}$ metodą KNN na podstawie wektora zmiennych objaśniających $\mathbf{x_j}$ wyznacza się jako:

$$\hat{\mathbf{y}}_\mathbf{j} = \frac{\sum_{i=1}^{N} y_i J(\mathbf{x_i}, \mathbf{x_j})}{K}, \tag{2.8}$$

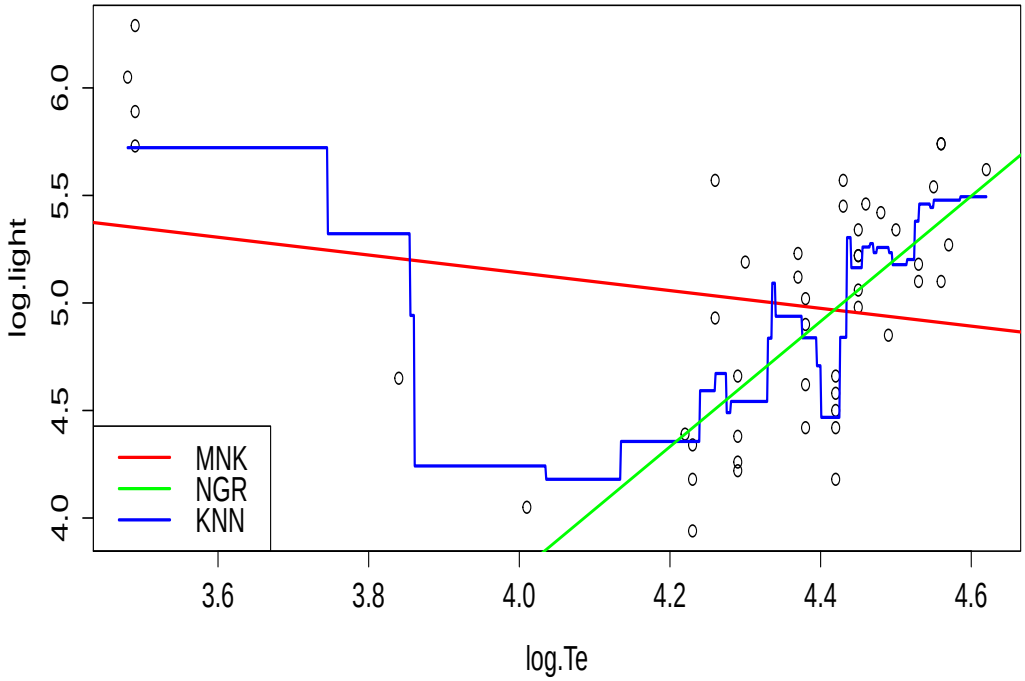
gdzie:

$$J(\mathbf{x_i}, \mathbf{x_j}) = \begin{cases} 1, \text{ gdy } \mathbf{x_i} \text{ jest jednym z K najbliższych sąsiadów } \mathbf{x_j} \\ 0, \text{ w przeciwnym przypadku} \end{cases}, \tag{2.9}$$

natomiast $\mathbf{x_i}$ jest jednym z K najbliższych sąsiadów $\mathbf{x_j}$, w przypadku gdy odległość $d(\mathbf{x_i}, \mathbf{x_j})$ należy do K najmniejszych odległości pomiędzy obserwacjami ze zbioru $\mathbf{Z_n}$, a $\mathbf{x_j}$. Najczęściej wykorzystywane odległości to odległość Euklidesa lub Mahalanobisa, jednak do znalezienia najbliższych sąsiadów można używać koncepcji głębi danych.

W przypadku zastosowania odległości Euklidesa złożoność obliczeniowa regresji KNN wynosi $O(KNp)$, gdzie K to liczba sąsiadów, N to ilość obserwacji w zbiorze $\mathbf{Z_N}$, natomiast p, to ilość zmiennych objaśniających.

5. ang. K-nearest neighbors - K-najbliższych sąsiadów



Rysunek 2.3: Oszacowanie regresji KNN, MNK i DRM dla zbioru starsCYG z pakietu robustbase, w przypadku KNN użyto 5 najbliższych sąsiadów.

Źródło: Obliczenia własne - R Project, pakiet robustbase [25]

Regresja KNN nie jest metodą odporną, jednak w przypadku odstawania pod względem zmiennej objaśniającej może radzić sobie lepiej niż regresja MNK (rys. 2.3).

Również dla zbiorów danych o skomplikowanej strukturze (np. w przypadku zależności nieliniowych) regresja KNN da lepsze wyniki niż metody liniowe (rys. 2.4).

Szerszy opis metod i zastosowań regresji nieparametrycznych można znaleźć w pracy[28].

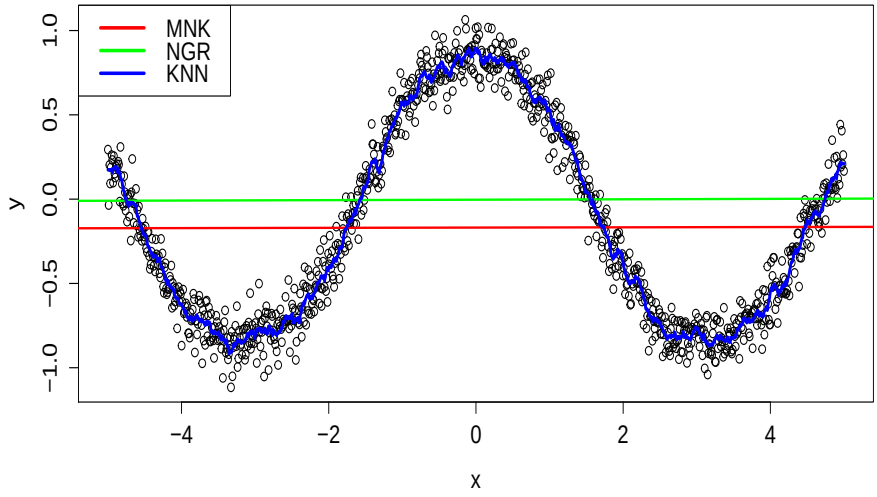
2.4. Odporność procedury statystycznej w przypadku ruchomego okna

Należy jednak zauważyć, że w pewnych zastosowaniach odporność procedury statystycznej może nie zawsze być cechą pożądaną. W dalszej części przedstawiono krótkie rozumowanie przedstawiające kiedy taka sytuacja może zaistnieć.

Przypuśćmy, iż monitorujemy pewien strumień danych $\{y_t, t \in \mathbb{N}\}$, a celem jest przewidywanie następnej obserwacji w oparciu o ruchome okno $W_{t,20}$. Prognoza następnej wartości wyznaczana jest na podstawie modelu:

$$y_t = \alpha + \beta \cdot t, \tag{2.10}$$

gdzie t oznacza moment w czasie. Parametry α i β estymowane są przy użyciu metod MNK



Rysunek 2.4: Oszacowanie regresji KNN, MNK i DRM dla zbioru danych o zależności $y = sin(cos(x)) + \epsilon$, $\epsilon \sim N(0, 0.1)$.

Źródło: Obliczenia własne - R Project

i NGR na podstawie obserwacji pochodzących z okna $W_{t,20}$. Na poniższych rysunkach (rys. 2.5, rys. 2.6) czerwonym kolorem zaznaczono obserwacje, które w danej chwili używane są do estymacji. W pewnej ustalonej chwili T = 41 następuje zmiana procesu generującego dane - pierwotnie pochodziły one z:

$$y_t = 2 + t + \epsilon. \tag{2.11}$$

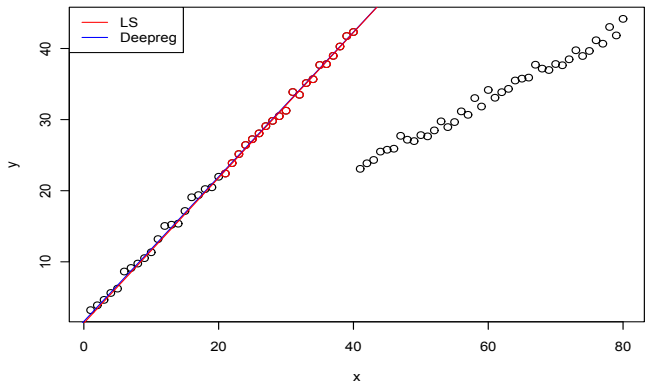
Po zmianie:

$$y_t = 1 + 0.5 \cdot t + \epsilon, \tag{2.12}$$

gdzie $\epsilon \sim N(0, 0.1)$.

