KTH Matematik

Matematisk Statistik

SF1918 Sannolikhetsteori och statistik, HT 2024
Laboration 2 för CINEK och CTMAT

1 Introduktion

Denna laboration kommer på redovisningstillfället att bedömas som antingen godkänd eller ej godkänd. De studenter som blir godkända på laborationen kan tillgodoräkna sig uppgift 12 på del I av tentamen och får tre bonuspoäng på del II. Observera att detta endast gäller vid den ordinarie tentamen och det första omtentamenstillfället.

Läs först igenom labbspecifikationen två gånger. Försäkra dig om att du förstår hur den Pythonkod som visas i texten fungerar. Svaren på förberedelseuppgifterna ska skrivas på papper och kunna redovisas **individuellt**. Arbete i grupp är tillåtet (och uppmuntras) med **högst två** personer per grupp.

2 Förberedelseuppgifter

1. En Rayleighfördelad stokastisk variabel X har täthetsfunktionen

$$f_X(x) = \begin{cases} \frac{x}{b^2} e^{-\frac{x^2}{2b^2}}, & x \geq 0, \\ 0, & x < 0. \end{cases}$$

   Antag nu att du har n stycken Rayleighfördelade variabler.

   a) Bestäm ML-skattningen av b som funktion av ett stickprov $x_1, x_2, \ldots, x_n$.

   b) Gör detsamma för MK-skattningen av b.

2. Beskriv hur du kan ta fram ett approximativt konfidensintervall för parametern b. Motivera varför det är rimligt att göra den approximation som du har gjort. Ledning: Använd MK-skattningen av b.

3. Beskriv idén bakom linjär regression. Ladda ner filen `tools.py` och importera modulen med hjälp av kommandot `import tools`. Beskriv hur man med hjälp av funktionen `tools.regress` kan skatta parametrarna i modellen

$$w_k = \log(y_k) = \beta_0 + \beta_1 x_k + \varepsilon_k \tag{1}$$

3 Nödvändiga filer

Börja med att ladda ner filen `data.zip` från kurshemsidan och extrahera filerna därifrån. Innehållet är

- `wave_data.txt`, en diskretiserad radiosignal.
- `birth.dat`, data om förstföderskor i Malmö 1991–1992.
- `birth.txt`, beskrivning av datat i `birth.dat`.
- `moore.dat`, utveckling av antalet transistorer per ytenhet för perioden 1972–2019.

Se till att filerna ligger i den katalog du kommer att arbeta i. För att kontrollera att du har lagt filerna rätt, skriv `%ls` och se om filerna ovan listas. (Detta fungerar endast i Jupyter Notebook eller IPython. I standardpythontolken måste du skriva `import os` och sedan `os.listdir()`.)

4 Laborationsuppgifter

Innan du börjar, glöm inte att importera de nödvändiga standardmodulerna med

```
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
```

Ladda sedan ner extramodulen `tools.py`, lägg den i samma katalog som du arbetar i, och importera den med

```
import tools
```

Problem 1 – Simulering av konfidensintervall

Ett konfidensintervall med konfidensgrad $1-\alpha$ för en (okänd) parameter μ innehåller det sanna μ med sannolikhet $1-\alpha$. Vi ska försöka förstå innebörden av detta begrepp med hjälp av simuleringar. Koden nedan använder $n = 25$ oberoende observationer från $N(2,1)$-fördelningen för att skatta ett konfidensintervall för väntevärdet med konfidensgrad 95%. Detta upprepas 100 gånger vilket ger 100 konfidensintervall. Hur många av dessa intervall kan förväntas innehålla det sanna värdet på μ?

```
## Problem 1: Simulering av konfidensintervall

# Parametrar
# Antal mätningar
```

```
n = 25
# Väntevärdet
mu = 2
# Standardavvikelsen
sigma = 1
# Ett minus konfidensgraden
alpha = 0.05
# Antal intervall
m = 100

# Simulera n observationer för varje intervall.
x = stats.norm.rvs(loc=mu, scale=sigma, size=(m, n))

# Skatta mu med medelvärdet.
xbar = np.mean(x, axis=-1)

# Beräkna kvantilerna och standardavvikelsen för
# medelvärdet.
lambda_alpha_2 = stats.norm.ppf(1 - alpha / 2)
D = sigma / np.sqrt(n)

# Beräkna undre och övre gränserna.
undre = xbar - lambda_alpha_2 * D
övre = xbar + lambda_alpha_2 * D
```

