Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

Wydział Inżynierii Metali i Informatyki Przemysłowej

Sprawozdanie z Laboratorium:

Aproksymacja – metoda najmniejszych kwadratów

Przedmiot: Metody Numeryczne

Kierunek: Inżynieria Obliczeniowa

Autor: Filip Rak

Prowadzący przedmiot: dr hab. inż. Marcin Hojny

Data: 15 marca 2024

Numer lekcji: 4

Grupa laboratoryjna: 4

Wstęp teoretyczny

Regresja liniowa jest modelem statystycznym polegającym na aproksymacji danych, przy czym model ten przedstawiany jest za pomocą prostego równania liniowego:

$$y = a_0 + a_1 x \tag{1}$$

Parametry a_0(wyraz wolny) i a_1 (współczynnik kierunkowy) określające tę linię są zwykle obliczane przy użyciu metody najmniejszych kwadratów. Jest to technika statystyczna służąca do znajdowania linii najlepiej dopasowującej się do zestawu punktów danych poprzez minimalizację sumy kwadratów różnic między wartościami obserwowanymi a wartościami przewidywanymi przez model.

$$a_1 = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \tag{2}$$

$$a_0 = \frac{\sum_{i=1}^{n} y_i \, n\sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \tag{3}$$

W odróżnieniu od interpolacji, gdzie szukamy funkcji dokładnie przechodzącej przez wszystkie punkty danych, cel aproksymacji polega na znalezieniu takiej funkcji, która minimalizuje średni błąd kwadratowy między danymi a funkcją, co oznacza, że dane te nie będą dokładnie na estymowanej linii.

Współczynnikiem korelacji nazywamy liczbę określającą w jakim stopniu zmienne x i y są współzależne. Współczynnik może przyjmować wartości od −1 (zupełna korelacja ujemna), przez 0 (brak korelacji) do +1 (zupełna korelacja dodatnia). Korelacje jesteśmy w stanie obliczyć przy wykorzystaniu następującego wzoru:

$$r = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}\sqrt{n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2}} \tag{4}$$

Implementacja

Implementację modelu regresji zrealizowano w języku C++ w postaci klasy `LinearRegression`. Klasa składa się z atrybutów opisujących estymowaną prostą oraz współczynnik korelacji, konstruktora, niezbędnych getterów i metody przewidującej wartość y dla znanego x. Poszczególne elementy klasy zostały szczegółowo opisane począwszy od następnej strony.

Pełna definicja klasy:

```
class LinearRegression {
	//Attributes
	double a0, a1, r;

public:
	//Constructor
	LinearRegression(int n, Point* p_arr)
	{
		//shared variables
		double xy = 0, xn = 0, yn = 0, x2 = 0, y2 = 0;
		for (int i = 0; i < n; i++)
		{
			xy += p_arr[i].getX() * p_arr[i].getY();
			x2 += pow(p_arr[i].getX(), 2);
			y2 += pow(p_arr[i].getY(), 2);
			xn += p_arr[i].getX();
			yn += p_arr[i].getY();
		}

		//shared by a0 and a1
		double denominator = (n * x2) - pow(xn, 2);

		//coefficients
		if (denominator == 0)
			throw std::invalid_argument("invalid point array");
		else
		{
			this->a1 = ((n * xy) - (xn * yn)) / denominator;
			this->a0 = ((x2 * yn) - (xn * xy)) / denominator;
		}

		denominator = sqrt((n * x2 - pow(xn, 2)) * (n * y2 - pow(yn, 2)));
		if (denominator == 0)
			this->r = 0;
		else
			this->r = ((n * xy) - (xn * yn)) / denominator;
	}

	//Getters
	double getIntercept() { return a0; }
	double getSlope() { return a1; }
	double getCorrelation() { return r; }

	//Methods
	double predict(double x) { return a0 + a1 * x; }
};
```

