

Introduktion til polynomiell regression i Python og numpy

Af Henrik Sterner (henrik.sterner@gmail.com)

Polynomiell regression er en udvidelse af lineær regression, hvor den afhængige variabel er en funktion af en polynomiell funktion af den uafhængige variabel.

I det følgende introduceres polynomiell regression og hvordan det kan implementeres i Python ved brug af numpy.

Indhold

- Hvad er et polynomium?
- Graden af et polynomium
- Polynomiell regression
- Eksempel på polynomiell regression
- Beregning af polynomiell regression i Python ved brug af numpy
- Visualisering af polynomiell regression i Python ved brug af matplotlib
- Eksempel på polynomiell regression med flere variable
- Beregning af polynomiell regression med flere variable i Python ved brug af numpy

Hvad er et polynomium?

Et polynomium er en funktion, der er givet ved en sum af potenser af en variabel, hvor koefficienterne er reelle tal. Her nogle eksempler:

$$f(x) = 3x^2 + 2x + 1$$

$$g(x) = 5x^3 + 4x^2 + 3x + 2$$

$$h(x) = 7x^4 + 6x^3 + 5x^2 + 4x + 3$$

Matematisk notation

Ved et polynomium af grad n forstås en funktion $f(x)$, der er givet ved:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0$$

hvor $a_n, a_{n-1}, \dots, a_2, a_1, a_0$ er reelle tal og $a_n \neq 0$.

Graden af et polynomium

Ved graden af et polynomium forstås den højeste potens af variabelen i polynomiet.

Eksempelvis er graden af polynomiet $f(x) = 3x^2 + 2x + 1$ lig 2, da den højeste potens af variabelen x er 2.

Polynomiell regression

Polynomiell regression er en udvidelse af lineær regression, hvor den afhængige variabel er en funktion af en polynomiell funktion af den uafhængige variabel.

Fordelene ved polynomiell regression er, at den kan tilpasses til en bred vifte af funktioner, og at den kan tilpasses til data, der ikke er lineært fordelt.

Eksempel på polynomiell regression

Betragt følgende datasæt:

x	y
1	1
2	4
3	9
4	16
5	25
6	36

Eksempel på polynomiell regression i numpy

Først importeres numpy:

```
import numpy as np
```

Dernæst defineres datasættet:

```
x = np.array([1, 2, 3, 4, 5, 6])  
y = np.array([1, 4, 9, 16, 25, 36])
```

Eksempel på polynomiell regression i numpy

Herefter beregnes polynomiell regression:

```
p = np.polyfit(x, y, 2)
```

Her er p en vektor med koefficienterne i polynomiet.

```
p = [ 1.00000000e+00 -1.59471172e-14  1.00000000e+00]
```

Det vil sige, at polynomiet er givet ved:

$$f(x) = 1.00000000e + 00x^2 - 1.59471172e - 14x + 1.00000000e + 00$$

Her betyder e+00, at der skal flyttes 0 decimaler til højre, og e-14 betyder, at der skal flyttes 14 decimaler til venstre. Dvs. tilnærmet:

$$f(x) = x^2 + 1$$

Visualisering af polynomiell regression i Python ved brug af matplotlib

Først importeres matplotlib:

```
x = np.array([1, 2, 3, 4, 5, 6])
y = np.array([1, 4, 9, 16, 25, 36])
p = np.polyfit(x, y, 2)
plt.scatter(x, y)
plt.plot(x, np.polyval(p, x))
plt.show()
```

Polynomiell regression med flere variable

Polynomiell regression kan også bruges til at tilpasse en funktion til data, der afhænger af flere variable.