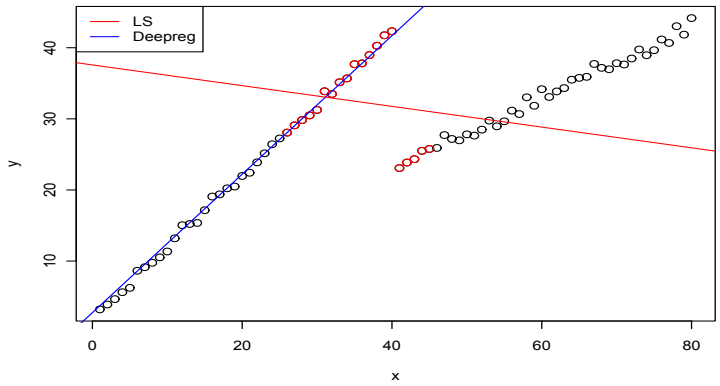
Na wykresie 2.5 przedstawione są oszacowania regresji metodą najmniejszych kwadratów (regresja nieodporna) i największej głębi regresyjnej (metoda odporna) tuż przed zmianą reżimu w oparciu o okno $W_{40,20}$. W tym przypadku wskazania obu estymatorów są bardzo zbliżone.

Wykres 2.6 przedstawia oszacowanie już po zmianie reżimu w którym 5 obserwacji (25% okna) pochodzi z drugiego reżimu.



Rysunek 2.5: Oszacowanie przed zmianą reżimu

Źródło: Obliczenia własne - R Project



Rysunek 2.6: Oszacowanie po zmianie reżimu, część obserwacji (15) pochodzi ze starego modelu generującego dane, 5 - z nowego.

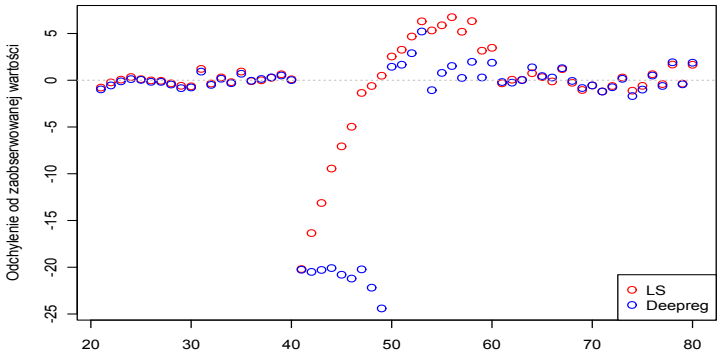
Źródło: Obliczenia własne - R Project

MNK przesunęła się w stronę nowych obserwacji, natomiast NGR nadal wskazuje na poprzedni model. Na rysunku 2.7 przedstawiono odchylenia predykcji następnej obserwacji od faktycznie zaobserwowanej wartości, wyznaczane jako:

$$\hat{\epsilon}_t = y_t - \hat{E}(y_t|W_{t-1,k}), \tag{2.13}$$

gdzie $\hat{\epsilon}_t$ to ocena odchylenia w chwili t, y_t - wartość strumienia w chwili t, $\hat{E}(y_t|W_{t-1,k})$ to prognoza warunkowej wartości oczekiwanej, wyznaczona na podstawie modelu 2.10 estymowanego w oparciu o okno $W_{t-1,k}$, k to ilość obserwacji w oknie, w tym przypadku 20.

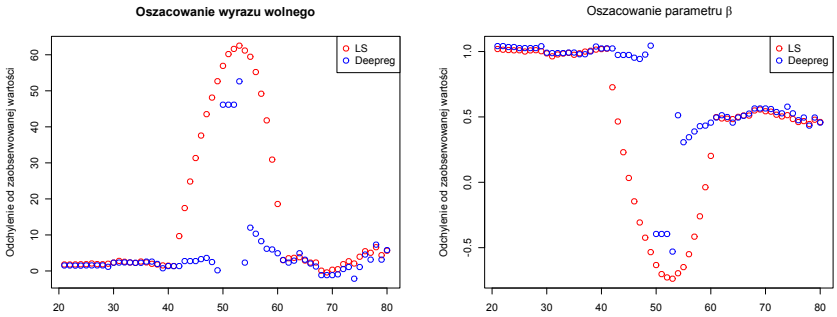
Należy zwrócić uwagę iż w przypadku MNK wraz z pojawianiem się nowych obserwacji błąd się zmniejsza, natomiast w przypadku NGR w pewnym momencie następuje gwałtowna zmiana z dużej wartości odchylenia od rzeczywistej wartości, do bliskiej zeru.



Rysunek 2.7: Różnica pomiędzy przewidywaną obserwacją a zaobserwowaną w przypadku zmiany reżimu.

Źródło: Obliczenia własne - R Project

Na kolejnych dwóch wykresach (rys. 2.8a i rys. 2.8b) zaprezentowano oszacowania parametrów modelu na kolejnych oknach. Należy zwrócić uwagę, iż w również w przypadku NGR w pewnym momencie obliczone parametry znacznie odbiegały od obu modeli.



(a) Oszacowanie wyrazu wolnego

(b) Oszacowanie parametru nachylenia

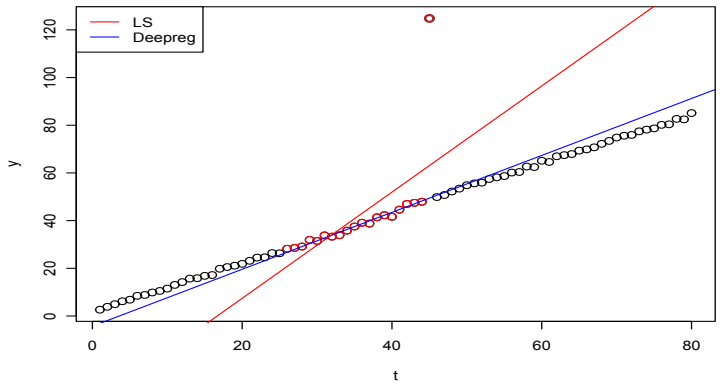
Rysunek 2.8: Oszacowania parametrów MNK i NGR w przypadku zmiany reżimu

Źródło: Obliczenia własne - R Project

Powyższy przykład wskazuje, że w pewnych sytuacjach brak odporności procedury może być zaletą - błędy związane z przewidywaniem następnej obserwacji były mniejsze w przypadku MNK.

Dla zachowania kompletności rozumowania należy również przedstawić zachowanie się obydwu procedur w przypadku gdy cecha odporności jest pożądana, to jest gdy pojawiają się obserwacje odstające.

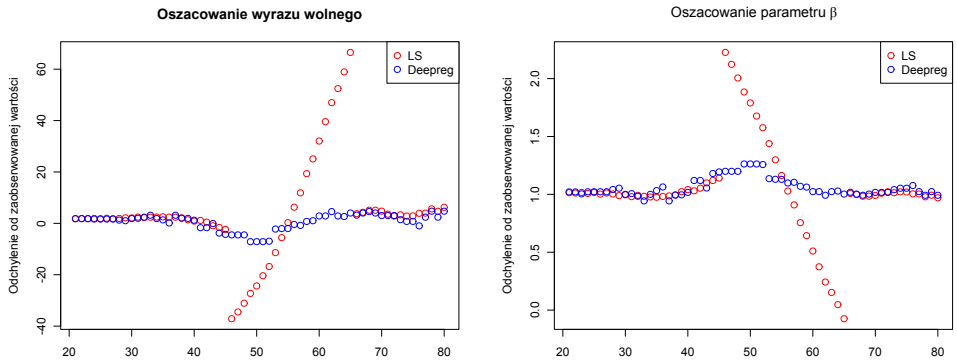
Na rysunku 2.9 przedstawiono krzywą regresji w momencie pojawienia się obserwacji odstającej (na wykresie zaznaczona brązowym kolorem, tak jak powyżej punkty użyte do estymacji są czerwone):



Rysunek 2.9: Krzywa regresji w przypadku pojawienia się obserwacji odstającej.

