STA414 Assignment 0

Raj Patel

January 24, 2020

0.1 Probability

0.1.1 Variance and Covariance

Let X and Y be two continuous, independent random variables.

1. [3pts] Starting from the definition of independence, show that the independence of X and Y implies that their covariance is 0.

Answer:

By (i) Definition of Covariance, and (ii) X and Y are independent iff $E[g(X)h(Y)] = E[g(X)]E[h(Y)]$ whenever all expectations exist for g and h, we have:

$$\begin{aligned}
\text{Cov}(X, Y) &= \mathbb{E}[(X - \mathbb{E}(X))(Y - \mathbb{E}(Y))] \\
&= \mathbb{E}[XY - X\mathbb{E}(Y) - Y\mathbb{E}(X) + \mathbb{E}(X))\mathbb{E}(Y)] \\
&= \mathbb{E}(XY) - \mathbb{E}(Y)(\mathbb{E}(XY)) - \mathbb{E}(X)(\mathbb{E}(Y)) + \mathbb{E}(X)\mathbb{E}(Y) \\
&= \mathbb{E}(X)\mathbb{E}(Y) - \mathbb{E}(Y)(\mathbb{E}(XY)) - \mathbb{E}(X)(\mathbb{E}(Y)) + \mathbb{E}(X)\mathbb{E}(Y) \\
&= \boxed{0}
\end{aligned}$$

2. [3pts] For a scalar constant a, show the following two properties starting from the definition of expectation:

$$\mathbb{E}(X + aY) = \mathbb{E}(X) + a\mathbb{E}(Y) \tag{1}$$
$$\text{var}(X + aY) = \text{var}(X) + a^2\text{var}(Y) \tag{2}$$

Answer:

Part (1) By definition of expectation and the property of linearity of expectation, we have

$$\begin{aligned}
\mathbb{E}(X + aY) &= \mathbb{E}(X) + \mathbb{E}(aY) \\
&= \boxed{\mathbb{E}(X) + a\mathbb{E}(Y)}
\end{aligned}$$

Part (2) By definition of variance, we know that Var(X + Y) = Var(X) + Var(Y) + 2Cov(X,Y). Using that and the result from part 1 that for independent random variable X and Y, Cov(X,Y) = 0, we can write this as:

$$\begin{aligned}
\mathrm{Var}(X + aY) &= \mathrm{Var}(X) + \mathrm{Var}(aY) + 2\mathrm{Cov}(X, aY) \\
&= \mathrm{Var}(X) + \mathbb{E}[(aY - \mathbb{E}(aY))(aY - \mathbb{E}(aY))] + 2a\mathrm{Cov}(X, Y) \\
&= \mathrm{Var}(X) + \mathbb{E}[a(Y - \mathbb{E}(Y))a(Y - \mathbb{E}(Y))] + 2a\mathrm{Cov}(X, Y) \\
&= \mathrm{Var}(X) + a^2\mathbb{E}[(Y - \mathbb{E}(Y))(Y - \mathbb{E}(Y))] + 2a\mathrm{Cov}(X, Y) \\
&= \mathrm{Var}(X) + a^2\mathrm{Var}(Y) + 2a\mathrm{Cov}(X, Y) \\
&= \boxed{\mathrm{Var}(X) + a^2\mathrm{Var}(Y)}
\end{aligned}$$

0.1.2 1D Gaussian Densities

1. [1pts] Can a probability density function (pdf) ever take values greater than 1?

Answer:

Yes. For a uniform distribution on [a,b], the density is $\frac{1}{b-a}$. So, if b=0.5 and a=0, then density is 2 in the region [0,0.5].

2. Let X be a univariate random variable distributed according to a Gaussian distribution with mean μ and variance σ^2.

[1pts] Write the expression for the pdf:

Answer:

For gaussian distribution, pdf is: $\frac{1}{(2\pi\sigma^2)^{0.5}} exp(-\frac{(x-\mu)^2}{2\sigma^2})$

[2pts] Write the code for the function that computes the pdf at x with default values $\mu = 0$ and $\sigma = \sqrt{0.01}$:

Answer:

```
function gaussian_pdf(x; mean=0., variance=0.01)
  return (1/sqrt(2*pi*variance)) * exp((-1/(2*variance))*((x - mean)^2))
end
```

```
gaussian_pdf (generic function with 1 method)
```

