Akademia Górniczo-Hutnicza
Wydział Informatyki
Kierunek Informatyka

Metody Obliczeniowe w Nauce i Technice

Laboratorium 2

Metoda najmniejszych kwadratów

Wojciech Michaluk, Kyrylo Iakymenko

Kraków, 8 marca 2024

1 Wprowadzenie

Podczas tego laboratorium zrealizujemy zadanie, w którym poznamy i wykorzystamy metodę najmniejszych kwadratów. Przeprowadzimy potrzebne obliczenia dla dwóch podejść w ramach tej metody, tzn. *liniowa* i *kwadratowa* metoda najmniejszych kwadratów.

W obu przypadkach wykorzystamy dane ze zbioru treningowego, aby na ich podstawie wyznaczyć charakterystyczne dla metody najmniejszych kwadratów wielkości i użyć ich do predykcji wyników z drugiego zbioru danych - służącego do walidacji. Następnie w każdym z wariantów porównamy otrzymane wyniki z rzeczywistymi, żeby zweryfikować poprawność metody i omówić ewentualne odstępstwa.

2 Opis zadania

W zadaniu należy zastosować metodę najmniejszych kwadratów, aby dokonać predykcji czy nowotwór jest złośliwy, czy łagodny. W zależności od rodzaju nowotworu inne są charakterystyki wzrostu, w tym wartości takich parametrów jak **promień** czy **tekstura**.

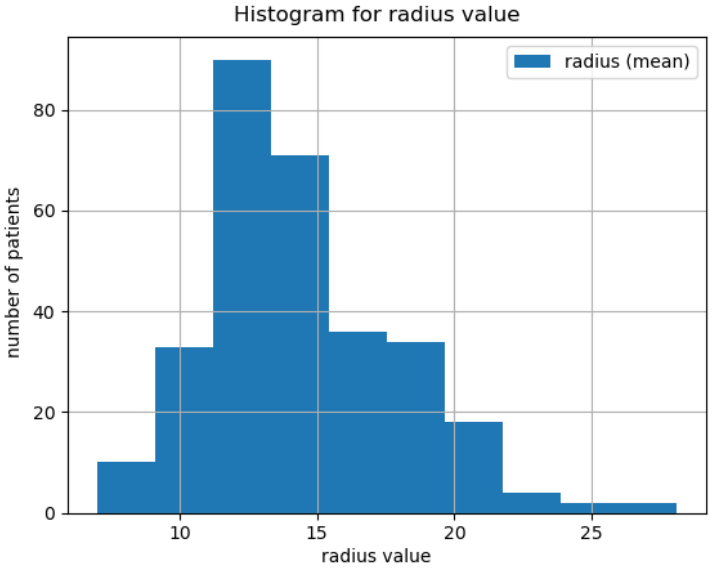
Dane potrzebne do rozwiązania zadania znajdują się w katalogu *dataset*, który został udostępniony w ramach tego laboratorium. Zawiera on pliki:

- **breast-cancer.labels** - opis wszystkich kolumn,
- **breast-cancer-train.dat** - dane treningowe, na podstawie których będziemy budować model,
- **breast-cancer-validate.dat** - dane służące do walidacji modelu i porównania uzyskanych wyników z rzeczywistymi.

Zgodnie z opisem kolumn, pierwszą z nich jest *patient ID* - nie jest ona szczególnie istotna w kontekście zadania, natomiast drugą jest *Malignant/Benign* - to kolumna kluczowa, zawierająca informacje o rodzaju nowotworu (to właśnie ona będzie punktem odniesienia dla uzyskanych wyników). W pozostałych kolumnach znajdują się charakterystyki nowotworu dla każdego z pacjentów.

