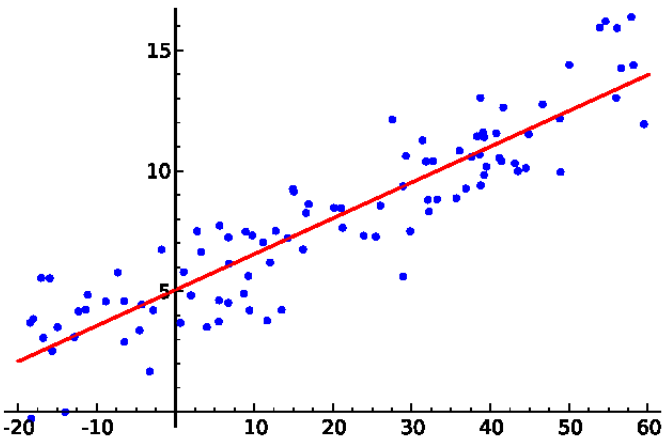
Teori och instruktioner för

Programmeringsuppgift

Minsta kvadratmetoden

i gymnasiekursen
Matematik specialisering

November 2021



Typsatt med LaTeX
kompilerad på en Raspberry Pi

Innehåll

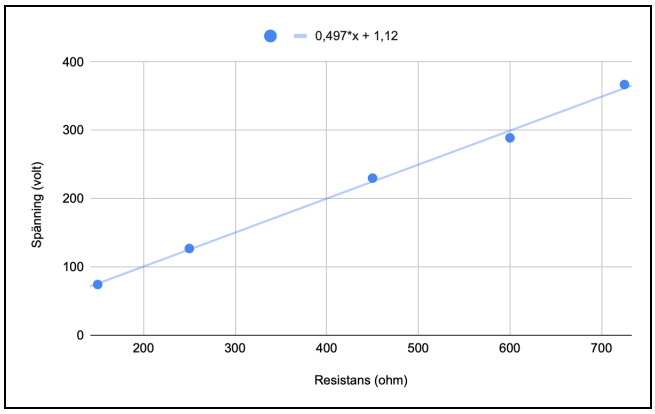
1 Inledande exempel

Du har tidigare anpassat en rät linje till ett antal mätvärden, det används ofta för att skapa matematiska modeller från ett begränsat antal värden. Det kan t ex gälla att vi vill ta reda på hur spänningen över en resistor beror på dess resistans givet att strömmen hålls konstant. Säg att vi har fem resistorer som vi mäter spänningen över, och att vi får följande värden för en och samma ström:

Resistans, R [Ω]	Spänning, U [V]
150	74.3
250	127
450	230
600	289
725	367

Tabell 1: Mätvärden

Detta kan plottas i ett diagram (nedanstående diagram är gjort i Google kalkylark):



Figur 1: Plottade mätvärden

Vi ser att punkterna nästan ansluter till en rät linje med ekvationen $y = 0.497x + 1.12$. Ekvationen tar kalkylarket fram åt oss med en metod som kallas *minsta kvadratmetoden*. Teorin för minsta kvadratmetoden hittar vi i den linjära algebran. Vi ska titta lite närmare på algoritmen med vilken man får fram ekvationen för den räta linje som ansluter bäst till ett antal punkter. Den här uppgiften går ut på att du ska skriva ett program utifrån algoritmen som bestämmer ekvationen för den räta linje som anpassas till ett godtyckligt antal punkter.

2 Minsta kvadratmetoden

2.1 Teckning av ekvationerna

Om vi vill teckna förutsättningarna för en rät linje utifrån värdena i Tabell 1 så erhålls följande ekvationssystem:

$$\begin{cases} 74.3 &= \beta_0 &+ &\beta_1 \cdot 150 \\ 127 &= \beta_0 &+ &\beta_1 \cdot 250 \\ 230 &= \beta_0 &+ &\beta_1 \cdot 450 \\ 289 &= \beta_0 &+ &\beta_1 \cdot 600 \\ 367 &= \beta_0 &+ &\beta_1 \cdot 725 \end{cases} \tag{1}$$

Detta ekvationssystem (1) saknar lösning eftersom punkterna inte ligger på en rät linje. Minsta kvadratmetoden går ut på att anpassa y-värdena så att så att det finns tal β_0 och β_1 som gör att alla likheter uppfylls.