Test your implementation against a standard implementation from a library:

```julia
# Test answers
using Test
using Distributions: pdf, Normal
using Random # Import Random Package

Random.seed!(414); #Set Random Seed
@testset "Implementation of Gaussian pdf" begin
  x = randn()
  @test gaussian_pdf(x) ≈ pdf.(Normal(0.,sqrt(0.01)),x)
  @test isapprox(gaussian_pdf(x) , pdf.(Normal(0., sqrt(0.01)),x))
  @test isapprox(gaussian_pdf(x,mean=10., variance=1) , pdf.(Normal(10., sqrt(1)),x)) #
checking non-default values

end
```

```
Test Summary:                   | Pass  Total
Implementation of Gaussian pdf |    3      3
Test.DefaultTestSet("Implementation of Gaussian pdf", Any[], 3, false)
```

3. [1pts] What is the value of the pdf at $x = 0$? What is probability that $x = 0$ (hint: is this the same as the pdf? Briefly explain your answer.)

Answer:

Part 1: Based on the function we created, pdf at x=0 is:

```julia
gaussian_pdf(0)
```

```
3.989422804014327
```

While analytically, pdf is: $\frac{1}{(2\pi\sigma^2)^{0.5}} exp(-\frac{(x-\mu)^2}{2\sigma^2})$. We know that x=0,mean=0 and variance=0.01. So, based on that, the pdf becomes: $\frac{10}{\sqrt{2\pi}}$ which is same as the value we get from the function created.

Part 2: Since x is continuous, P(x=0) = 0.

4. A Gaussian with mean μ and variance σ^2 can be written as a simple transformation of the standard Gaussian with mean 0. and variance 1..

[1pts] Write the transformation that takes $x \sim \mathcal{N}(0., 1.)$ to $z \sim \mathcal{N}(\mu, \sigma^2)$:

Answer:

If $x \sim N(0., 1.)$ and $z \sim N(\mu, \sigma^2)$, then we can write the relation between them as:

$x = \frac{z-\mu}{\sigma}$
$\therefore \boxed{z = \mu + \sigma x}$

[2pts] Write a code implementation to produce n independent samples from $\mathcal{N}(\mu, \sigma^2)$ by transforming n samples from $\mathcal{N}(0., 1.)$.

Answer:

```
using Distributions: Normal
function sample_gaussian(n; mean=0., variance=0.01)
  # getting n samples from standard gaussian
  x = randn(n)

  # transform x to z with N(mean,variance)
  z = x*sqrt(variance) .+ mean
  return z
end;
```

[2pts] Test your implementation by computing statistics on the samples:

Answer:

```
using Test
using Distributions: Normal
using Statistics
@testset "Numerically testing Gaussian Sample Statistics" begin

  #  Test1:
  # Sampling 1000000 from the normal distribution with mean = 0 and variance = 0.01
  x = sample_gaussian(100000; mean=0., variance=0.01)

  #  Declaring true mean and variance
  x_mean = 0.
  x_var = 0.01

  # Testing true mean and variance with the numerical statistics
  @test isapprox(Statistics.mean(x) , x_mean, atol=1e-2)
  @test isapprox(var(x) , x_var, atol=1e-2)

  # Test2:
  # Sampling 1000000 from the normal distribution with mean = 3 and variance = 1
  y = sample_gaussian(100000; mean=3., variance=1.)

  #  Declaring true mean and variance
  y_mean = 3.
  y_var = 1.

  # Testing true mean and variance with the numerical statistics
  @test isapprox(Statistics.mean(y) , y_mean, atol=1e-2)
  @test isapprox(var(y) , y_var, atol=1e-2)
end;
```

```
Test Summary:                                   | Pass  Total
Numerically testing Gaussian Sample Statistics |    4      4
```

5. [3pts] Sample 10000 samples from a Gaussian with mean 10. an variance 2. Plot the **normalized** `histogram` of these samples. On the same axes `plot!` the pdf of this distribution.

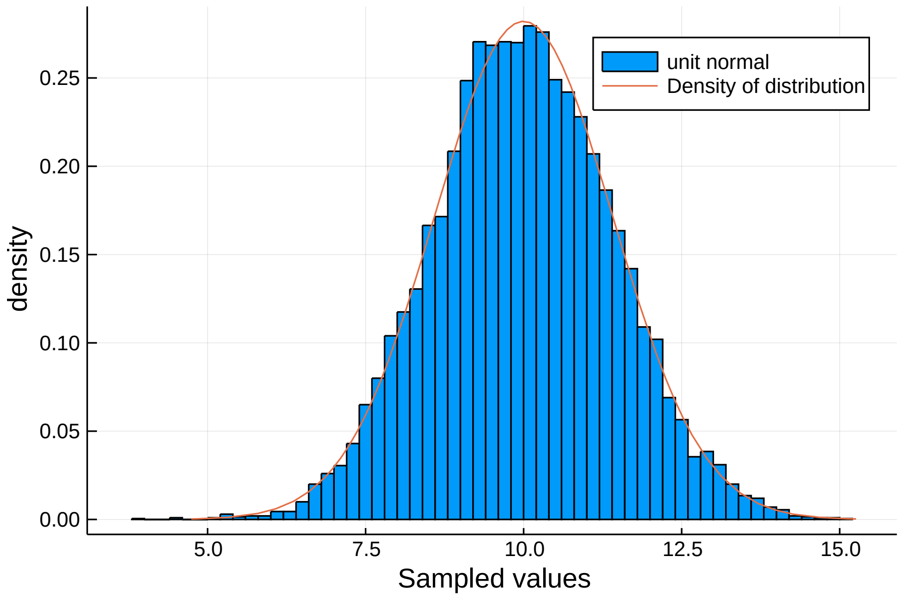
Confirm that the histogram approximates the pdf.