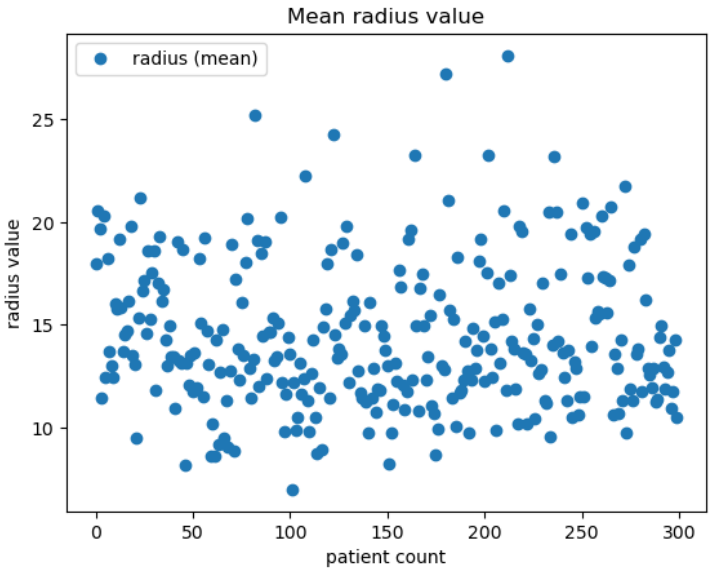
2.1 Czynności wstępne

Najpierw wczytaliśmy dane z plików, używając funkcji `pd.io.parsers.read_csv`, gdzie *pd* oznacza skrót od biblioteki *pandas*. Następnie na podstawie wybranej kolumny danych, używając funkcji *hist* oraz *plot*, rysujemy histogram oraz wykres dla niej. W naszym przypadku wybraną kolumną jest *radius (mean)*. Poniżej przedstawiamy uzyskany histogram.



Rysunek 1: Histogram dla kolumny *radius (mean)* - dane testowe

Teraz z kolei prezentujemy wykres wartości dla tej kolumny.



Rysunek 2: Wykres dla wartości z kolumny *radius (mean)* - dane testowe

W następnych sekcjach przedstawiamy kolejne kroki w metodzie najmniejszych kwadratów, przy czym dokładny opis czynności zawieramy dla metody **liniowej**. Kroki podjęte w *kwadratowej* metodzie są analogiczne. Kolejno będziemy wyznaczać:

1. Macierze reprezentacji danych A dla obu zbiorów
2. Wektor b dla obu zbiorów, zawierający informacje z kolumny *Malignant/Benign*, zakodowany jako 1, jeśli nowotwór jest złośliwy (Malignant) lub -1, jeśli jest łagodny (Benign). Wektor b jest wspólny dla obu metod.
3. Wektor wag w zbudowany na podstawie macierzy A i wektora b ze zbioru testowego, który później będzie użyty w predykcji wyniku dla zbioru walidującego.
4. Współczynnik uwarunkowania macierzy.
5. Wektor predykcji p, zbudowany na podstawie macierzy A dla zbioru **walidującego** i wyliczonego wektora wag w.

3 Liniowa metoda najmniejszych kwadratów

Przy tym podejściu tworzymy reprezentację dla danych ze wszystkich kolumn zawierających charakterystyki nowotworu, tzn. oprócz pierwszych dwóch kolumn.

3.1 Macierz reprezentacji danych

Macierz A tworzymy poprzez kod przedstawiony poniżej, przy czym dane wczytane ze zbioru treningowego to `training_set`, a te ze zbioru walidującego to `validate_set`.

```
training_set_data = training_set.drop(['patient ID',
  'Malignant/Benign'], axis = 1) #dane treningowe
A_linear_training = training_set_data.values

validate_set_data = validate_set.drop(['patient ID',
  'Malignant/Benign'], axis = 1) #dane sprawdzające
A_linear_validate = validate_set_data.values
```

Listing 1: Tworzenie macierzy A

Macierz dla danych treningowych to `A_linear_training`, z kolei macierz dla danych walidujących to `A_linear_validate`.