Vi kan sedan plotta dessa konfidensintervall för att se hur bra de överensstämmer med den sanna parametern.

```
## Problem 1: Simulering av konfidensintervall (forts.)

# Skapa en figur med storlek 4 × 8 tum.
plt.figure(figsize=(4, 8))

# Rita upp alla intervall
for k in range(m):
    # Rödmarkera alla intervall som missar mu.
    if övre[k] < mu or undre[k] > mu:
        color = 'r'
    else:
        color = 'b'
    plt.plot([undre[k], övre[k]], [k, k], color)

# Fixa till gränserna så att figuren ser lite bättre ut.
b_min = np.min(undre)
```

```
b_max = np.max(övre)
plt.axis([b_min, b_max, -1, m])

# Rita ut det sanna värdet.
plt.plot([mu, mu], [-1, m], 'g')

# Visa plotten.
plt.show()
```

Vad visar de horisontella strecken och det vertikala strecket? Hur många av de 100 intervallen innehåller det sanna värdet på μ? Stämmer resultatet med dina förväntningar? Kör simuleringarna flera gånger.

Variera nu μ, σ, n och α (en i taget) och ser hur de olika parametrarna påverkar resultatet.

Problem 2 – Maximum likelihoodskattning och minsta kvadratskattning

I denna uppgift ska vi undersöka två olika punktskattningar av värdet på parametern i en Rayleighfördelning. Koden nedan genererar en samling Rayleighfördelade stokastiska variabler med parametervärde 4 och plottar sedan skattningarna `est_ml` och `est_mk`. Använd dina två skattningar från förberedelseuppgift 1.

```
## Problem 2: Maximum likelihood, minsta kvadrat
M = 10000
b = 4

# Simulera M utfall med parameter b.
x = stats.rayleigh.rvs(scale=b, size=M)

# Skapa figur och plotta histogrammet.
plt.figure()
plt.hist(x, 40, density=True)

est_ml = # Skriv din ML-skattning här
est_mk = # Skriv din MK-skattning här

# Plotta de två skattningarna.
plt.plot(est_ml, 0.2, 'r*', markersize=10)
plt.plot(est_mk, 0.2, 'g*', markersize=10)
plt.plot(b, 0.2, 'bo')

plt.show()
```

Ser din skattning bra ut? Kontrollera hur täthetsfunktionen ser ut genom att plotta den med din skattning:

```
## Problem 2: Maximum likelihood, minsta kvadrat (forts.)

# Skapa figur.
plt.figure()

# Visa histogrammet.
plt.hist(x, 40, density=True)

# Plotta täthetsfunktionen för den skattade parametern.
x_grid = np.linspace(np.min(x), np.max(x), 60)
pdf = stats.rayleigh.pdf(x_grid, scale=est_ml)
plt.plot(x_grid, pdf, 'r')

plt.show()
```

Problem 3 – Konfidensintervall för Rayleighfördelning

Vi ska nu undersöka en Rayleighfördelad signal, bestämma en punktskattning av parametervärdet samt ta fram ett konfidensintervall för parametern. Ladda in data genom att skriva `y = np.loadtxt('wave_data.dat')`. Filen innehåller en signal som du kan plotta genom att skriva följande kod.

```
## Problem 3: Konfidensintervall for Rayleighfordelning

# Ladda data.
y = np.loadtxt('wave_data.dat')

# Plotta en bit av signalen samt histogrammet.
plt.figure(figsize=(4, 8))

plt.subplot(2, 1, 1)
plt.plot(y[:100])

plt.subplot(2, 1, 2)
plt.hist(y, density=True)

plt.show()
```