Źródło: Obliczenia własne - R Project

Wykresy 2.10a i 2.10b prezentują oszacowania parametrów modelu dla kolejnych okien w przypadku występowania obserwacji odstającej, liczone według wzoru 2.13. Oszacowania parametrów MNK przez okres w którym obserwacja odstająca znajduje się w oknie używanym do estymacji modelu, znacznie odbiegają od procesu generującego dane. Natomiast w przypadku NGR nie występuje żadna istotna zmiana.

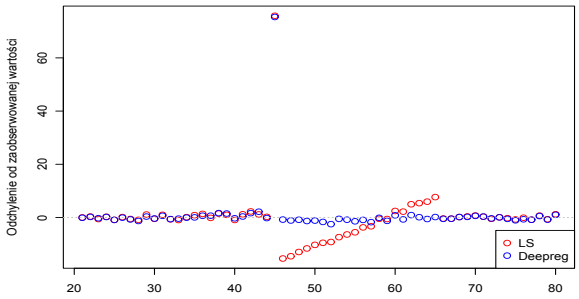


(a) Oszacowanie wyrazu wolnego (b) Oszacowanie parametru nachylenia

Rysunek 2.10: Oszacowania parametrów MNK i NGR w przypadku pojawienia się obserwacji odstającej

Źródło: Obliczenia własne - R Project

Rysunek 2.11 przedstawia różnicę pomiędzy zaobserwowaną wartością a przewidywaną na podstawie poprzedniego okna. Obserwacja odstająca zaburza wskazania modelu MNK, aż do momentu, w którym zostaje wyłączona z okna. Z tego powodu błąd przez pewien okres kształtuje się na znacznie wyższym poziomie niż dla metody NGR.



Rysunek 2.11: Różnica pomiędzy przewidywaną obserwacją a zaobserwowaną w przypadku pojawienia się obserwacji odstającej

Źródło: Obliczenia własne - R Project

Należy zwrócić uwagę, iż w przypadku zmiany reżimu i obserwacji odstającej pojawienie się nowej obserwacji prowadziło do zmniejszenia się się błędu w przypadku MNK, natomiast dla NGR dla zmiany reżimu przez pewien czas odchylenie utrzymywało się na wysokim poziomie, a dla obserwacji odstającej na niskim i nie odbiegało od poprzednich wartości.

Rozdział 3

Własności zaproponowanych metod

3.1. Analizowany problem

Analizowanym problemem będzie przewidywanie warunkowej wartości oczekiwanej na podstawie okna k ostatnich obserwacji, w przypadku strumienia danych o dwóch reżimach. Dodatkowo analizę przeprowadzono w przypadku występowania obserwacji odstających.

3.2. Model analizowanego strumienia danych

Jako model strumienia danych przyjęto proces CHARME[1]. CHARME jest to ogólna struktura do opisu szeregów czasowych ze zmiennymi reżimami, za które można przyjąć wiele procesów liniowych i nieliniowych, takich jak AR [2], GARCH[3], czy SV[4].

W modelu CHARME ukryty łańcuch Markowa Q_t ze skończoną ilością stanów $1, 2, ..., K$ opisuje dynamikę kształtowania się procesu X_t zdefiniowanego jako:

$$X_t = \sum_{k=1}^{K} S_{tk}(m_k(X_{t-1}, ..., X_{t-p}) + \sigma((X_{t-1}, ..., X_{t-p})\epsilon_t) + b_t\Theta, \tag{3.1}$$

gdzie $S_{tk} = 1$ gdy $Q_t = k$ i $S_{tk} = 0$ w przeciwnym przypadku, oraz m_k, σ_k, $k = 1,..K$ to nieznane funkcje, $\epsilon \sim iid$ z wartością oczekiwaną równą 0, natomiast $b_t\Theta$ odpowiada za pojawianie się obserwacji odstających, b_t jest nieobserwowalną zmienną losową o rozkładzie dwupunktowym, Θ jest obserwacją odstającą.

W badaniach symulacyjnych przyjęto dwa reżimy w postaci modeli $AR(1) \sim GARCH(1,1)$ o parametryzacji:

$$x_{t+1} = \mu + \theta x_t + Z_t, \tag{3.2}$$

1. ang. Conditional Heteroscedastic Autoregressive Mixture of Experts
2. ang. Autoregressive Model - model autoregresji
3. ang. Generalized Auto-Regressive Conditional Heteroskedasticity model - uogólniony model autoregresji z heteroskedastycznością warunkową
4. ang. Stochastic Volatility - model wariancji stochastycznej

$$Z_t = \sigma_t \epsilon_t, \tag{3.3}$$

$$\sigma_t^2 = c_0 + \alpha Z_{t-1} + \beta \sigma_{t-1}. \tag{3.4}$$

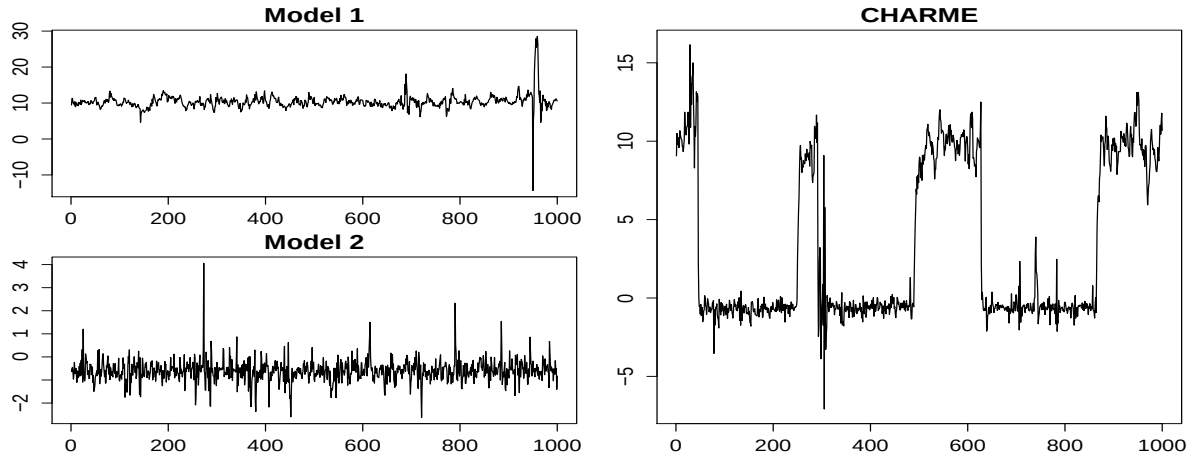
Parametry modelu:

Reżim 1: $\mu = 2$, $\theta = 0.8$, $c_0 = 0.1$,$\alpha = 0.1$, $\beta = 0.75$,

Reżim 2: $\mu = -0.5$, $\theta = 0.2$, $c_0 = 0.1$, $\alpha = 0.6$, $\beta = 0.1$.

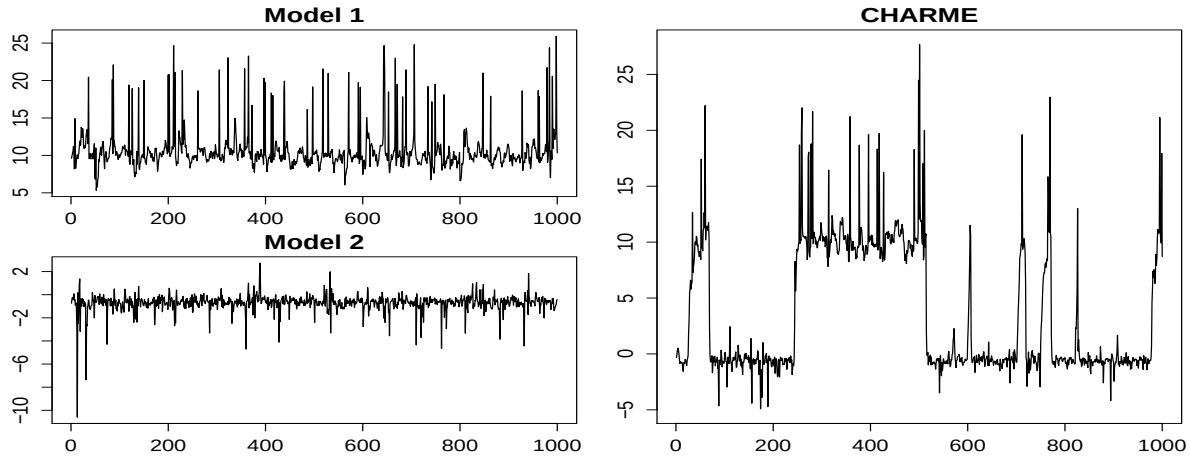
W obu przypadkach ϵ_t to rozkład t-studenta o czterech stopniach swobody.