3.2 Wektor informacji o typie nowotworu

Tworzone tutaj wektory b są używane zarówno dla liniowej, jak i kwadratowej metody najmniejszych kwadratów. Dla danych testowych wektor ten nazywamy `b_training` i zadajemy go kodem:

```
b_training = np.array(training_set[['Malignant/Benign']]).flatten()
b_training = np.where(b_training == 'M', 1, -1)
```

Wektor dla danych walidujących jest tworzony analogicznie:

```
b_validate = np.array(validate_set[['Malignant/Benign']]).flatten()
b_validate = np.where(b_validate == 'M', 1, -1)
```

Listing 2: Tworzenie wektorów b dla obu zbiorów

3.3 Znajdowanie wektora wag

Szukamy wektora wag w, dla którego $Aw \cong b$. Dla takiego wektora będzie spełnione, że $||Aw - b||$ przyjmuje najmniejszą wartość. Zauważamy, że nie jest trywialne wyznaczyć ten wektor, ponieważ macierz A nie musi być kwadratowa (istotnie, w tym przypadku nie jest), zatem użycie funkcji `numpy.linalg.solve` bezpośrednio nie jest możliwe, nie można obliczyć macierzy odwrotnej. Aby poradzić sobie z tą niedogodnością, korzystamy z tzw. *równania normalnego*. Rozważmy najpierw proste przekształcenie:

$$Aw = b \quad | \cdot A^T (\text{od lewej strony}) \tag{1}$$

$$A^T A w = A^T b \tag{2}$$

Stąd widać, skąd się wzięła postać równania normalnego - wygląda ono następująco:

$$w = (A^T A)^{-1} A^T b \tag{3}$$

przy czym macierz $(A^T A)^{-1} A^T$ to jest tzw. *macierz pseudoodwrotna*. Dzięki temu możemy wyznaczyć wektor wag w, ale ten sposób jest szczególnie wrażliwy na współczynnik uwarunkowania macierzy. W naszym programie do obliczenia wektora w korzystamy jednak wprost z zależności (2), aby zgodnie z poleceniem użyć funkcji *solve* - choć jest to równoważne (3).
Kod generujący potrzebne wielkości przedstawiamy poniżej.

```
AT_linear = np.transpose(A_linear_training) #transponowana macierz
w_linear = np.linalg.solve(np.dot(AT_linear, A_linear_training),
  np.dot(AT_linear, b_training)) #wektor wag
```

Listing 3: Znajdowanie wektora wag w, wykorzystując równanie normalne

3.4 Współczynnik uwarunkowania macierzy

Współczynnik ten, oznaczany jako $cond(A)$, normalnie jest wyrażony wzorem

$$cond(A) = ||A|| \cdot ||A^{-1}||.$$

Tak jak wspomnieliśmy w poprzedniej sekcji, macierz A nie jest kwadratowa, więc nie istnieje macierz do niej odwrotna. Musimy zatem użyć macierzy $A^T A$ (która jest kwadratowa), zatem wzór przyjmuje postać

$$cond(A^T A) = ||A^T A|| \cdot ||(A^T A)^{-1}|| \tag{4}$$

Pomiędzy współczynnikiem uwarunkowania policzonym ze wzoru (4) a $cond(A)$ zachodzi relacja

$$cond(A^T A) = cond(A)^2.$$

W naszym programie obliczamy ten współczynnik dwoma sposobami - korzystając z funkcji `numpy.linalg.norm` oraz za pomocą funkcji `numpy.linalg.cond`. Okazuje się, że wyniki różnią się między sobą. Obliczeniom odpowiada następująca sekcja kodu:

```
ATA_linear = np.dot(AT_linear, A_linear_training)
cond_linear_norm = (np.linalg.norm(ATA_linear) *
  np.linalg.norm(np.linalg.inv(ATA_linear)))
cond_linear_cond = np.linalg.cond(ATA_linear)
```

Listing 4: Obliczenie współczynnika uwarunkowania na dwa sposoby

gdzie `ATA_linear` to macierz $A^T A$, a `cond_linear_norm` oraz `cond_linear_cond` oznaczają wartości współczynnika uwarunkowania policzone z wykorzystaniem odpowiednich funkcji. W pierwszym przypadku wynosi ona $2.0781757966471704 \cdot 10^{12}$, natomiast drugi wynik to $1.809248222570959 \cdot 10^{12}$. Widzimy, że nasze zadanie jest źle uwarunkowane. Jeżeli ε byłoby błędem względnym naszych danych, to $\varepsilon \cdot cond(\phi(x))$ oznaczałoby błąd względny naszych wyników. Wtedy dla tak dużych wartości współczynników uwarunkowania macierzy, błąd względny naszej predykcji byłby zbyt duży, żeby móc go wykorzystywać w realnych przypadkach.