Om du ändrar `y[:100]` till `y` så kan du se hela signalen. Skatta parametern i datat på samma sätt som i Problem 2. Spara din skattning som `est`. Ta fram ett konfidensintervall för skattningen och spara övre respektive undre värdet som `upper_bound` respektive `lower_bound`. Plotta nu intervallet för

din skattning av parametern samt kontrollera hur täthetsfunktionen ser ut genom att plotta den med din skattning precis som i föregående problem.

```
## Problem 3: Konfidensintervall (forts.)

# Plotta histogrammet och skattningen.
plt.figure()

plt.hist(y, density=True)
plt.plot(lower_bound, 0.6, 'g*', markersize=10)
plt.plot(upper_bound, 0.6, 'g*', markersize=10)

# Plotta täthetsfunktionen med den skattade parametern.
x_grid = np.linspace(np.min(y), np.max(y), 60)
pdf = stats.rayleigh.pdf(x_grid, scale=est)

plt.plot(x_grid, pdf, 'r')

plt.show()
```

Ser fördelningen ut att passa bra?

Rayleighfördelningen kan till exempel användas för att beskriva hur en radiosignal avtar. Experimentella mätningar på Manhattan har visat att Rayleighfördelningen beskriver radiosignalers fädning (engelska: fading) på ett bra sätt i den sortens stadsmiljö [1].

Problem 4 – Jämförelse av fördelningar hos olika populationer

I denna uppgift undersöker vi en datamängd visuellt med hjälp av Python för att se om vi kan göra några intressanta iakttagelser. Filen `birth.dat` innehåller data om 747 förstföderskor i Malmö under åren 1991–1993. En beskrivning av data finns i filen `birth.txt`. Filen innehåller olika 26 variabler, varav några är numeriska (såsom längd och vikt hos modern) och andra kategoriska, dvs. antar ett av 2–3 fixa värden (exempelvis `1` om barnet var planerat och `2` om det inte var planerat). Använd informationen i `birth.txt` och funktionen `plt.hist` för att generera en figur med fyra olika histogram som visar fördningarna för barnets födelsevikt, moderns ålder, moderns längd respektive moderns vikt.

Det är av medicinskt intresse att bestämma riskfaktorer som ökar sannolikheten för att ett barn föds med alltför låg födelsevikt. Låg födelsevikt definieras som en födelsevikt under 2500 g, mycket låg födelsevikt som en födelsevikt under 1500 g och extremt låg födelsevikt som en födelsevikt under 1000 g. Vi kan användra den givna datamängden för att försöka dra slutsatser om riskfaktorer för låg födelsevikt genom att jämföra viktfördelningen

för barn vars mödrar har en viss riskfaktor med viktfördelning för barn vars mödrar saknar riskfaktorn.

En känd riskfaktor för låg födelsevikt hos nyfödda är rökning. Vi undersöker därför skillnaden i födelsevikt mellan barn vars mödrar röker respektive inte röker under graviditeten. I filen `birth.txt` ser man att kolonn 20 i `birth.dat` innehåller rökvanor och att värdena `1` och `2` betyder att modern inte röker under graviditeten, medan värdet `3` betyder att hon gör det. Skapa två variabler `x` och `y` för födelsevikter hörande till icke-rökande respektive rökande mödrar enligt:

```
## Problem 4: Fördelningar av givna data

# Ladda datafilen.
birth = np.loadtxt('birth.dat')

# Definiera filter beroende på om modern röker (kolonn 20
# är 3) eller inte (kolonn 20 är 1 eller 2). Notera att
# eftersom indexering i Python börjar med noll så betecknas
# kolonn 20 med indexet 19.
non_smokers = (birth[:, 19] < 3)
smokers = (birth[:, 19] == 3)

# Extrahera födelsevikten (kolonn 3) för de två kategorierna.
x = birth[non_smokers, 2]
y = birth[smokers, 2]
```

Vad som händer här är att `birth[:, 19] < 3` returnerar en vektor av `True` och `False` och att bara de rader av kolonn 3 (födelsevikterna) i `birth` för vilka jämförelsen är sann, väljs ut. Använd funktionen `np.shape` eller kommandot `%whos` för att se storleken på vektorerna `x` och `y`.