Rysunki 3.1 i Rysunki 3.2 przedstawiają przykładowe trajektorie procesu CHARME zdefiniowanego w powyższy sposób.



Rysunek 3.1: Przykładowe trajektorie dla procesów wchodzących w skład analizowanego modelu CHARME i samego modelu bez obserwacji odstających

Źródło: Obliczenia własne - R Project



Rysunek 3.2: Przykładowe trajektorie dla procesów wchodzących w skład analizowanego modelu CHARME i samego modelu z obserwacjami odstającymi (5%)

Źródło: Obliczenia własne - R Project

3.3. Wyniki symulacji w przypadku braku występowania obserwacji odstających

W tabeli 3.1 zebrano parametry oceniające jakość użytych modeli. Zostały one wyznaczone symulacyjne na podstawie 500 trajektorii modelu CHARME.

Zdefiniowane zostały jako:

$$RMSE = \frac{\sum_{t=s}^{T}(E(x_t|\Psi_{t-1}) - \hat{E}(x_t|W_{t-1,k}))^2}{T-s}, \tag{3.5}$$

$$ME = \frac{\sum_{t=s}^{T}(E(x_t|\Psi_{t-1}) - \hat{E}(x_t|W_{t-1,k}))}{T-s}, \tag{3.6}$$

$$MAE = \frac{\sum_{t=s}^{T}|E(x_t|\Psi_{t-1}) - \hat{E}(x_t|W_{t-1,k})|}{T-s}, \tag{3.7}$$

$$max(\hat{\epsilon}) = max_{t\in(s,\dots,T)}(E(x_t|\Psi_{t-1}) - \hat{E}(x_t|W_{t-1,k})), \tag{3.8}$$

$$min(\hat{\epsilon}) = min_{t\in(s,\dots,T)}(E(x_t|\Psi_{t-1}) - \hat{E}(x_t|W_{t-1,k})), \tag{3.9}$$

gdzie:

- T - ilość obserwacji w szeregu. W symulacjach $T = 1000$.
- s - moment w którym wyznaczono pierwszą prognozę, wcześniejsze obserwacje zostały użyte jako pierwsze okno do wyznaczania modelu. W symulacjach $s = 101$.
- Ψ_t - historia procesu do chwili t.
- $W_{t,k}$ - okno danych, k oznacza długość okna, w symulacjach $k = 100$.
- $E(x_t|\Psi_{t-1})$ - wartość oczekiwana warunkowa procesu w chwili t wyznaczona analitycznie na podstawie parametrów reżimu w którym proces aktualnie się znajdował.
- $\hat{E}(x_t|W_{t-1,k})$ - prognoza wartości oczekiwanej warunkowa w chwili t wyznaczona na podstawie okna $W_{t-1,k}$.

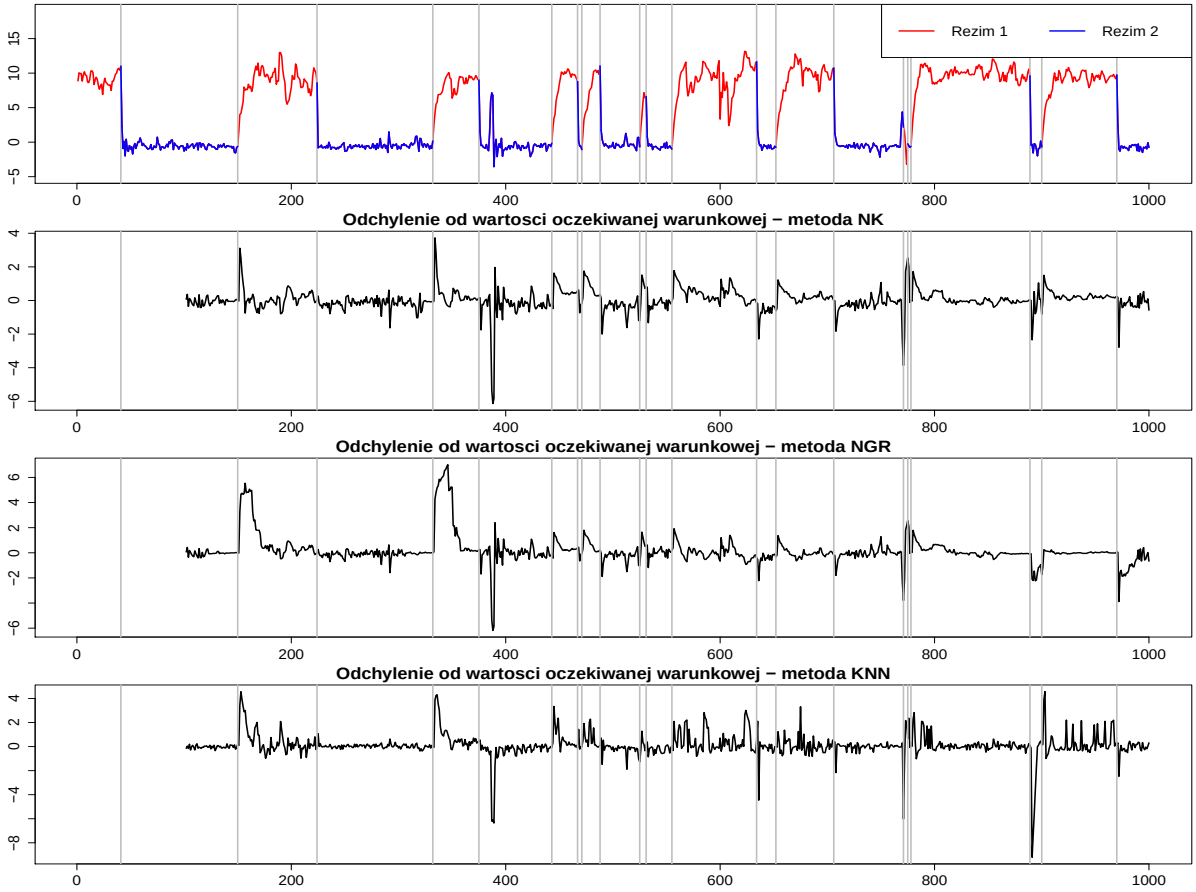
Najlepszą metodą okazała się regresja najmniejszych kwadratów. Należy jednak zwrócić uwagę na to, że również w jej przypadku nastąpiły największe odchylenia od prawdziwej wartości oczekiwanej warunkowej.

Tablica 3.1: Podstawowe parametry oceniające jakość użytych modeli

Metoda	RMSE	ME	MAE	$max(\hat{\epsilon})$	$min(\hat{\epsilon})$
Najmniejsze kwadraty	0.501	0.0112	0.253	19.63	-21.994
Metoda NGR	1.099	0.1665	0.430	15.67	-14.673
KNN(5) - Euklides	0.8577	0.0534	0.385	14.12	-11.29

Źródło: Obliczenia własne - R Project

W dalszej analizie zbadano zachowanie się poszczególnych algorytmów w momencie zmiany reżimu. Na poniższym wykresie 3.3 szarą pionową linią zaznaczono moment zmiany reżimu.