3.5 Predykcja typu nowotworu

Wektor predykcji p obliczamy, mnożąc liniowa reprezentację zbioru walidującego przez wektor wag w, wyliczony z równania (2) - odpowiada temu kod:

```
p_linear = np.dot(A_linear_validate, w_linear) #p_linear to wektor predykcji
```

Jeśli $p[i]$ ma wartość dodatnią to przyjmujemy, że osoba ma (prawdopodobnie) nowotwór złośliwy, w przeciwnym wypadku wskazujemy nowotwór łagodny. Najpierw przemapujemy wartości z wektora `p_linear` na odpowiednio 1 dla nowotworu złośliwego i -1 dla nowotworu łagodnego. Następnie porównujemy przemapowany wektor `p_linear` i wektor `b_validate`, co ilustruje kod poniżej:

```
p_linear_check = np.where(p_linear > 0, 1, -1) #przemapowany wektor p_linear
p_linear_compare = np.where(p_linear_check == b_validate, True, False)
#p_linear_compare zawiera wartości logiczne: True, gdy
#predykcja była prawidłowa, False w przeciwnym przypadku
```

Listing 5: Sprawdzenie predykcji typu nowotworu

Wskażemy liczbę wszystkich pacjentów, liczbę poprawnie rozpoznanych typów nowotworów oraz liczbę niepoprawnie rozpoznanych nowotworów, przy czym dokonujemy tu podziału na tzw. *false positives* - kiedy model przewiduje nowotwór złośliwy, a w rzeczywistości był łagodny - oraz *false negatives*, oznaczające sytuację odwrotną. W kodzie wygląda to następująco:

```
#przypadki false positives
p_linear_false_positives = np.where((p_linear_check == 1) & (b_validate == -1))

#przypadki false negatives
p_linear_false_negatives = np.where((p_linear_check == -1) & (b_validate == 1))
```

Listing 6: Sprawdzenie przypadków *false positives* i *false negatives*

Wyniki są przedstawione w poniższej tabeli.

Wszyscy pacjenci	poprawna predykcja	niepoprawna predykcja	false positives	false negatives
260	252	8	6	2

Tabela 1: Statystyki predykcji dla modelu liniowego

4 Kwadratowa metoda najmniejszych kwadratów

Teraz przejdziemy do analizy kwadratowej reprezentacji. Proces dopasowania modelu oraz oceny wiarygodności na podstawie zbioru walidacyjnego wygląda prawie identycznie jak w przypadku liniowym - szczegółowa implementacja wraz z krótkimi opisami znajduje się w pliku *.ipynb*. Tu skupimy się raczej na analizie uzyskanych wyników.

Reprezentacja kwadratowa daje modelowi większą elastyczność w dopasowywaniu się do danych, ponieważ może wyrażać nieliniowe zależności między cechami a klasą nowotworu. Jednakże, może być bardziej skłonna do przeuczenia, zwłaszcza w przypadku małych zbiorów danych.

Na potrzeby klasyfikacji używaliśmy tylko 4 cech w porównaniu do wszyskich 30 dla modelu liniowego. Wyniki klasyfikacji nowotworów przez model kwadratowy przedstawiamy poniżej.

Wszyscy pacjenci	poprawna predykcja	niepoprawna predykcja	false positives	false negatives
260	240	20	15	5

Tabela 2: Statystyki predykcji dla modelu kwadratowego

Porównując te wyniki z reprezentacją liniową, zauważamy, że reprezentacja kwadratowa wydaje się być mniej skuteczna. Liczba niepoprawnie sklasyfikowanych przypadków dla reprezentacji kwadratowej jest ponad dwukrotnie wyższa niż dla reprezentacji liniowej. Powiązane to może być z mniejszą liczbą cech używanych do dopasowania modelu oraz stosunkowo małym rozmiarem danych treningowych (tylko 300 przypadków).