Eksempelvis:

x1	x2	y
1	1	3
3	2	8
6	3	15
7	4	24
9	5	35

Beregning af polynomiell regression med flere variable i Python ved brug af numpy

Først defineres datasættet:

```
x = np.array([[1, 1], [3, 2], [6, 3], [7, 4], [9, 5]])
y = np.array([3, 8, 15, 24, 35])
```

Herefter beregnes polynomiel regression:

```
p = np.polyfit(x, y, 2)
```

Evaluering af polynomiel regression

Polynomiel regression kan evalueres ved brug af forskellige metoder, herunder bl.a.:

- R-squared/ R^2
- Mean squared error/MSE
- Mean absolute error/MAE
- Max error/ME
- Mean squared logarithmic error/MSLE

R-squared/ R^2

R-squared er en statistisk metode til at evaluere, hvor godt en model passer til data. R-squared er en værdi mellem 0 og 1, hvor 1 indikerer, at modellen passer perfekt til data. Den udregnes ved:

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

hvor y_i er den faktiske værdi, \hat{y}_i er den forudsagte værdi og \bar{y} er gennemsnittet af de faktiske værdier.

R-squared/ R^2 i Python

R-squared kan udregnes i Python ved brug af numpy:

```
yhat = np.polyval(p, x)
ybar = np.mean(y)
R2 = 1 - np.sum((y - yhat)**2)/np.sum((y - ybar)**2)
```

Mean squared error/MSE

Mean squared error er en statistisk metode til at evaluere, hvor godt en model passer til data. Mean squared error er en værdi mellem 0 og ∞ , hvor 0 indikerer, at modellen passer perfekt til data. Den udregnes ved:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

hvor y_i er den faktiske værdi og \hat{y}_i er den forudsagte værdi.

På dansk: Gennemsnittet af kvadraterne af forskellene mellem de faktiske og forudsagte værdier.

Mean squared error/MSE i Python

Mean squared error kan udregnes i Python ved brug af numpy:

```
MSE = np.mean((y - yhat)**2)
```

Mean absolute error/MAE

Mean absolute error er en statistisk metode til at evaluere, hvor godt en model passer til data. Mean absolute error er en værdi mellem 0 og ∞ , hvor 0 indikerer, at modellen passer perfekt til data. Den udregnes ved:

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

hvor y_i er den faktiske værdi og \hat{y}_i er den forudsagte værdi.

På dansk: Gennemsnittet af de absolutte forskelle mellem de faktiske og forudsagte værdier.

Mean absolute error/MAE i Python

Mean absolute error kan udregnes i Python ved brug af numpy:

```
MAE = np.mean(np.abs(y - yhat))
```

Max error/ME

Max error er en statistisk metode til at evaluere, hvor godt en model passer til data. Max error er en værdi mellem 0 og ∞ , hvor 0 indikerer, at modellen passer perfekt til data. Den udregnes ved:

$$ME = \max(|y_i - \hat{y}_i|)$$

hvor y_i er den faktiske værdi og \hat{y}_i er den forudsagte værdi.

På dansk: Den største af de absolutte forskelle mellem de faktiske og forudsagte værdier.

Max error/ME i Python

Max error kan udregnes i Python ved brug af numpy:

```
ME = np.max(np.abs(y - yhat))
```

Styrker ved polynomiell regression

Styrker: * Kan tilpasses til en bred vifte af funktioner * Kan tilpasses til data, der ikke er lineært fordelt * Kan tilpasses til data, der afhænger af flere variable * Kan evalueres ved brug af forskellige metoder * Kan implementeres i Python ved brug af numpy * Relativt hurtig at beregne og implementere

Svagheder ved polynomiell regression

- Kan være svær at fortolke. Højere grad af polynomium kan give mere komplekse modeller, der kan være svære at fortolke.
- Kan være følsom over for outliers. Outliers kan have stor indflydelse på modellen.
- Kan være følsom over for overfitting. Højere grad af polynomium kan give modeller, der passer for godt til data.

Matematikken bag polynomiell regression

Bag polynomiell regression ligger matematikken bag polynomier og lineær algebra. Polynomiell regression kan implementeres ved brug af lineær algebra og matricer.