Uppgift: Skriv om ekvationssystemet på formen $A\vec{x} = \vec{y}$.

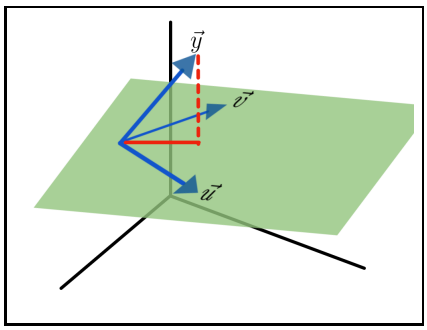
Innan vi går in på själva räknealgoritmen för Minsta kvadratmetoden så vill jag visuellt åskådliggöra vad som händer.

2.2 En visuell beskrivning av Minsta kvadratmetoden

För att demonstera en metod som någorlunda enkelt påvisar om det är lösbart eller inte så presenterar jag ett mindre system nedan. Det består av tre ekvationer och två obekanta, och jag skriver det på formen $A\vec{x} = \vec{y}$.

$$\begin{bmatrix} 1 & -3 \\ 3 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 3 \end{bmatrix} \tag{2}$$

De båda kolonnerna i matrisen A, $\vec{u} = (1, 3, 0)$ och $\vec{v} = (-3, 1, 0)$ kan ses som vektorer i ett och samma plan (två vektorer i 3D bildar alltid ett plan) medan vektorn $\vec{y}$ är en vektor som inte ligger i detta plan. Figur 2 nedan illustrerar situationen (inspirerat av [1, p. 379]) *Om* $\vec{y}$ hade legat i samma plan som $\vec{u}$ och $\vec{v}$ skulle ekvationssystemet (2) haft en lösning.



Figur 2: Vektorn $\vec{y}$ projicerad på $\vec{u}\vec{v}$-planet

Detta kommer sig av att lösningen måste vara en vektor som är en linjärkombination av $\vec{u}$ och $\vec{v}$, och sådana kan inte "bryta sig ur" det plan som de ingående komponenterna ligger i.

Vad gör vi då för att approximera en lösning? Jo, vi *ortogonalprojicerar* $\vec{y}$ på det aktuella planet. Det är den röda, heldragna, linjen i Figur 2 som motsvarar denna ortogonalprojektion. Ju närmare planet som $\vec{y}$ ligger från början, desto mindre kommer avvikelsen att bli efter projektionen.

2.3 Algoritmen för Minsta kvadratmetoden

Nedan så presenteras den matematiska algoritmen för ortogonalprojektion. Om du kommer att läsa en kurs i linjär algebra på högskola eller universitet kommer den att få sin förklaring, men den ingår inte i denna gymnasiekurs. Det finns även länk till en textmaterial med textmaterial med länkade videor i referenserna nedan för den som redan nu är nyfiken.

För att ortogonalprojicera multipliceras från vänster med A^T på båda sidor [2, p. 85]. A^T är *transponatet* till A, vilket innebär att raderna blir kolonner och kolonnerna blir rader. T ex gäller

$$A = \begin{bmatrix} 1 & -3 \\ 3 & 1 \\ 0 & 0 \end{bmatrix} \Rightarrow A^T = \begin{bmatrix} 1 & 3 & 0 \\ -3 & 1 & 0 \end{bmatrix} \tag{3}$$

Ekvationssystemet (1) tecknas på matrisform som

$$\begin{bmatrix} 1 & 150 \\ 1 & 250 \\ 1 & 450 \\ 1 & 600 \\ 1 & 725 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} 74.3 \\ 127 \\ 230 \\ 289 \\ 367 \end{bmatrix} \tag{4}$$

