Optymalizacja w Analizie Danych - algorytm Least Angle Regression (LARS)

Piotr Janus, Paulina Tomaszewska

sem. zimowy 2019/2020

1 Wstęp teoretyczny

1.1 Duże zbiory danych

W Analizie Danych często używa się zbiorów danych o dużej liczbie zmiennych objaśniających. Jednak taka sytuacja rodzi wiele problemów w trakcie stosowania modeli uczenia maszynowego:

- wymaga długiego czasu przetwarzania
- zbiór może zawierać zmienne będące szumem (o małej mocy predykcyjnej), które mogą pogarszać jakość predykcji
- może wystąpić przekleństwo wymiarowości (ang. *curse of dimensionality*) - w sytuacji gdy predyktorów jest więcej niż obserwacji

1.2 Modele regularyzowane

W odpowiedzi na problemy omówione powyżej wprowadzono do modeli uczenia maszynowego regularyzację, która polega na tym, że w zadaniu optymalizacji uwzględnia się dodatkowy czynnik ograniczający wartości współczynników odpowiadające predyktorom. Zaproponowano wiele modyfikacji dla zadania regresji, oto kilka z nich [2]:

- regresja grzbietowa

$$\min_{\beta} ||X * \beta - Y||_2^2 + \alpha||\beta||_2^2$$

  Dodatkowy warunek ma na celu minimalizowanie normy l2 wektora współczynników β, co finalnie sprawia, że współczynniki mają małą wartość bezwzględną.

- regresja Lasso

$$\min_{\beta} ||X * \beta - Y||_2^2 + \alpha||\beta||_1$$

W regresji Lasso użycie normy l1 sprzyja temu aby wektor β był rzadki (tzn. miał wiele zer - jest to tożsame z tym że dane predyktory uznane są za nieistotne).

- regresja *Elastic Net*

$$\min_{\beta} \frac{1}{2n_{samples}} ||X * \beta - y||_2^2 + \alpha\rho||\beta||_1 + \frac{\alpha(1-\rho)}{2}||\beta||_2^2$$

Regresja *Elastic Net* łączy w sobie cechy regresji grzbietowej oraz Lasso, poprzez zastosowanie dwóch typów norm, których istotność względem siebie kontroluje parametr ρ.

W przypadku tego projektu punktem wyjściowym do dalszych rozważań będzie regresja Lasso, której celem jest selekcja zmiennych.

1.3 Selekcja zmiennych

Istnieje wiele metod selekcji zmiennych, m.in:

- metoda w przód (ang. *Stepwise Forward Selection*)

  Model zaczyna swoje działanie jako pusty, tzn. bez zmiennych objaśniających. Następnie w kolejnych iteracjach algorytmu dodawana jest zmienna o największej mocy predykcyjnej o ile jest ona większa od ustalonej granicznej wielkości. Metoda ta z uwagi na prostotę jest szybka, jednak może być zbyt zachłanna, ponieważ w każdym kroku dodaje do zbioru aktywnego całą zmienną (nie przypisuje jej żadnego współczynnika). Można więc powiedzieć, że działanie tego algorytmu jest binarne tzn. decyduje tylko czy zmienna jest istotna czy nie - nie ma wariantu pośredniego, czyli wskazującego że dana zmienna jest istotna w pewnym stopniu.

- metoda *Forward Stagewise*

  Metoda ta jest odpowiedzią na ograniczenia metody w przód, ponieważ pozwala na dodawanie zmiennej do zbioru aktywnego w pewnym stopniu tzn. poprzez przypisanie jej pewnego współczynnika. W przeciwieństwie do metody w przód metoda *Stagewise* gdy w danej iteracji stwierdzi, że konkretna zmienna ma największą moc predykcyjną nie dodaje jej w całości do zbioru aktywnego a jedynie zwiększa o ε współczynnik odpowiadający temu predyktorowi. To rozwiązanie, mimo iż prowadzi do poprawnych rozwiązań, wymaga jednak bardzo wielu iteracji, tym samym jest wolne i nieefektywne.