Analiza wyników pokazuje, że reprezentacja kwadratowa jest bardziej podatna na błędy, szczególnie jeśli chodzi o fałszywie dodatnie ("false-positive") przypadki, co może prowadzić do potencjalnych nadinterpretacji wyników jako nowotwory złośliwe, gdy w rzeczywistości są one łagodne. Jednocześnie liczba fałszywie ujemnych przypadków jest stosunkowo niska, co wskazuje na to, że metoda ta może być bardziej skuteczna w identyfikowaniu rzeczywistych nowotworów złośliwych.

5 Analiza końcowa i wnioski

Na potrzeby bardziej dogłębnego porównania dwóch modeli (liniowego i kwadratowego) stworzymy nowy model, który będzie zaliczał nowotwór do łagodnych tylko wtedy, gdy został zaklasyfikowany jako łagodny w obu modelach.

```
comparison_matrix = np.where(p_square_check == p_linear_check,
p_linear_check, -1)
np.where(np.where(comparison_matrix == b_validate, True, False))[0].size # 252
#nowy model zgadza się ze zbiorem walidacyjnym w 252/260 przypadków
```

Listing 7: Zgodność predykcji dla nowego modelu

Przedstawiamy także wyniki, ile zdarzyło się przypadków false positives i false negatives.

```
p_comparison_false_negatives = np.where((comparison_matrix == -1) &
(b_validate == 1))
p_comparison_false_positives = np.where((comparison_matrix == 1) &
(b_validate == -1))

p_comparison_false_positives[0].size # 2
p_comparison_false_negatives[0].size # 6
```

Listing 8: Przypadki false positives i false negatives dla nowego modelu

Widzimy, jak niewiele różnią się wyniki w obu metodach. $252/260 \approx 97\%$ przypadków zostało zaklasyfikowanych poprawnie. W ten sposób możemy wykorzystywać nasze modele, żeby unikać jak największej ilości przypadków false-positive (w naszym przypadku, tylko 2 nowotwory łagodne zostały zaklasyfikowane do złośliwych). To prowadzi do ciekawego spostrzeżenia, że możemy skorzystać z różnic w działaniu modeli, żeby poprzez zmianę podejścia do liczenia wyników próbować w dużym stopniu wyeliminować jeden z typów błędów.

Jednakże, najważniejszym spostrzeżeniem na potrzeby naszego porównania dwóch podejść jest to, jak duży wpływ ma struktura macierzy na wynik końcowy w metodzie najmniejszych kwadratów. W tym przypadku prawdopodobnie zaszło zjawisko nadmiernego dopasowania (zwłaszcza w przypadku modelu kwadratowego), ale to nie jest zaskakujące ze względu na mały rozmiar danych. Warto też zwrócić uwagę na to, że obydwa modele dużo częściej (ok. 3 razy częściej) myliły się, zaliczając nowotwory łagodne do złośliwych niż na odwrót. Ciężko stwierdzić z czym to może być związane, prawdopodobnie chodzi o jakieś nierównomierne tendencje danych, których nie byliśmy w stanie wyeliminować w testowanych modelach, wykorzystując stosunkowo małe zbiory testowe.

Podsumowując, oczywiście model liniowy był bardziej skuteczny. Wyniki klasyfikacji były ponad dwa razy lepsze w porównaniu do modelu kwadratowego. Z drugiej strony, teoretycznie model kwadratowy ma większy potencjał z danymi, których charakterystyki nie wykazują się liniowym zachowaniem i mógłby pokazać się z lepszej strony na większych zbiorach danych.

Nawiązując jeszcze do różnych wyników przy obliczaniu współczynnika uwarunkowania: różnice te mogą wynikać z innego sposobu obliczania go - funkcja *cond* może obliczyć ten współczynnik na 7 możliwych sposobów (w zależności od opcjonalnego parametru), arytmetyka zmiennoprzecinkowa także mogła się tu dać we znaki.

Literatura

[1] Materiały pomocnicze do laboratorium zamieszczone na platformie Teams w katalogu *lab02/lab2-intro.pdf*.