Answer:

```
function gaussian_pdf_temp(x; mean=0., variance=0.01)
  return (1/sqrt(2*pi*variance)) * exp.((((x .- mean).^2) .* (-1/(2*variance))))
end

using Plots
using Distributions
using StatPlots

histogram(sample_gaussian(10000; mean = 10.,variance = 2.),normalize=true , label="unit
normal")
plot!(Normal(10,sqrt(2)),  label = "Density of distribution")
```



0.2 Calculus

0.2.1 Manual Differentiation

Let $x, y \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$, and square matrix $B \in \mathbb{R}^{m \times m}$. And where x' is the transpose of x. Answer the following questions in vector notation.

1. [1pts] What is the gradient of $x'y$ with respect to x?

Answer: $\nabla_x x^T y = [\frac{d \sum_{k=1}^{m} x_k y_k}{dx_i}] = [y_i] = \boxed{y}$

2. [1pts] What is the gradient of $x'x$ with respect to x?

Answer: $\nabla_x x^T x = [\frac{d \sum_{k=1}^{m} x_k^2}{dx_i}] = [2x_i] = \boxed{2x}$

3. [2pts] What is the Jacobian of $x'A$ with respect to x?

Answer:

J = Jacobian of ($f_3 = [\sum_{=1}^{m} x_i A_{i1} ... \sum_{k=1}^{m} x_i A_{in}]$)

$$J = \begin{bmatrix} \frac{d\sum_{k=1}^{m} x_k A_{k1}}{dx_1} & \dots & \frac{d\sum_{k=1}^{m} x_k A_{k1}}{dx_m} \\ \vdots & \ddots & \vdots \\ \frac{d\sum_{k=1}^{m} x_k A_{kn}}{dx_1} & \dots & \frac{d\sum_{k=1}^{m} x_k A_{kn}}{dx_m} \end{bmatrix} = \begin{bmatrix} A_{11} & \dots & A_{m1} \\ \vdots & \ddots & \vdots \\ A_{1n} & \dots & A_{mn} \end{bmatrix} = \boxed{A^T}$$

4. [2pts] What is the gradient of $x'Bx$ with respect to x?

Answer: $\nabla_x x^T Bx = [\frac{d\sum_{k=1}^{m}\sum_{j=1}^{m} x_j b_{jk} x_k}{dx_i}] = [\sum_{k=1}^{m} b_{jk}x_k + \sum_{j=1}^{m} b_{jk}x_j] = Bx + B^T x = \boxed{(B + B^T)x}$

0.2.2 Automatic Differentiation (AD)

Here, we use an accepted AD library i.e., Zygote.jl (julia) to implement and test our answers.

[1pts] Create Toy Data Answer:

```
# Choosing dimensions of toy data
m = 3
n = 4

# Making random toy data with correct dimensions

x = vec(rand(3,1)) # as x is a column vector
y = vec(rand(3,1)) # as y is a  column vector
A = rand(3,4) # A is a m * n Matrix
B = rand(3,3) # B is a m * m Matrix
```

```
3×3 Array{Float64,2}:
 0.733746    0.442975  0.973124
 0.0918403   0.744545  0.0769872
 0.771936    0.943475  0.907889
```

[1pts] Test to confirm that the sizes of your data is what you expect:

Here, we want to make sure that our toy data is the size we expect. If we use size function on a column vector, it gives us one value which is the number of rows in a column vector. Based on that, we test for number on rows in vector x and y as they are column vectors.