Metody *Stepwise* i *Forward Stagewise* opierają się na podobnym pomyśle tzn. dołączania zmiennych do modelu w oparciu o metrykę. Metoda *Stagewise* daje lepsze wyniki niż *Stepwise* jednak z uwagi na konstrukcję jest algorytmem wolnym. Kompromisem między wydajnością a poprawnością wyniku jest stosowanie metody Lasso. W algorytmie LARS [4] zdecydowano się jeszcze usprawnić

dotychczasowe rozwiązania w kontekście skrócenia czasu ich wykonywania. Poniżej prezentujemy wynik porównania czasu działania opisanych algorytmów w ramach ich gotowych implementacji w języku R [1], które stanowić będą nasz późniejszy punkt odniesienia.

```
Unit: milliseconds
     expr      min       lq      mean    median       uq      max neval
    lasso  50.9882  58.2938  72.14184  63.46715  80.3463 187.8892    50
      lar  43.5313  53.0623  68.48089  59.81220  68.2452 191.1862    50
 stepwise 130.5537 153.5825 189.46449 171.72380 227.0430 329.9308    50
```

Rysunek 1: Benchmark dla 50 wywołań każdego z algorytmów na zbiorze *diabetes*

Duże dysproporcje w czasie wykonywania funkcji wynikają z tego, że LARS wykonał tylko 10 kroków (czyli tyle ile jest zmiennych objaśniających) w każdym wywołaniu funkcji podczas gdy Stagewise aż 6000 [4].

1.4 Least Angle Regression (LARS)

Metoda LARS została wprowadzona w 2004 r. na Uniwersytecie Stanford. Jest ona koncepcyjnie podobna do metody *Stagewise* jednak znacznie szybsza. Zamiast w kolejnych iteracjach algorytmu zwiększać współczynniki o ε, dokonywany jest krok w odpowiednim kierunku o optymalnej długości. Kierunki te są wybierane w taki sposób aby był równy kąt (tym samym korelacja) między wszystkimi zmiennymi w zbiorze aktywnym.

W LARS korelacja jest zdefiniowana w inny sposób niż w ten do którego jesteśmy przyzwyczajeni czyli do współczynnika korelacji Pearsona. W metodzie LARS korelacja wyraża się wzorem:

$$c = X^T * (y - \mu)$$

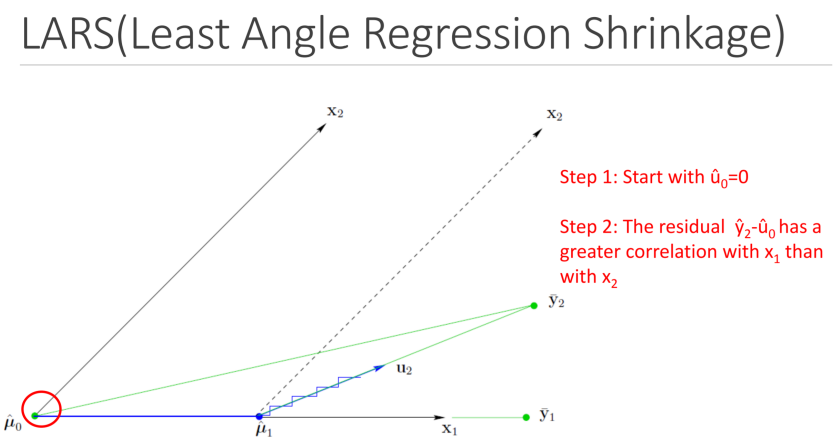
gdzie:
X-wszystkie zmienne objaśniające
y - zmienna objaśniana po odjęciu jej średniej
μ - jest estymatorem zmiennej objaśnianej wyliczanej na podstawie aktualnego wektora współczynników β zgodnie ze wzorem

$$\mu = X^T * \beta$$

Warto zauważyć, że korelacja jest obliczana jako iloczyn skalarny (co z innego punktu widzenia jest iloczynem norm wektorów i kosinusa kąta między nimi). Iloczyn skalarny jest maksymalny gdy kosinus jest maksymalny a ma to miejsce gdy kąt między wektorami jest jak najmniejszy - stąd nazwa metody 'Regresja w oparciu o najmniejszy kąt' (ang. *Least Angle Regression*).

Jak już zostało to podkreślone, w LARS długości kroków dokonywanych w kierunku zmiennych o największej mocy predykcyjnej mają optymalną długość. Przykładowo w drugim kroku algorytmu jest to wyliczane w taki sposób aby

residuum (czyli różnica między predyktorem a jego estymatorem) miało taką samą korelację ze zmienną *x1* jak i *x2* co jest równoznaczne z tym że residuum w kroku poprzednim dzieli kąt między *x1* i *x2* na połowę. Poprzez ten zabieg wraz z dodaniem do modelu kolejnej zmiennej objaśniającej współczynniki β odpowiadające zmiennym obecnym w zbiorze aktywnym są proporcjonalnie zmniejszane.



