Wybrane estymatory parametrów prostej regresji w przypadku wielkich zbiorów danych

Zygmunt Zawadzki

April 23, 2013

Contents

Abstract

Wprowadzenie

1 Wielkie zbiory danych

1.1 Co to jest wielki zbiór danych

Nie istnieje sztywna granica rozmiaru która definiowałaby zbiór jako wielki. Pewną wskazówką może być klasyfikacja oparta na objętości zbioru danych w bajtach wprowadzona przez Hubera [?]:

- malutki 10^2 bajtów
- mały 10^4 bajtów
- średni 10^6 bajtów
- duży 10^8 bajtów
- ogromny 10^{10} bajtów
- monstrualny 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. Wielki zbiór danych można również opisywać przy pomocy 3V[?]: Volume (rozmiar), Velocity (szybkość napływu nowych informacji), Variety (różnorodność). Rozmiar rozumiany jest tu jako ilość miejsca potrzebnego do przechowywania danego zbioru. Dopisać o velocity i variety

1.2 Problemy związane z analizą wielkich zbiorów danych

1.2.1 Rozmiar zbioru danych w konteście zajmowanej przestrzeni dyskowej

Dopisać

1.2.2 Rozmiar zbioru danych a złożoność obliczeniowa

Podstawowym problemem w przypadku wielkiego zbioru danych jest fakt iż z racji jego wielkości nie można w używać do jego analizy określonych algorytmów gdyż zwyczajnie czas działania takiego programu byłby zbyt duży. By zobrazować problem można odwołać się do prostego rozumowania - jeżeli złożoność algorytmu który ma być użyty do analizy wynosi $O(n^3)$, a zbiór danych ma 10^6 elementów - wtedy ilość operacji do wykonania wynosi około 10^{18} - gdzie najszybszy aktualnie istniejący superkomputer dysponuje mocą obliczeniową rzędu $18 \cdot 10^{15}$ flops (flops - Floating point Operations Per Second, ilość operacji zmienno przecinkowych na sekundę). Czas działania takiego programu w idealnej sytuacji wyniósłby około minuty - przy czym należy zwrócić uwagę, że pomijane są koszty czasowe związane z komunikacją między procesorami i czy w ogóle algorytm mógłby skorzystać z architektury wieloprocesorowej superkomputera.

Natomiast w przypadku zbioru danych określanego jako ogromny według klasyfikacji Hubera (rozmiar 10^10 bajtów - to ok 10^9 elementów licząc że każdy element jest liczbą zmiennoprzecinkową podwójnej precycji według standartu IEEE zajmującą 8 bajtów) algorytm o złożoności czasowej $O(n^3)$ potrzebowałby około 1700 lat obliczeń z pełną wydajnością. Nawet w przypadku wzrostu wydajności superkomputerów do poziomu 1 eksaflopsa (10^{18} operacji na sekundę), czas wykonania wyniósłby około 30 lat - należy zaznaczyć, że wprowadzenie superkomputerów o tej wydajności szacuje się na rok około 2019.

Huber odnosząc się do kwestii złożoności obliczeniowej algorytmów używanych do analizy wielkich zbiorów danych stwierdza, że powinna być ona niższa niż $O(n^{\frac{3}{2}})$, w innym przypadku można w zasadzie zapomnieć o takim algorytmie [?].

Jednym ze sposobów radzenia sobie z wielkością problemu jest strategia "Dziel i zwyciężaj" polegająca na podzieleniu zadania na mniejsze podzadania, rozwiązaniu ich i połączeniu ich wyników w jeden ostateczny wynik. Takie podejście do problemu ułatwia zastosowanie technik programowania równoległego znacznie skracając czas wykonywania się algorytmu.

Zastosowanie techniki "Dziel i zwyciężaj" bardzo łatwo można zademonstrować na przykładzie mnożenia macierzowego. Mając 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. Czas wykonywania obliczeń wyniesie $n \cdot m \cdot p$ - łatwo jednak zauważyć, że obliczanie elementu $c_{i,j}$ w żaden sposób nie zależy od obliczania innych elementów macierzy C. W tym przypadku stosując metodę "dziel i zwyciężaj" podzadaniem będzie

wyznaczanie elementu $c_{i,j}$. 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.