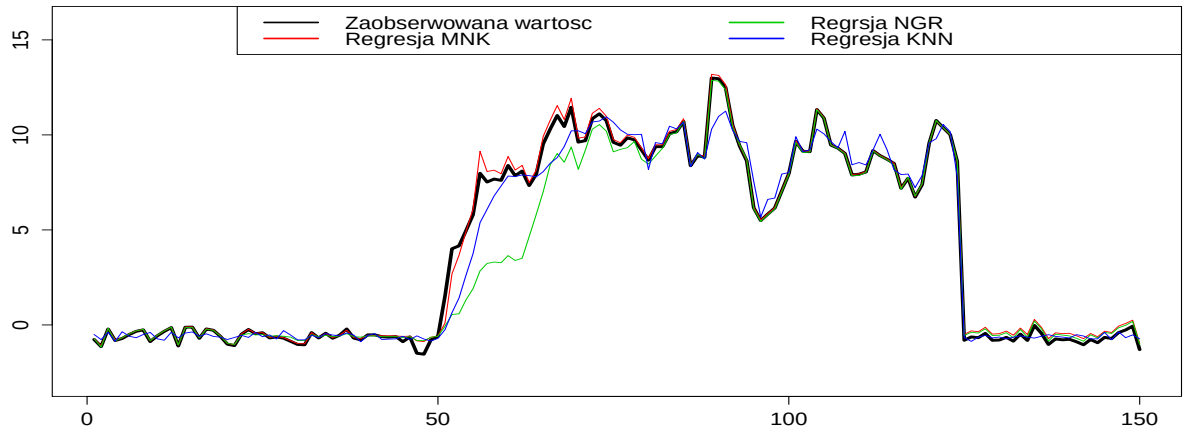


Rysunek 3.3: Odchylenia od wartości oczekiwanej warunkowej dla przykładowej trajektorii modelu CHARME

Źródło: Obliczenia własne - R Project

W powyższym przypadku można zaobserwować zjawisko opisane w 2.4. Otóż przez pewien okres, po zmianie reżimu w przypadku regresji NGR pojawiające się obserwacje

traktowane były jako odstające, przez co, wskazania były bardziej związane z poprzednim reżimem niż z nowym. Natomiast regresje nieodporne znacznie szybciej przesunęły się w stronę nowego reżimu. Obserwacja ta sugeruje, iż w przypadku regresji na oknie danych dla procesu o zmiennych reżimach, przy braku występujących obserwacji odstających powinno raczej używać się metod nieodpornych, gdyż są one w stanie szybciej przejść ze starego reżimu do nowego. Na wykresie 3.4 przedstawiono fragment przykładowej trajektorii procesu CHARME w momencie zmiany reżimu, wraz z krzywymi przedstawiającymi predykcję wartości oczekiwanej wyznaczoną na podstawie poprzedzającego ją okna.



Rysunek 3.4: Zestawienie wartości przewidywanych na podstawie ostatniego okna i faktycznej trajektorii procesu.

Źródło: Obliczenia własne - R Project

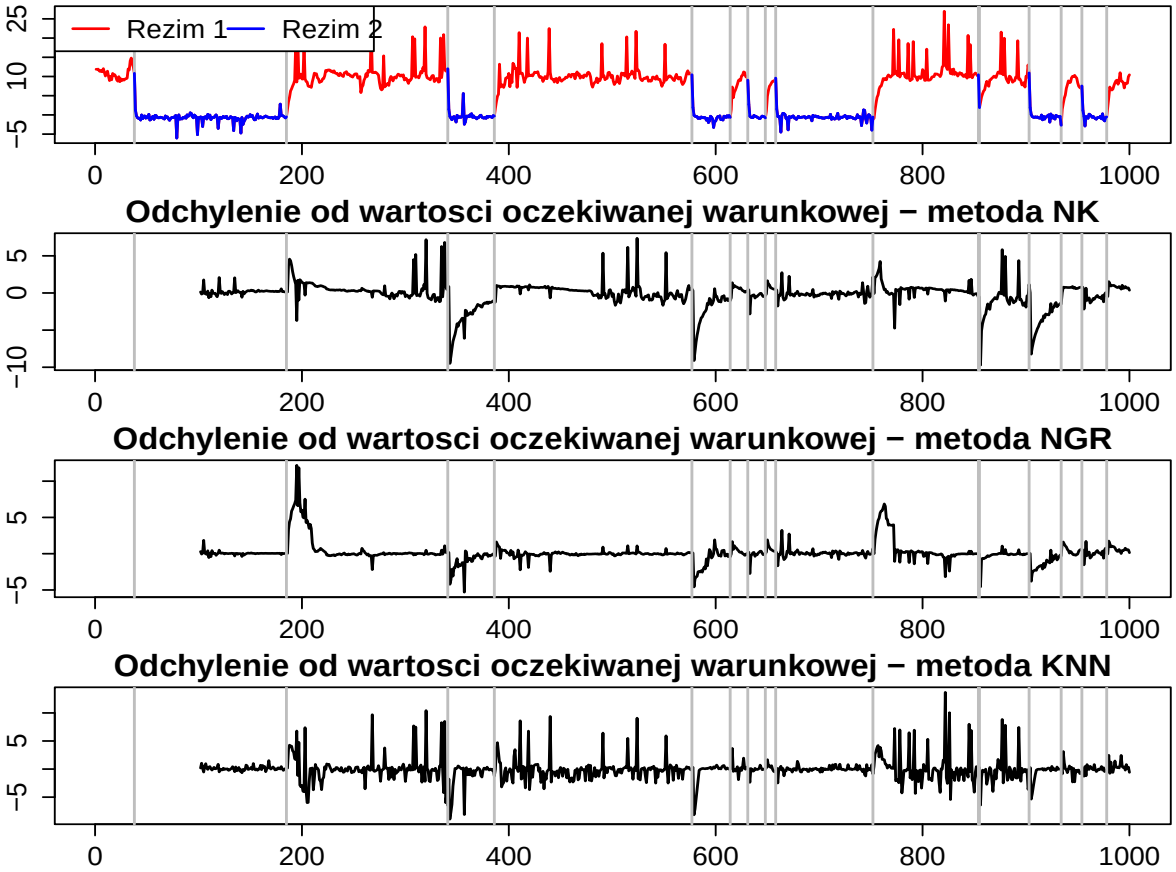
3.4. Wyniki symulacji w przypadku występowania obserwacji odstających

W tabeli 3.2 zebrano podstawowe parametry oceniające jakość użytych modeli. W przypadku występowania obserwacji odstających najlepszym estymatorem wartości oczekiwanej warunkowej okazała się metoda największej głębi regresyjnej. Wzrost błędu predykcji spowodowany późniejszą reakcją na zmianę reżimu miał znacznie mniejsze znaczenie dla oszacowania niż pojawiające się obserwacje odstające.

Tablica 3.2: Podstawowe parametry oceniające jakość użytych metod w przypadku występowania obserwacji odstających

Metoda	RMSE	ME	MAE	$max(\hat{\epsilon})$	$min(\hat{\epsilon})$
Najmniejsze kwadraty	5.011	-0.490	2.759	131.32	-232.525
Metoda NGR	2.133	0.252	0.800	74.904	-85.286
KNN(5) - Euklides	6.041	-0.110	2.435	121.299	-46.15

Rysunek 3.5 przedstawia kształtowanie się odchyleń od wartości oczekiwanej przy założeniu pojawiania się obserwacji odstających.



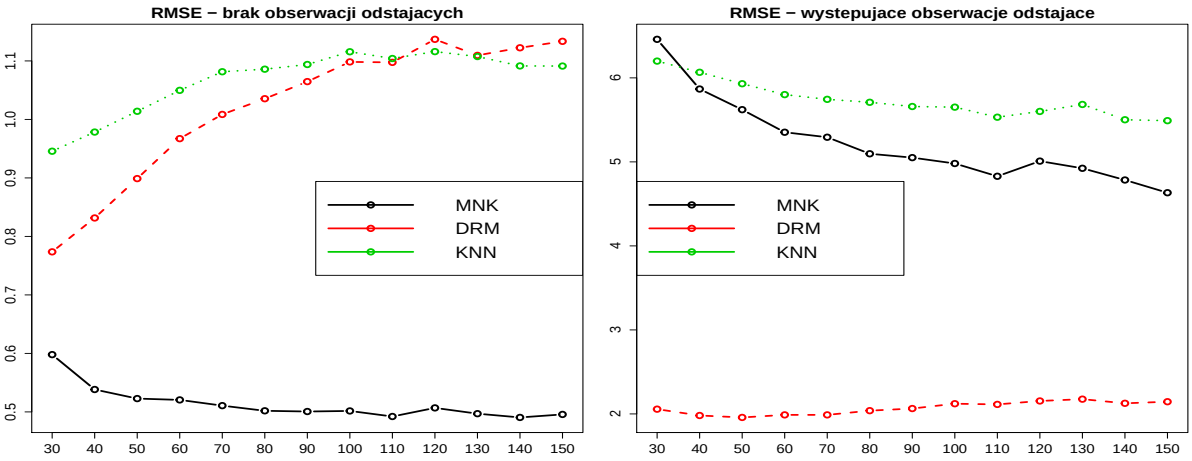
Rysunek 3.5: Odchylenia od wartości oczekiwanej warunkowej dla przykładowej trajektorii modelu CHARME