Rysunek 2: Schemat ilustrujący metodę LARS
`https://medium.com/@phanindra.josh/the-gifted-regressor-lasso-lars-60d78785e2f4`

Algorytm LARS:

1. odejmij średnią od zmiennej objaśnianej
2. inicjalizacja: zbiór aktywny zmiennych objaśniających jest na początku pusty (wektor współczynników β ma same zera), residual jest równy zmiennej objaśnianej
3. znajdź zmienną objaśniającą, która ma największą korelację z residualem (tworzy najmniejszy kąt z residualem)
4. idź w kierunku zmiennej znalezionej w poprzednim punkcie tak długo aż inna zmienna w zbiorze nieaktywnych nie będzie miała takiej samej bądź większej korelacji z residualem
5. sprawdź warunki stopu, jeśli nie są spełnione idź do pkt. 5, w przeciwnym przypadku zakończ
6. dołącz kolejną zmienną, idź w kierunku wyznaczonym przez aktualną i poprzednią zmienną tak długo aż opisano w pkt. 4
7. wróć do pkt. 5

Algorytm jest zatrzymywany w dwóch przypadkach:

- wartość błędu średniokwadratowego jest większa niż w poprzedniej iteracji
- w zbiorze aktywnym są już wszystkie zmienne objaśniające

Metoda LARS daje wyniki zbliżone do Lasso i *Stagewise*. Poprzez drobne modyfikacje w algorytmie LARS można uzyskać dwie wspomniane metody jednak będą one wówczas wyliczane w szybszy sposób niż poprzez korzystanie z ich oryginalnych implementacji. Należy podkreślić, że poprzez efektywne zaimplementowanie metody LARS, koszt obliczeniowy dla wszystkich m kroków jest tego samego rzędu co w przypadku rozwiązania najmniejszych kwadratów dla pełnego zestawu zmiennych objaśniających m [4].

2 Implementacja

Implementacja algorytmu LARS bazująca na oryginalnej publikacji [4] powstała w języku Python. Jednak zastosowano kilka implementacyjnych "sztuczek", które usprawniły wykonywanie algorytmu.

- W ramach algorytmu trzeba odwrócić macierz G_a (macierz pomocnicza w algorytmie), co jest trudne w sytuacji gdy macierz jest osobliwa (wyznacznik jest równy zero, a współczynnik uwarunkowania macierzy bardzo duży). Dlatego zastosowaliśmy manewr w postaci dodania na przekątnej macierzy G_a małego współczynnika regularyzacji $\epsilon = 10^{-12}$. Jest to przykład realizacji regularyzacji Tichonowa.
- W celu znalezienia macierzy odwrotnej nie wyliczano wyznacznika a następnie macierzy minorów lecz rozwiązano układ równań liniowych w oparciu o metodę najmniejszych kwadratów. Zastosowana funkcja *lstsq* z pakietu *numpy.linalg* pozwala na uzyskanie przybliżonego rozwiązania w przypadku gdy analizowana macierz nie jest kwadratowa i pełnego rzędu. Tym samym zastosowana metoda pozwala na znalezienie "macierzy pseudo-odwrotnej" nawet w przypadku gdy macierz wejściowa jest prostokątna. Alternatywnie można było użyć innej funkcji z tego samego pakietu - *pinv*, która jest stricte dedykowana wyznaczaniu (Moore-Penrose) pseudo-odwrotnych macierzy. Metoda ta w przeciwieństwie do poprzedniej nie rozwiązuje układu równań lecz oparta jest o dekompozycję SVD.
- w artykule sprecyzowano metody pozwalające na weryfikację poprawności implementowanej metody [wzór 2.7 z publikacji]:

  1. $$\|\mu\| = 1$$
  2. $$(xa)^T * (ua) > Aa * np.ones((1, n))$$

gdzie:
ua - wektor dzielący kąt między zmiennymi objaśniającymi na pół
xa - zbiór zmiennych objaśniających ze zbioru aktywnego pomnożonych przez współczynniki określające kierunek poszukiwań (+/- 1)
Aa - zmienna pomocnicza

dlatego w naszym kodzie wprowadziliśmy asercje, które sprawdzają powyższe warunki z uwzględnieniem pewnego ε. Jeżeli warunki nie są spełnione, wówczas zatrzymywane jest wykonywanie programu.

Na etapie porównywania efektów działania naszej implementacji i gotowej z pakietu lars w R zauważyliśmy, że dla zbiorów o dużej liczbie zmiennych objaśniających oba te algorytmy generują znacznie różniące się wektory współczynników β. Wnikliwa analiza logów z implementacji w R pozwoliła zauważyć, że wprowadzono tam modyfikację, nie podaną wprost w oryginalnej publikacji algorytmu LARS [4]. Okazało się, bowiem że w sytuacji gdy w kolejnych iteracjach algorytmu kilka zmiennych objaśniających cechuje się taką samą korelacją (z dokładnością do wartości +/- ε), wówczas nie tylko wybierana jest ta o najmniejszym indeksie, ale także pozostałe zmienne o tym samym c są usuwane trwale ze zbioru nieaktywnych. W naszym pierwotnym podejściu nie następowało usuwanie. Jednak interpretacja za usuwaniem tych zmiennych jest taka, że skoro korelacja innej zmiennej jest taka sama, to podążanie na późniejszym etapie algorytmu w tym kierunku nie będzie nic wnosiło. Dlatego też postanowiono takie zmienne usuwać ze zbioru zmiennych-kandydatów, równocześnie takie rozwiązanie przyspieszy działanie algorytmu.