Vi har en lang række slides, der beskriver grundlæggende lineær algebra og matricer (se hjemmesiden), som er nødvendige for at forstå matematikken bag polynomiell regression. Særligt vigtigt er slides om rank, determinant, invers, egenvektorer og egenvektorer.

Recap af lineær uafhængighed og rank

Givet en $m \times n$ matrix A på formen:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

Betragt kolonnerne i A som vektorer i \mathbb{R}^m . Kald kolonnerne i A for a_1, a_2, \dots, a_n .

Linear kombination og Lineær uafhængighed

En linear kombination af vektorerne a_1, a_2, \dots, a_n er givet ved:

$$c_1 a_1 + c_2 a_2 + \dots + c_n a_n$$

hvor c_1, c_2, \dots, c_n er reelle tal.

En vektor a_i er lineært uafhængig af vektorerne $a_1, a_2, \dots, a_{i-1}, a_{i+1}, \dots, a_n$, hvis den ikke kan skrives som en lineær kombination af de andre vektorer. Dvs.:

$$c_1 a_1 + c_2 a_2 + \dots + c_{i-1} a_{i-1} + c_{i+1} a_{i+1} + \dots + c_n a_n \neq a_i$$

Rank

Rank er lig antallet af lineært uafhængige kolonner i en matrix. Dvs.

$$\text{rank}(A) = \text{antallet af lineært uafhængige kolonner i } A$$

Det kan vises, at rank er lig antallet af lineært uafhængige rækker i en matrix. Dvs.

$$\text{rank}(A) = \text{antallet af lineært uafhængige rækker i } A$$

Dvs. rank er mindre eller lig antallet af rækker og kolonner i en matrix:

$$\text{rank}(A) \leq \min(m, n)$$

Rank kaldes fuldrank, hvis rank er lig antallet af rækker eller kolonner i en matrix.

Pseudo-invers og invers

En matrix A har en invers, hvis den er kvadratisk og har fuld rank. Dvs.

$$A \in \mathbb{R}^{n \times n} \text{ og } \text{rank}(A) = n$$

I så fald er der en matrix A^{-1} , således at:

$$AA^{-1} = A^{-1}A = I$$

hvor I er identitetsmatricen.

Hvis en matrix ikke har en invers, kan

$$AA^{-1} \approx I$$

hvor A^{-1} kaldes for pseudo-inversen.

mxn matrix

En $m \times n$ matrix A har ikke en invers, hvis $m \neq n$ eller $\text{rank}(A) < n$. Dvs. kun kvadratiske matricer med fuld rank har en invers.

Hvis A er en $m \times n$ matrix, så er $A^T A$ en $n \times n$ matrix, og AA^T er en $m \times m$ matrix.

mxn matrix fortsat

Vi har følgende resultater:

- $A^T A$ har fuld rank, hvis A har fuld rank.
- AA^T har fuld rank, hvis A har fuld rank.
- Hvis A har fuld rank, så har $A^T A$ og AA^T en invers.

Vi kalder følgende for den pseudoinverse af en matrix:

$$A^+ = (A^T A)^{-1} A^T$$

Polynomiell regression og matematikken bag

Antag vi har givet n punkter med hver m koordinater. Dvs. vi har n punkter i \mathbb{R}^m . Kald det i 'te punkt for p_i .

Lad nu v_i være de korresponderende værdier til punkterne p_i . Dvs. v_i er værdien af en funktion i punktet p_i .

Polynomiell regression går ud på at finde en polynomiefunktion $f(x)$, således at $f(p_i) \approx v_i$ for alle i .

Minimalisering af fejl

Polynomiell regression går ud på at finde en polynomiefunktion $f(x)$, således at $f(p_i) \approx v_i$ for alle i .

Mere konkret vil vi finde en polynomielfunktion $f(x)$, således at:

$$E = \sum_{i=1}^n (f(p_i) - v_i)^2$$