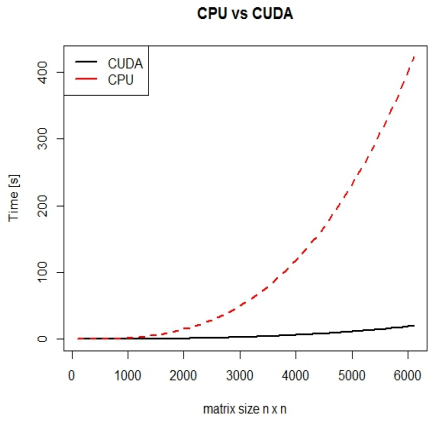

Figure 1: Tutaj muszę jeszcze zakodzić te testy

Warto również zwrócić uwagę na rozwijający się dynamicznie sektor obliczeń prowadzonych na GPU (Graphical Procesor Unit). Poniższy wykres pokazuje czas wykonywania się algorytmu mnożenia macierzy na karcie graficznej NVidia GT640 z zastosowaniem technologi CUDA (więcej na temat obliczeń na GPU można znaleźć w [?][?]):

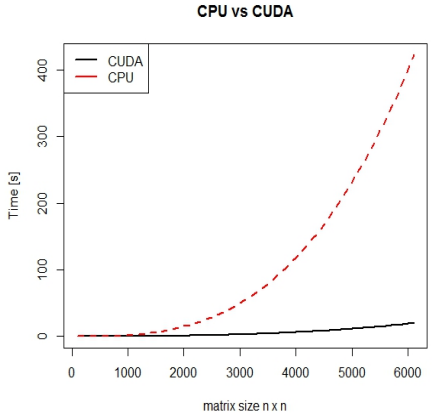

Figure 2: Tutaj też muszę zakodzić

W przypadku obliczeń równoległych istnieje jednak górne ograniczenie określające maksymalny stopień przyspieszania który można uzyskać używając wielu procesorów. Niech $S(p, n)$ oznacza przyspieszenie uzyskane dla problemu wielkości n przy użyciu p procesorów, s niech oznacza sekwencyjną proporcję programu, wtedy:

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

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 [?].

Dodatkowym problemem związanym używaniem algorytmów opartych o przetwarzanie równoległe jest również trudność w ich implementacji - by w pełni skorzystać z możliwości które daje architektura wieloprocesorowa trzeba zapoznać się z wieloma aspektami dość niskopoziomowymi z punktu widzenia analityka, takimi jak zarządzanie pamięcią, czy komunikacja między procesorami - w przeciwnym przypadku zysk uzyskany ze zrównoleglenia algorytmów może okazać się niewielki.

1.2.3 Szybkość napływania danych

Szybkość napływu informacji jest drugim z możliwych problemów który może pojawić się w konteście analizy wielkich zbiorów danych. Najprostszym przykładem bardzo dynamicznego zbioru danych są dane giełdowe, gdzie ceny transakcyjne i informacje o zmianach na książce zleceń danego instrumentu pojawiają się w bardzo krótkich interwałach, oczywiście należy wziąć jeszcze pod uwagę ilość obserwowanych spółek giełdowych. Bardzo często może się więc okazać że czas potrzebny do aktualizacji danego modelu znacznie przekracza okres do pojawienia się nowych danych.

W przypadku danych wysokiej częstotliwości jednym z możliwych rozwiązań jest zbieranie danych w pakiety, a dopiero po osiągnięciu określonego rozmiaru, lub po określonym czasie, aktualizacja modelu. Takie podejście może być jednak nie wystarczające we wszytkich zastosowaniach - firma IBM zwraca uwagę, że jest bardzo niebezpieczne przechodzenie przez ulicę jeżeli otrzymuje się jedynie zdjęcie pokazujęce jak wyglądał ruch 5 minut wcześniej. Podejście to szczególne znaczenie w przypadku handlu wysokiej częstotliwości (HFT - High Frequency Trading), gdzie kluczowe znaczenie mają jak najbardziej aktualne dane, gdyż pozycje zawierane są na bardzo krótki okres - nawet poniżej 1 sekundy [?].

Tutaj trzba dopisać o strumieniach.

1.2.4 Struktura zbioru

Struktura wielkiego zbioru danych może być bardzo skomplikowana - przez co samo przygotowanie go do analizy może okazać się bardzo czasochłonne.[?] [1] Tutaj jeszcze do napisania - problemy z wizualizacją, problemy z jakością danych, przetwarzanie danych .

[1] "I probably spend more time turning messy source data into something usable than I do on the rest of the data analysis process combined."