Atrybuty klasy

Klasa przechowuje trzy, prywatne liczby zmiennoprzecinkowe podwójnej precyzji: Zmienne `a0` i `a1`, będące wyrazem wolnym oraz współczynnikiem kierunkowym modelu, oraz zmienną `r`, będącą współczynnikiem korelacji. Wszystkie te wartości są obliczane w konstruktorze, na etapie tworzenia obiektu klasy.

```
double a0, a1, r;
```

Omówienie poszczególnych elementów konstruktora klasy:

Konstruktor przyjmuje następujące parametry: liczbę całkowitą `n`, zawierającą wielkość tablicy punktów oraz tablicę obiektów klasy `Point.` Klasa `Point` jest prostą strukturą reprezentującą punkt za pomocą współrzędnych x i y typu double, wyposażoną w niezbędne akcesory (gettery i settery).

```
LinearRegression(int n, Point* p_arr)
```

Uważnie analizując wzory (2), (3) i (4) możemy zauważyć, iż występujące w nich sumy często się powtarzają. Wykorzystując tę zależność, w celu ograniczenia redundancji i poprawienia optymalizacji, konstruktor na pierwszym etapie swojego działania oblicza wartości wszystkich sum potrzebnych do obliczenia trzech współczynników `a0`, `a1` i `r`.

```
double xy = 0, xn = 0, yn = 0, x2 = 0, y2 = 0;
for (int i = 0; i < n; i++)
{
     xy += p_arr[i].getX() * p_arr[i].getY();
     x2 += pow(p_arr[i].getX(), 2);
     y2 += pow(p_arr[i].getY(), 2);
     xn += p_arr[i].getX();
     yn += p_arr[i].getY();
}
```

Obliczany jest wspólny mianownik dla współczynników `a0` i `a1`.

```
double denominator = (n * x2) - pow(xn, 2);
```

Wykorzystując instrukcję warunkową przed wykonaniem dzielenia upewniamy się, że wartość mianownika nie jest równa zero. Wartość mianownika będzie równa zero w sytuacji w której każdy punkt przekazany konstruktorowi będzie miał jednakowe położenie. W takim wypadku wyrzucamy wyjątek informujący o podaniu niepoprawnej tablicy punktów. W przypadku przeciwnym obliczamy wartości współczynników `a0` i `a1`, wykorzystując wzory (2) i (3).

```
if (denominator == 0)
      throw std::invalid_argument("invalid point array");
else
{
      this->a1 = ((n * xy) - (xn * yn)) / denominator;
      this->a0 = ((x2 * yn) - (xn * xy)) / denominator;
}
```

Wzorem (3) obliczamy mianownik dla współczynnika korelacji `r`. W przypadku w którym mianownik jest równy zero, uznajemy, że zmienne nie są skorelowane, reprezentując to poprzez ustawienie wartości `r` na zero. W przypadku przeciwnym obliczamy wartość zgodnie ze wzorem (3).

```
denominator = sqrt((n * x2 - pow(xn, 2)) * (n * y2 - pow(yn, 2)));
if (denominator == 0)
      this->r = 0;
else
      this->r = ((n * xy) - (xn * yn)) / denominator;
```

Akcesory klasy

Klasa zawiera trzy gettery zwracające wartości prywatnych atrybutów, punkt przecięcia z osią Y, współczynnik kierunkowy oraz współczynnik korelacji.

```
double getIntercept() { return a0; }
double getSlope() { return a1; }
double getCorrelation() { return r; }
```

Metoda `predict`

Zaimplementowana została metoda `predict`. Jej jedynym parametrem jest znany x na podstawie którego z wykorzystaniem wzoru (1) przewidywana jest wartość y.

```
double predict(double x) { return a0 + a1 * x; }
```

Testy na wybranych przykładach

Skuteczność implementacji modelu regresji liniowej została przetestowana na kilku starannie wybranych zestawach danych. Wyniki zostały porównane z wynikami otrzymanymi w programie Microsoft Excel.

Testy z danymi liniowymi.