Źródło: Obliczenia własne - R Project

3.5. Długość okna

W przypadku analizy strumienia danych przy pomocy okna należy ustalić jego długość. W tym kontekście pojawia się dylemat, czy ustalone okno ma być krótkie tak, by

uwzględniać jedynie najbardziej aktualny stan strumienia, czy przeciwnie - powinno być długie, by lepiej lepiej reprezentować strumień w okresie stabilności. W dalszej części przeprowadzono analizę porównawczą omawianych algorytmów z uwagi na długość okna danych. Dla algorytmu KNN ilość najbliższych sąsiadów wyznaczana była jako jedna dziesiąta długości okna. Na rysunku 3.6 przedstawiono średni błąd RMSE w zależności od długości okna.



Rysunek 3.6: RMSE dla modeli dla różnych długości okna

Źródło: Obliczenia własne - R Project

W przypadku braku obserwacji odstających, dla regresji MNK rozmiar okna nie miał kluczowego znaczenia - dla rozmiaru powyżej 70 obserwacji błąd utrzymywał się na zbliżonym poziomie, jednak w przypadku regresji metodą największej głębi regresyjnej im okno było krótsze tym odchylenie od zaobserwowanych wartości było mniejsze. W przypadku występowania obserwacji odstających dla metody NGR wybór długości okna nie wpływał znacząco na oszacowania, natomiast dla regresji MNK wraz z wydłużeniem się okna błąd malał.

Rozdział 4

Przykład empiryczny - RegTrend - prosta strategia typu Trend Following

W dzisiejszym świecie bardzo dużą rolę w handlu giełdowym odgrywa handel algorytmiczny. Decyzja o kupnie lub sprzedaży określonego aktywa zostaje podjęta na bazie określonej strategii inwestycyjnej z wyłączeniem udziału człowieka. W takim podejściu możliwe jest monitorowanie dziesiątek tysięcy aktywów z różnych giełd światowych i wyszukiwanie okazji do transakcji, których człowiek nie jest w stanie dostrzec.

Jako przykład zastosowania modelu strumienia danych i prostej regresji dla handlu algorytmicznego skonstruowano prostą strategię typu Trend Following [1], wykorzystującą regresję na oknie. Strategii nadano nazwę RegTrend. Następnie przeprowadzono proces sprawdzania jej użyteczności.

4.1. Opracowanie sygnału generującego zdarzenie

Pierwszym etapem w tworzeniu strategii algorytmicznej jest przygotowanie sygnału (lub zestawu), który będzie określał jakie działanie ma zostać podjęte. Poniżej znajduje się lista przykładowych operacji:

- zajęcie pozycji na aktywie - czy w danym momencie należy kupić (zająć długą pozycję) lub sprzedać (zająć krótką pozycję) dany aktyw, lub koszyk aktywów ,
- Stop Loss - czy należy w danym momencie zamknąć pozycję z powodu, że aktualna strata przekracza ustalony wcześniej poziom,
- Take Profit - czy zamknąć pozycję, gdyż został osiągnięty oczekiwany zysk,
- powiększenie pozycji - czy należy powiększyć daną pozycję na aktywie.

W przypadku analizowanej strategii RegTrend, sygnał otwarcia pozycji opierał się na współczynniku nachylenia regresji na oknie ostatnich 40 cen zamknięcia dla danego instrumentu. Jeżeli był on dodatni zajmowana zostawała była pozycja długa (dany aktyw był kupowany), jeżeli ujemny - krótka (sprzedaż aktywa). W przypadku zmiany znaku

1. Trend Following - strategia mająca na celu podążanie za trendem w którym aktualnie znajduje się dany instrument

tego współczynnika pozycja była obracana (przejście z pozycji krótkiej na długą i odwrotnie). Do zamykania pozycji zastosowano również mechanizm Trailing Stop polegający na tym, że w przypadku zmiany ceny korzystnej ze względu na zajętą pozycję, poziom na jakim zostaje wykonane zlecenie stop loss podnosi się. Pozwala to zachować osiągnięty zysk, w przypadku możliwych dalszych niekorzystnych zmian na rynku. Stop Loss został ustawiony na poziomie 5%.

W przypadku wystąpienia zlecenia Stop Loss chwilowo zostawało także skracane okno na jakim była wyznaczana regresja do 10 ostatnich obserwacji. Wynika to z przekonania autora, że w przypadku, gdy zajęta pozycja przynosi dużą stratę, w ostatnim krótkim czasie nastąpiła gwałtowna zmiana sytuacji rynkowej (np. głębokie załamanie). Należy więc ograniczyć się do ostatniej, znacznie krótszej historii.

Dodatkowym mechanizmem ograniczającym ostateczną możliwą stratę jest wyłączanie strategii w przypadku, gdy aktualny kapitał spadł do poziomu poniżej 80% kapitału początkowego. To podejście ma na celu ujęcie w pewien sposób faktu, że o ile w przypadku testu na danych historycznych widzi się całą trajektorię przedstawiającą wynik strategii, to dla aktualnych danych dalsza trajektoria nie jest znana, a wraz z rosnącą stratą rośnie przekonanie, że należy interweniować manualnie i taką strategię wyłączyć. Ma to również znaczenie w przypadku funduszy inwestycyjnych i budowania "track record'u", czyli historii wyniku finansowego. Jeżeli w początkowym okresie fundusz zrealizował dużą stratę ma mniejsze szanse na znalezienie kolejnych inwestorów.

4.2. Backtest - analiza wyników

Następnym etapem analizy strategii jest przeprowadzenie „backtestu". Polega on na zastosowaniu danej strategii na danych historycznych dla określonej grupy aktywów. Jako porównanie dla RegTrend przyjęto strategię „Buy-and-hold"[2] polegającą na zakupie danego aktywa na początku okresu i sprzedaży na końcu.

Rysunek 4.1 przedstawia wyniki strategii dla spółki Google. Wykres zawiera trzy krzywe:

Balance - jest to stan środków na koncie w danej chwili wraz z wartością otwartych pozycji po cenie otwarcia liczone według wzoru 4.1,

Equity - jest to aktualny balans skorygowany o aktualne zyski bądź straty na pozycjach. Krzywa equity prezentuje stan środków, jaki znalazłby się na koncie, jeśli w danej chwili zostałyby zamknięte wszystkie otwarte pozycje po aktualnej cenie rynkowej. Krzywa Equity została wyznaczona na podstawie wzoru 4.4.

Buy-and-hold - przedstawia krzywą Equity dla strategii „Buy-and-hold".