2.1 Modyfikacje transformujące LARS w Lasso

W LARS optymalna długość kroku (γ) to taka, aż inny predyktor ze zbioru nieaktywnych nie będzie miał modułu korelacji równej korelacji którejkolwiek zmiennej ze zbioru aktywnego po pokonaniu odległości γ. W LARS predyktor po przeniesieniu go do zbioru aktywnego już nigdy z niego nie będzie usunięty.

W modyfikacji LARS w celu otrzymania metody Lasso zmianie ulega sposób wyznaczania długości kroku metody [3]. Modyfikacja ta wynika z Lematu: w Lasso niezerowy element j wektora β musi mieć taki sam znak jak korelacja zmiennej objaśniającej j

$$sign(\beta_j) = sign(c_j).$$

Tymczasem korelacja jest funkcją ciągłą, która nie zmienia znaku w trakcie trwania pojedynczej iteracji algorytmu LARS. Dlatego widać sprzeczność z lematem za każdym razem gdy współczynnik wektora β zmienia znak. Warto podkreślić, że w algorytmie LARS analizowana równość znaków nie musi być spełniona.

W Lasso w ramach iteracji idzie się w kierunku zmiennej o największe korelacji tak długo jak w przypadku LARS albo tak długo aż nie zmieni się znak

współczynnika β przy którejkolwiek ze zmiennych ze zbioru aktywnego. Warto podkreślić, że drugi ze wspomnianych warunków także spełnia kryteria optymalności sprecyzowane w LARS. Kiedy długość kroku w Lasso jest opisana poprzez zmianę znaku, wówczas predyktory których współczynnik zmienił znak są ustawiane na zero co jest tożsame z usunięciem danej zmiennej objaśnianej ze zbioru aktywnego (kluczowa różnica w stosunku do LARS). W przypadku gdy w danej iteracji algorytmu następuje usunięcie zmiennej ze zbioru aktywnego nie jest w niej dokonywane przyłączenie do tego zbioru innej zmiennej.

2.1.1 Zarys algorytmu Lasso

Współczynniki wektora β w metodzie LARS są aktualizowane według wzoru:

$$\beta_j(\gamma) = \hat{\beta}_j + \gamma * \hat{d}_j$$

Można więc zauważyć, że zmiana znaku współczynnika nastąpi gdy

$$\gamma_j = -\hat{\beta}_j / \hat{d}_j$$

Wówczas w ramach Lasso kierunek poszukiwań powinien być zmieniony o najmniejszą spośród dodatnich wartość γ_j.
Jeśli γ_j jest mniejsza od optymalnego kroku LARS następuje niespełnienie warunków lematu. Wówczas algorytm usuwa ze zbioru aktywnego zmienną odpowiadająca najmniejszej dodatniej wartości γ_j i wykonuje krok o długości γ_j.
W przeciwnej sytuacji, czyli gdy γ_j jest większa od optymalnego kroku LARS, wówczas algorytm wykonuje krok o dł. odpowiadającej metodzie LARS i w tej iteracji postępuje zgodnie z tym dogmatem.

3 Analiza porównawcza naszej implementacji i gotowych rozwiązań

W celu porównywania naszej implementacji z tą w języku R skorzystaliśmy z możliwości wykonywania skryptów z rozszerzeniem .py w RStudio poprzez użycie pakietu *reticulate*. Analiza porównawcza stanowi załącznik do raportu.

Literatura

[1] Lars w R. `https://www.rdocumentation.org/packages/lars/versions/1.2/topics/lars`. Dostęp: 29-12-2019.

[2] Modele regularyzowane. `https://ksopyla.com/machine-learning/modele-regresji-liniowej-z-scikit-learn/`. Dostęp: 29-12-2019.

[3] Chris Fraley and Tim Hesterberg. Least angle regression and lasso for large datasets. *Statistical Analysis and Data Mining*, 1:251–259, 2009.

[4] Robert Tibshirani, Iain Johnstone, Trevor Hastie, and Bradley Efron. Least angle regression. *The Annals of Statistics*, 32(2):407–499, Apr 2004.