Med metoden att multiplicera med transponatet så erhålls

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 150 & 250 & 450 & 600 & 725 \end{bmatrix} \begin{bmatrix} 1 & 150 \\ 1 & 250 \\ 1 & 450 \\ 1 & 600 \\ 1 & 725 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 150 & 250 & 450 & 600 & 725 \end{bmatrix} \begin{bmatrix} 74.3 \\ 127 \\ 230 \\ 289 \\ 367 \end{bmatrix} \tag{5}$$

Detta kan se klurigt ut, men det är enbart matrismultiplikation (som dessutom ska göras på dator eftersom det är en programmeringsövning). Resultatet kommer att bli

$$\begin{bmatrix} 5 & 2\,175 \\ 2\,175 & 1\,173\,100 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} 1\,087.3 \\ 585\,870 \end{bmatrix} \tag{6}$$

Vi ser att det är ett system med två ekvationer och två obekanta (så kommer det alltid att bli när en rät linje ska approximeras). Sådana system är enkla att lösa eftersom det finns en metod med få steg att invertera den ingående 2×2-matrisen (se kap. 1 i kurskompendiet [3]). Just det här ekvationssystemet får lösningen $\beta_0 = 1.120$ och $\beta_1 = 0.497$. Det innebär att vi kan approximera en linje genom punkterna med ekvationen $y = 0.497x + 1.12$ (där y är spänningen och x är resistansen i exemplet med mätvärden i Tabell 1). Det är exakt detsamma som Google kalkylark beräknade åt oss i Figur 1.

3 Uppgift

3.1 Uppgiftsformulering

Du ska skriva ett program i Python som implementerar Minsta kvadratmetoden enligt ovanstående beskrivning av algoritmen. I själva huvudprogrammet ska `x`-vektorn och `y`-

vektorn anges separat, det är dessa som kommer att motsvara kolumnerna med värden i Tabell 1. Det ska kunna vara ett godtyckligt antal par av mätvärden i respektive vektor. Dessa ska sedan skickas till en funktion som returnerar de eftersökta värdena på β_0 och β_1 (som är definierade i ekvationssystem (1)).

3.2 Tips

- Börja med att utföra importen av `NumPy`: `import numpy as np`
- Skapa dina vektorer `x` och `y` som `matrix`, t ex `x = np.matrix([1, 2, 3])`.
- För att skapa en tom matris med m rader och 2 kolonner kan du använda `A = np.empty((m, 2))` (där m är antalet element i `x`- vektorn)
- För att ersätta en kolonn med en vektor `v` i matrisen `A` kan du utföra `A[:,1] = v`. Detta ersätter kolonnen med ordningsnummer 1, dvs den andra kolonnen (`NumPy` är nollindexerat). Här gäller att vektorn `v` måste vara en radvektor med lika många element som matrisen `A` har rader.
- Matriser kan transponeras med metoden `transpose`, t ex `A.transpose()`. Det brukar vara bra att spara transponatet i en separat variabel, t ex `A_T = A.transpose()`.
- En matris kan multipliceras med en annan matris eller vektor (av rätt dimension, förstås) med `@`-operatorn, t ex `AB = A@B`.
- En matris, t ex `A`, kan inverteras genom att utföra `np.linalg.inv(A)`. Det brukar vara bra att spara inversen i en separat variabel, t ex `A_inv = np.linalg.inv(A)`.

4 Referenser

[1] D. C. Lay, *Linear Algebra and its Applications*. Pearson Education Limited, 2016.

[2] J. R. Chasnov, "Matrix algebra for engineers." `https://www.math.hkust.edu.hk/~machas/matrix-algebra-for-engineers.pdf`, 2018. Accessed on 2021-11-13.

[3] R. Skjelnes, "Linjär algebra, tio förrätter och två efterrätter." `https://www.kth.se/polopoly_fs/1.1007126.1599206012!/LinearAlgebra.pdf`, 2014. Matematiska institutionen, KTH. Accessed on 2021-11-13.