Test 1.1:

Znane punkty: (1, 5), (2, 7), (3, 9), (4, 11), (5, 13)

Przewidywany punkt: $x = 5$

Wynik Excel: $a_1 = 2,\ a_0 = 3, r = 1, y = 13$

Wynik implementacji: $a_1 = 2,\ a_0 = 3, r = 1, y = 13$

Testy z rzeczywistymi / nieliniowymi danymi.

Test 2.1:

Znane punkty: (28.0, 172.72), (36.0, 167.64), (34.0, 165.1), (27.0, 175.26), (45.0, 172.72), (27.0, 160.02), (65.0, 160.02), (33.0, 160.02), (26.0, 167.64), (32.0, 165.1), (30.0, 170.18)

Przewidywany punkt: $x = 36$

Wynik Excel: $a_1 = -0.16,\ a_0 = 172, r = -0.31, y = 167.39$

Wynik implementacji: $a_1 = -0.16,\ a_0 = 172, r = -0.32, y = 167.39$

Test 2.2:

Znane punkty: (22.0, 165.1), (40.0, 167.64), (35.0, 167.64), (40.0, 160.02)

Przewidywany punkt: $x = 10$

Wynik Excel: $a_1 = -0.06,\ a_0 = 167.1, r = -0.14, y = 170.45$

Wynik implementacji: $a_1 = -0.06,\ a_0 = 167.1, r = -0.14, y = 170.45$

Testy z dużym rozstrzałem punktów:

Test 3.1

Znane punkty: (10, 95), (50, 480), (90, 920)

Przewidywany punkt: $x = 3$

Wynik Excel: $a_1 = 10.31,\ a_0 = -17.29, r = 0.99, y = 13.65$

Wynik implementacji: $a_1 = 10.31,\ a_0 = -17.29, r = 0.99, y = 13.65$

Test 3.2

Znane punkty: (5, 45), (20, 195), (35, 340), (50, 510), (65, 640), (80, 815), (95, 940)

Przewidywany punkt: $x = 6$

Wynik Excel: $a_1 = 10.06,\ a_0 = -5.12, r = 0.99, y = 55.24$

Wynik implementacji: $a_1 = 10.06,\ a_0 = -5.12, r = 0.99, y = 55.24$

Opracowanie wyników testów

Poniższe opracowanie przedstawia wyniki testów implementacji modelu regresji liniowej przeprowadzone na wybranych zestawach danych. Porównano je z odpowiednikami uzyskanymi za pomocą programu Microsoft Excel, który służył jako punkt odniesienia.

Wszystkie przeprowadzone testy wykazały wysoką zgodność między wynikami zaimplementowanego modelu a tymi uzyskanymi w Excelu. Szczególną uwagę można zwrócić na Test 2.1, gdzie niewielka różnica w wartościach współczynnika korelacji r może być przypisana drobnym różnicom w precyzji obliczeń numerycznych.

W testach 1.1, 2.2 oraz 3.1 i 3.2, wyniki były identyczne z wynikami Excela, co potwierdza, że zaimplementowany model regresji liniowej przewiduje parametry z dużą dokładnością.

*	***Excel***				***Implementacja***			
Test	a_1	a_0	r	y	a_1	a_0	r	y
1.1	2	3	1	13	2	3	1	13
2.1	−0.16	172	-0.31	167.39	−0.16	172	−0.32	167.39
2.2	−0.06	167.1	−0.14	170.45	−0.06	167.1	−0.14	170.45
3.1	10.31	−17.29	0.99	13.65	10.31	−17.29	0.99	13.65
3.2	10.06	−5.12	0.99	55.24	10.06	−5.12	0.99	55.24

Tabela 1

Podsumowanie:

Na podstawie otrzymanych wyników można stwierdzić, że zaimplementowany model regresji liniowej jest niezawodny i może być skutecznie wykorzystywany do analizy danych. Model ten może być zastosowany zarówno do danych idealnie liniowych, jak i danych rzeczywistych, które mogą zawierać pewne nieregularności. Dodatkowo, testy z dużym rozstrzałem punktów pokazały, że model jest w stanie radzić sobie z danymi posiadającymi szeroki zakres wartości, zachowując przy tym wysoką precyzję estymacji.