Following that, we use size() function on a matrix. When we use size() function on a matrix, it gives us a vector of 2 values where the first value is the number of rows and the second is column. Based on that, we test for number of rows and columns in matrix A and B using size() function.

```
@testset "Sizes of Toy Data" begin
```

```julia
  @test size(x)[1] == 3 # Rows in column vector x
  @test size(y)[1] == 3 # Rows in column vector y

  @test size(A)[1] == 3 # Rows in matrix A
  @test size(A)[2] == 4 # Columns in matrix A
  @test size(B)[1] == 3 # Rows in matrix B
  @test size(B)[2] == 3 # Columns in matrix B

end;
```

```
Test Summary:      | Pass  Total
Sizes of Toy Data |    6      6
```

Automatic Differentiation

1. [1pts] Compute the gradient of $f_1(x) = x'y$ with respect to x?

Here, zygote, while computing gradient, returns a tuple of gradients, one for each argument. But, we just want with respect to "x", which is the first element in all our cases. As a result, we will index the tuple of gradient with [1] to get gradient with respect to x in all the following questions where we are asked to compute gradients.

```julia
# Use Auto Differentiation Tool
using Zygote: gradient

f1(x,y) = x'*y
df1dx = gradient(f1,x,y)[1]
df1dx
```

```
3-element Array{Float64,1}:
 0.47416145867287285
 0.6762641934796711
 0.9955240023164769
```

2. [1pts] Compute the gradient of $f_2(x) = x'x$ with respect to x?

```julia
f2(x) = x'*x
df2dx = gradient(f2,x)[1]
df2dx
```

```
3-element Array{Float64,1}:
 1.7632915257534125
 0.7789011893968381
 0.5330367155704199
```

3. [1pts] Compute the Jacobian of $f_3(x) = x'A$ with respect to x?

If we try the usual `gradient` function to compute the whole Jacobian, it would give an error. So, we use the following code to compute the Jacobian instead.

```
function jacobian(f, x)
    y = f(x)
    n = length(y)
    m = length(x)
    T = eltype(y)
    j = Array{T, 2}(undef, n, m)
    for i in 1:n
        j[i, :] .= gradient(x -> f(x)[i], x)[1]
    end
    return j
end
```

```
jacobian (generic function with 1 method)
```

```
f3(x) = (x'*A)
df3dx = jacobian(x -> f3(x), x)
df3dx
```

```
4×3 Array{Float64,2}:
 0.104215  0.257865  0.380962
 0.958234  0.716282  0.602106
 0.138519  0.987647  0.33412
 0.957227  0.895428  0.165908
```

[2pts] Briefly, explain why `gradient` of f_3 is not well defined (hint: what is the dimensionality of the output?) and what the `jacobian` function is doing in terms of calls to `gradient`. Specifically, how many calls of `gradient` is required to compute a whole `jacobian` for $f : \mathbb{R}^m \to \mathbb{R}^n$?

Answer:

- Because the function we are supposed to apply gradient on is a row vector and not a scalar, gradient is not well defined. In this particular case, we want to find Jacobian of x'A. Now, looking at the dimensions, x'A is of the dimension $1 \times n$. As that is not a scalar, we can not use gradient function on x'A.
- Mathematically, the gradient of f_3 is not well defined because it is a map from $\mathbb{R}^m \to \mathbb{R}^n$. It is only well defined for maps $\mathbb{R}^m \to \mathbb{R}$.
- To understand what the function Jacobian does in terms of calling gradient function, we first observe that in our case, the Jacobian for f_3, is given by a column vector with dimension $n \times 1$, where i-th element is gradient of i-th element of row vector f_3.
- So, what the jacobian function does is that for each element of the row vector of the result we input (i.e., in this case $x'A$), it computes the gradient with respect to x. The resulting vector is used as the respective row for the output of jacobian function.

- As the dimension of f_3 is $1 \times n$, to calculate the complete Jacobian, we need to call 'gradient' function n times.

4. [1pts] Compute the gradient of $f_4(x) = x'Bx$ with respect to x?

```
f4(x, B) = x' * B * x
df4dx = gradient(f4,x,B)[1]
df4dx
```

```
3-element Array{Float64,1}:
 1.9671826903223844
 1.3234166938401206
 2.4198825329901754
```

5. [2pts] Test all your implementations against the manually derived derivatives in previous question

```
# Test to confirm that AD matches hand-derived gradients
@testset "AD matches hand-derived gradients" begin
  @test df1dx == y
  @test df2dx == 2*x
  @test df3dx == A'
  @test df4dx  == (B' * x + B * x)[:]
end
```

```
Test Summary:                      | Pass  Total
AD matches hand-derived gradients |    4      4
Test.DefaultTestSet("AD matches hand-derived gradients", Any[], 4, false)
```

```
using Weave
weave("A0.jmd", doctype = "md2pdf")
```