Ett alternativ till histogram för att visualisera en datamängd är *kärnestimatorer*. Dessa interpolerar en täthetsfunktion för datat genom att "släta ut" det med hjälp av en viss kärnfunktion. Koden nedan använder sådana kärnestimatorer för att visualisera datat ovan.

```
## Problem 4: Fördelningar av givna data (forts.)

# Skapa en stor figur.
plt.figure(figsize=(8, 8))

# Plotta ett låddiagram över x.
plt.subplot(2, 2, 1)
plt.boxplot(x)
plt.axis([0, 2, 500, 5000])
```

```
# Plotta ett låddiagram över y.
plt.subplot(2, 2, 2)
plt.boxplot(y)
plt.axis([0, 2, 500, 5000])

# Beräkna kärnestimator för x och y. Funktionen
# gaussian_kde returnerar ett funktionsobjekt som sedan
# kan evalueras i godtyckliga punkter.
kde_x = stats.gaussian_kde(x)
kde_y = stats.gaussian_kde(y)

# Skapa ett rutnät för vikterna som vi kan använda för att
# beräkna kärnestimatorernas värden.
min_val = np.min(birth[:, 2])
max_val = np.max(birth[:, 2])
grid = np.linspace(min_val, max_val, 60)

# Plotta kärnestimatorerna.
plt.subplot(2, 2, (3, 4))
plt.plot(grid, kde_x(grid), 'b')
plt.plot(grid, kde_y(grid), 'r')

plt.show()
```

Vad betyder plottarna? Vilka slutsatser kan du dra om rökande mödrars påverkan på barns födelsevikt?

Välj nu ut en annan av de kategoriska variablerna i datat som du misstänker kan påverka födelsevikten och undersök med samma metod om det förefaller föreligga ett samband mellan variabeln och födelsevikten. Observera att om du väljer en kategorisk variabel med tre olika värden, så behöver du göra om den till en variabel med två olika värden på samma sätt som för rökvanevariabeln ovan.

En viktig sak att tänka på är att alla fält inte är ifyllda i `birth.dat`. Dessa översätts då till NaN (eng. "not a number") i Python, vilket kan skapa problem för vissa funktioner (se nedan). Funktionen `np.isnan` i NumPy kan användas för att sortera bort NaN-värden genom att bilda en mask. Till exempel kan uttrycket `x[~np.isnan(x)]` filtrera bort alla NaN-värden i vektorn `x`.

Problem 5 – Test av normalitet

Många statistiska metoder baseras på antagandet att datat är normalfördelat. Det är därför av intresse att kunna avgöra om en given datamängd är normalfördelad eller ej. En metod för att visuellt undersöka om en datamängd är normalfördelad ges av funktionen `stats.probplot` som jämför den

empiriska datamängdens kvantiler med kvantilerna för en normalfördelning. Denna funktion måste anropas på ett lite annorlunda sätt:

```
_ = stats.probplot(x, plot=plt)
```

Först behöver vi fånga returvärdet genom att tilldela variabeln `_`, annars visas detta i notebooken. (Variabelnamnet `_` används ofta per konvention i Python för att beteckna ett värde som inte kommer att användas.) Sedan, för att funktionen ska veta var plotten ska visas måste vi ge den plottmodulen `plt` som argument.

Funktionen `stats.probplot` har en bugg som innebär att den röda linjen inte visas om `x` innehåller ett NaN-värde. Kom därför ihåg att filtrera ut sådana värden genom att anropa `x[~np.isnan(x)]`.

Undersök med denna funktion om variablerna för barnets födelsevikt, moderns ålder, moderns längd eller moderns vikt kan sägas vara normalfördelade. Om inte, beskriv på vilket sätt de olika variablerna avviker från normalfördelning slumpvariabler.

Metoden `stats.probplot` bygger på att man gör en visuell bedömning av en graf och innehåller därför ett visst mått av subjektivitet. Det finns dock statistiska test för att avgöra normalitet. Ett sådant är Jarque–Beras test av normalitet som bygger på en jämförelse mellan det empiriska datats skevhet och kurtosis och motsvarande storheter för en normalfördelning. För en slumpvariabel X med väntevärde μ och standardavvikelse σ definieras skevheten γ och kurtosisen κ som

$$\gamma = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] \quad \text{respektive} \quad \kappa = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right].$$

I Jarque–Beras test av normaltitet används testvariabeln

$$JB = \frac{n}{6}\left(S^2 + \frac{1}{4}(K-3)^2\right),$$

där n är antalet observationer och S och K är skattningar av datats skevhet och kurtosis. Under nollhypotesen att datat är normalfördelat, så är testvariabeln approximativt χ^2-fördelad med två frihetsgrader.