2. ang. Buy and Hold - Kup i Trzymaj

$$B_t = K_0 + \sum_{i=1}^{N} R_i, \tag{4.1}$$

gdzie K_0 to kapitał początkowy, N to ilość zamkniętych pozycji, to chwili t włącznie, natomiast R_i to zysk z i-tej pozycji obliczany według formuły 4.2 dla długiej pozycji lub 4.3 dla pozycji krótkiej.

$$R_i = (P_c(1-C) - P_o(1+C)) \cdot \frac{K_o}{P_o(1+C)}, \tag{4.2}$$

$$R_i = (P_o(1-C) - P_c(1+C) \cdot \frac{K_o}{P_o(1+C)}, \tag{4.3}$$

gdzie P_o, to cena zamknięcia dla instrumentu finansowego w dniu w którym wystąpił sygnał otwarcia pozycji, natomiast P_c, to cena instrumentu w dniu w którym pozycja została zamknięta. C jest to wysokość prowizji, w tym przypadku wynosi ona 0.02%. K_o to wolne środki pieniężne w chwili w dniu otwarcia pozycji, iloraz $\frac{K_o}{P_o(1+C)}$ oznacza zrealizowany wolumen w chwili otwarcia (dopuszczalne są części ułamkowe akcji).

$$E_t = B_t + PnL_t, \tag{4.4}$$

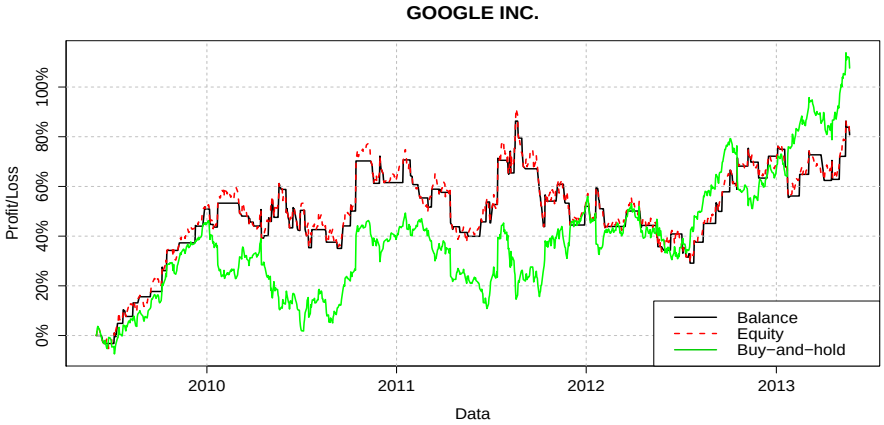
gdzie E_t to wartość Equity w chwili t, B_t wartość Balance wyznaczona ze wzoru 4.1, PnL_t - niezrealizowany zysk lub strata z otwartej pozycji w chwili t. Jeżeli nie ma otwartej żadnej pozycji wtedy $B_t = E_t$.

Wartości Balance, Equity dla RegTrend i „Buy and Hold" wyznaczone ze wzorów 4.1 i 4.4 wyrażone są w pieniądzu. By ułatwić interpretację wykresów i uniezależnić przedstawione wyniki od kapitału początkowego wartości te zostały przeskalowane według wzoru 4.5, by przedstawiały procentowa zmiana względem kapitału początkowego przeznaczonego na strategię.

$$S_t\% = \frac{L_t}{K_0} - 1, \tag{4.5}$$

gdzie L_t to wartość Balance lub Equity w chwili t, natomiast K_0 to kapitał początkowy przeznaczony na strategię.

W tym przypadku strategia daje wyniki zbliżone do strategii „Buy and Hold".



Rysunek 4.1: Krzywe Balance, Equity dla strategii RegTrend zostosowanej dla spółki Google, porównane ze strategią „Buy-and-hold"

Źródło: Obliczenia własne - R Project

W dalszej części przeprowadzono analizę dla szerszego spektrum aktywów (2093 instrumentów pochodzące z NASDAQ). Wymaganiem by dany aktyw został dołączony do testu był jedynie fakt, czy posiada on wystarczająco długą historię (czy był notowany od dnia 6 maja 2009 roku, do 22 maja 2013). Poniżej przedstawiono wykres (rys. 4.2) analogiczny jak w przypadku wyniku strategii dla spółki Google. Prezentuje on sumaryczny wynik dla strategii dla całej gamy aktywów. Wartości Balance i Equity w przypadku portfela aktywów została wyznaczona na podstawie wzorów 4.6 i 4.7.

$$B_t = \sum_{j=1}^{M}(K_{j,0} + \sum_{i=1}^{N_j} R_{ji}), \tag{4.6}$$

$$E_t = B_t + \sum_{j=1}^{M} PnL_{jt}, \tag{4.7}$$

gdzie M to ilość aktywów w portfelu (w analizowanym przypadku $M = 2093$), inne oznaczenia są analogiczne jak w przypadku 4.1 i 4.4, z tą różnicą, że odnoszą się do wartości związanych z j-otym instrumentem, Dla każdego instrumentu kapitał początkowy był równy ($K_{1,0} = K_{2,0} = K_{M,0}$). Również wartości Equity i Balance zostały przeskalowane podobnie jak w przypadku 4.5.

Rysunek 4.2: Krzywe Balance, Equity dla strategii RegTrend zastosowanej dla szerokiego rynku, porównane ze strategią „Buy-and-hold".

Źródło: Obliczenia własne - R Project

Dla zestawu aktywów RegTrend okazał się znacznie gorszy od prostej strategii "Buy-and-hold". W analizowanym okresie, przy założeniu, że w każdy aktyw inwestowana byłaby taka sama kwota, RegTrend przyniósłby 13% stratę (3.42% średniorocznie), natomiast „Buy-and-hold" zarobiłby 108% (20.1% średniorocznie).

Przypadek, w którym strategia systematycznie traci (krzywa Balance jest nachylona ujemnie) oznacza, że założenia leżące u podstawy konstrukcji strategii są najprawdopodobniej nieprawdziwe. W takiej sytuacji użyteczne może być przeprowadzenie backtestu dla odwróconego sygnału. W pierwotnej wersji pozycja długa była zajmowana, gdy współczynnik nachylenia regresji był dodatni, natomiast w wersji odwróconej jest to sygnał do zajęcia pozycji krótkiej, analogicznie w przypadku ujemnego parametru nachylenia. Poziom Stop Loss i mechanizm Trailing Stop pozostały niezmienione.

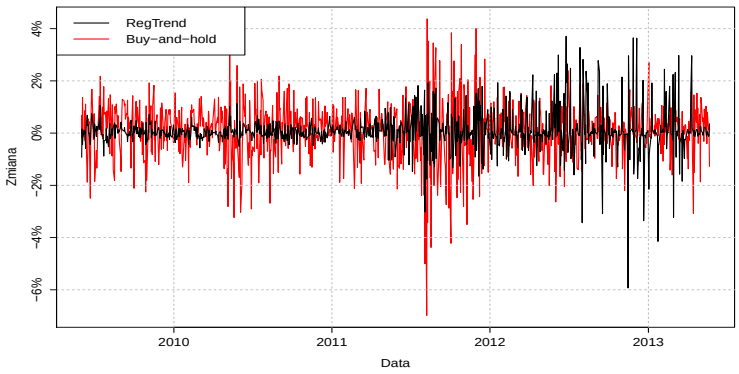
Poniżej (rys. 4.3) prezentowany jest wynik dla strategi RegTrend z odwróconym sygnałem.

Rysunek 4.3: Krzywe Balance i Equity dla odwróconej strategii RegTrend zastosowanej dla szerokiego rynku, porównane ze strategią „Buy-and-hold".

Źródło: Obliczenia własne - R Project

Powyższy wykres (rys. 4.3) wskazuje , że pierwotne założenia odnośnie kontynuacji trendu w analizowanym okresie nie sprawdziły się, jednak sygnał odwrotny dał zadowalający rezultat, od pewnego momentu przewyższający strategię porównawczą. W analizowanym okresie zysk ze strategii wyniósłby 122% (22% średniorocznie)w porównaniu do 108% (20.1% średniorocznie) dla „Buy-and-hold".

Należy jednak zwrócić uwagę, iż w przypadku analizy wyniku backtestu znaczenie ma kształt krzywych Balance i Equity. Może zdarzyć się sytuacja, w której główny zysk pochodzi z pojedynczej transakcji lub z jednego określonego okresu. W przypadku odwróconego RegTrend krzywe Equity i Balance są dosyć gładkie w początkowym okresie, jednak w ostatnim roku zmienność zwrotów drastycznie rośnie (rys. 4.4).

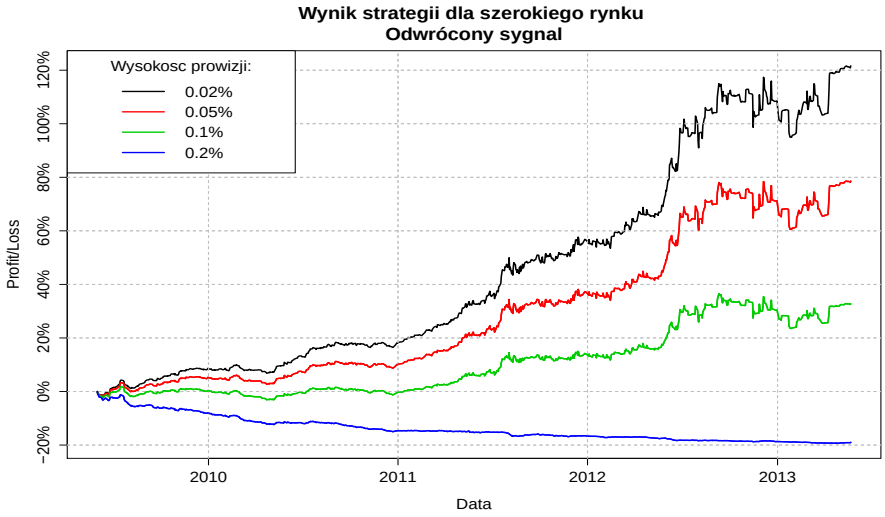


Rysunek 4.4: Krzywe logarytmicznych zmian Balance dla odwróconej wersji RegTrend i strategii "Buy-and-hold".

Źródło: Obliczenia własne - R Project

Dalsza analiza strategii powinna ująć wrażliwość na zmianę kosztów transakcyjnych. W wielu przypadkach są one w stanie diametralnie zmienić wynik, w szczególności jeżeli strategia wykonuje bardzo dużo transakcji.

Poniżej (rys.4.5) prezentowane są krzywe Balance w zależności od kilku wybranych wielkości prowizji:



Rysunek 4.5: Krzywe Balance w przypadku różnych wysokości prowizji.

Źródło: Obliczenia własne - R Project

Na korzyść strategii przemawia fakt, że nawet pięciokrotne podwyższenie kosztów transakcyjnych nie spowodowało, że strategia zaczęła diametralnie tracić.

4.3. Wnioski

Pierwotnie przyjęte założenia strategii, mówiące że jeżeli występuje dodatnie nachylenie krzywej regresji czasu względem ceny wyznaczonej dla ostatnich dwóch miesięcy, wtedy dany aktyw znajduje się w trendzie wzrostowym, który będzie kontynuowany - należy więc zająć pozycję długą (analogicznie dla ujemnego nachylenia krzywej regresji), okazały się w przypadku rynku NASDAQ nieadekwatne, jednak ich przeciwieństwo dało zadowalające wyniki.

Fakt, iż wyniki odwróconej strategii były zbliżone do rynkowych sugeruje, że reguła decyzyjna polegająca na monitorowaniu współczynnika nachylenia regresji może być użyteczna w handlu algorytmicznym. Dodatkowo podejście oparte na strumieniach danych i oknach, pozwala na monitorowanie bardzo dużej ilości aktywów, co może szczególne znaczenie w kontekście redukcji ryzyka poprzez dywersyfikację portfela.

Rozdział 5

Wnioski końcowe

W pracy zaprezentowano podejście do monitorowania strumieni danych przy pomocy algorytmów prostej regresji: metody najmniejszych kwadratów, największej głębi regresyjnej oraz regresji k najbliższych sąsiadów. Pierwsza metoda wykazała najlepsze własności w przypadku braku występowania obserwacji odstających, jednak jej wskazania znaczenie się pogorszyły wraz pojawienia się nietypowych obserwacji. W tej sytuacji zdecydowanie lepszym algorytmem okazała się metoda największej głębi regresyjnej.

Stosując wyżej wymieniony metody należy mieć na uwadze między innymi fakt, iż dla strumienia o częstych zmianach reżimu cecha odporności określonej procedury statystycznej może być niepożądana, co zostało pokazane w symulacjach. W takim przypadku lepsze efekty można uzyskać poprzez skrócenie okna na którym wyznaczany jest model.

Na szczególną uwagę zasługuje techniczny aspekt analizy wielkich zbiorów danych jakim jest dostępność wygodnych narzędzi ułatwiających prowadzenie obliczeń równoległych. Przykładowo w pakiecie R, by móc skorzystać z większej ilości rdzeni procesora wystarczy zaledwie kilka linijek dodatkowego kodu. Jednocześnie najbardziej wydajne techniki związane z obliczeniami nadal wymagają dużej wiedzy i zasobów finansowych.

Spis rysunków

Spis tablic

Literatura

[1] S. Aelst, P. Rousseeuw, M. Hubert, and A. Struyf. The deepest regression method. Technical report, Department of Mathematics and Computer Science, U.I.A, 2000.

[2] I. Aldridge. *High-frequency trading.* John Wiley & Sons, Inc., 2010.

[3] F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber. Bigtable: A distributed storage system for structured data. *OSDI '06 Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*, 7:15–15, 2006.

[4] J. Choy. Visualizing data techniques, including auto-charting and big data for sas. *SAS Global Forum*, 2012.

[5] T. Cormen, Ch. Leiserson, and R. Rivest. *Wprowadzenie do algorytmów.* Wydawnictow Naukowo Techniczne, 2001.

[6] Z. Czech. *Wprowadzenie do obliczeń równoległych.* PWN, 2010.

[7] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. Technical report, Google Inc, 2004.

[8] M. Doman and R. Doman. *Modelowanie zmienności i ryzyka.* Wolters Kluwer Business, 2009.

[9] J. Guha. Large complex data: divide and recombine with rhipe. *Journal of the International Statistical Institute*, 1(1), 2012, w druku.

[10] W. Hardle. *Applied nonparametric regression.* Cambridge University Press, 1990.

[11] D. Harel and Y. Feldman. *Rzecz o Istocie Informatyki.* Wydawnictow Naukowo Techniczne, 2008.

[12] A. Heijs. Big data: rethinking text visualization. Technical report, Treparel, 2013.

[13] P. Huber. *Data analysis: what can be learned from the past 50 years.* John Wiley & Sons, 2011.

[14] S. Kairam, D. MacLean, M. Savva, and J. Heer. Graphprism: compact visualization of network structure. *Advanced Visual Interfaces*, 2012.

[15] A. Karbowski and E. Niewiadomska-Szynkiewicz. *Programowanie równoległe i rozproszone.* Oficyna Wydawnicza Politechniki Warszawskiej, 2009.

[16] D. Kirk and W. Hwu. *Programming massively parallel processors.* Pearson Eduction, 2011.

[17] D. Kosiorowski. Głębia położenia-rozrzutu w strumieniowej analizie danych ekonomicznych. *Przegląd Statystyczny - numer specjalny 1 w związku z Kongresem Statystyki Polskiej*, 1:87–108.

[18] D. Kosiorowski. *Wstęp do wielowymiarowej analizy statystycznej zjawisk ekonomicznych.* Wydawnictwo Uniwersytetu Ekonomicznego w Krakowie, 2008.

[19] D. Kosiorowski, M. Bocian, A. Wegrzynkiewicz, and Z. Zawadzki. *depthproc: Package for depths.* R package version 1.0.

[20] M. Lourakis. A brief description of the levenberg-marquardt algorithm implemented by levmar. Technical report, Institute of Computer Science, 2005.

[21] C. Nadungodage, Y. Xia, F. Li, and F. Ge. Streamfitter: A real time linear regression analysis system for continuous data streams. *Database Systems for Advanced Applications*, 6588:458–461, 2012, w druku.

[22] O'Reilly. *Big data now: 2012 Edition.* O'Reilly Media, Inc., 2012.

[23] J. Reese and S. Zaranek. Gpu programming in matlab. Technical report, MathWorks, 2011.

[24] P. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79:871 – 880, 1984.

[25] P. Rousseeuw, Ch. Croux, V. Todorov, A. Ruckstuhl, M. Salibian-Barrera, T. Verbeke, M. Koller, and M. Maechler. *robustbase: Basic robust statistics.* R package version 0.9-7.

[26] J. Sanders and E. Kandrot. *CUDA by example.* Pearson Eduction, 2011.

[27] Ch. Strauch. Nosql databases: a step to database scalability in web environment. *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services.*

[28] Anna Węgrzynkiewicz. Nieparametryczne estymatory jądrowe regresji w zastosowaniach ekonomicznych. *Praca magisterska, Wydział Zarządzania UEK w Krakowie*, 2013. maszynopis.

[29] H. Wickham and Y. Hue. *bigvis: Tools for visualisation of big data sets.* R package version 0.1.