Använd funktionen `stats.jarque_bera` för att avgöra om variablerna för barnets födelsevikt, moderns ålder, moderns längd och/eller moderns vikt är normalfördelade på signifikansnivån 5%.

Problem 6 - Konfidensintervall för skillnad mellan väntevärden för födelsevikter

Vi skall nu gå vidare till att studera skillnaden mellan väntevärden i två populationer, t. ex. skillnaden i födelsevikt för barn vars mammor röker

respektive inte röker under graviditeten. (Om ni vill kan ni ta två andra populationer eller andra variabler att studera!)

Som innan har vi vektorerna `x` och `y` som motsvarar födelsevikten för icke-rökande respektive rökande mammor. För att skatta skillnaden mellan populationernas väntevärde använder vi som vanligt skillnaden mellan stickprovesmedelvärdena:

```
np.mean(x) - np.mean(y)
```

Vad får du för konfidensintervall för skillnaden mellan väntevärdena?

Problem 7 - Enkel linjär regression

Linjär regression utvecklades under sent 1700-tal av en ung Gauss. Metoden fick ett genomslag när den förutspådde banan för den genom tiderna först upptäckta asteroiden Ceres. Linjär regression används än flitigare idag med tillämpningar inom i stort sett all vetenskap som behandlar data. Fördjupning i ämnet ges i kursen SF2930 Regressionsanalys.

I denna uppgift ska vi undersöka fenomenet Moores lag. Ladda in datat `moore.dat` på samma sätt som tidigare. I datat så är `y` antalet transistorer per ytenhet medan `x` representerar årtalet. Det betyder att om vi plottar dessa variabler mot varandra så ser vi en plot av utvecklingen över tid av antalet transistorer per ytenhet. Antalet transistorer per ytenhet förefaller öka exponentiellt, vilket gör det rimligt att anta att logaritmen av antalet transistorer per ytenhet ökar linjärt med tiden. Inför modellen

$$w_i = \log(y_i) = \beta_0 + \beta_1 x_i + \varepsilon_i.$$

Bilda en matris X som har formen

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix},$$

där n är antalet observationer i datat och ta sedan fram en skattning av $\beta = (\beta_0, \beta_1)$ med hjälp av funktionen `tools.regress`. Vi betecknar skattningen med $\hat{\beta}_0 = (\hat{\beta}_0, \hat{\beta}_1)$. Plotta din skattade modell

$$\log(\hat{y}) = X\hat{\beta},$$

genom att jämföra $\hat{y}$ med datat y. Detta görs enklast med ett punktdiagram (`plt.scatter`) för punkterna (x, y) där du sedan plottar linjen $\log(\hat{y}) = X\hat{\beta}$. Plotta sedan residualerna på följande sätt.

```
## Problem 6: Regression

# Bilda residualerna.
res = w - X @ beta_hat

# Skapa figur.
plt.figure(figsize=(4, 8))

# Plotta kvantil-kvantil-plot för residualerna.
plt.subplot(2, 1, 1)
_ = stats.probplot(res, plot=plt)

# Plotta histogram för residualerna.
plt.subplot(2, 1, 2)
plt.hist(res, density=True)

plt.show()
```

Vilken fördelning ser de ut att komma från? Om du skattar $\hat{\beta}$ med hjälp av data från 1972 till 2019, vad är då din prediktion för antalet transistorer år 2025?

Referenser

[1] Dmitry Chizhik, Jonathan Ling, Peter W. Wolniansky, Reinaldo A. Valenzuela, Nelson Costa, and Kris Huber (2003). Multiple-input-multiple-output measurements and modeling in Manhattan *IEEE Journal on Selected Areas in Communications*, Vol **21**, p. 321-331.

[2] Blom, G., Enger, J., Englund, G., Grandell, J., och Holst, L., (2005). Sannolikhetsteori och statistikteori med tillämpningar.