Assignment 0

Hanyu Qi and 1003046250

January 24, 2020

```julia
using Weave
weave("/Users/mac/Desktop/Julia/A0.jmd", doctype = "md2pdf")
```

1 Probability

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 the definition of independence: X and Y are independent if and only if $P(X,Y) = P(X)P(Y)$

By the definition of covariance: $Cov(X,Y) = E(XY) - E(X)E(Y)$

By the definition of expectation:

$$E(X) = \int_R xf(x)dx$$

where $f(x)$ is pdf of X.

$$E(Y) = \int_R yf(y)dy$$

where $f(y)$ is pdf of Y.

$$E(XY) = \int_R \int_R xyf(x,y)dxdy$$

where $f(x,y)$ is joint pdf of XY.

Since X and Y are independent, we can write their pdf as $f(x,y) = f(x)f(y)$

So we can write $E(XY) = \int_R \int_R xyf(x,y)dxdy = \int_R xf(x)dx \int_R yf(y)dy = E(X)E(Y)$

Therefore, $Cov(X,Y) = E(XY) - E(X)E(Y) = E(X)E(Y) - E(X)E(Y) = 0$ if X and Y are independent.

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

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

Answer:

We know $E(X + aY) = \int_R \int_R (x + ay) f(x, y) dxdy$ where $f(x, y)$ is joint pdf of XY.

We can rewrite it as $\int_R \int_R (x + ay) f(x, y) dxdy = \int_R \int_R x f(x, y) dxdy + \int_R \int_R ay f(x, y) dxdy$

Since X and Y are independent, so $f(x, y) = f(x)f(y)$ and we get $\int_R x f(x)dx \int_R f(y)dy + a \int_R y f(y)dy \int_R f(x)dx$

Since $\int_R f(x)dx = 1$ and $\int_R f(y)dy = 1$, we get $E(X + aY) = E(X) + aE(Y)$

By the definition of variance, $var(X + aY) = E((X + aY)^2) - E^2(X + aY)$

We can expand it as $E(X^2 + 2aXY + a^2Y^2) - [E(X) + aE(Y)]^2$

We know $E(X^2 + 2aXY + a^2Y^2) = E(X^2) + 2aE(XY) + a^2E(Y^2)$ and $[E^2(X) + aE(Y)]^2 = E^2(X) + 2aE(X)E(Y) + a^2E^2(Y^2)$ by above.

Since X and Y are independent, we know $E(XY) = E(X)E(Y)$

So $var(X + aY) = E(X^2) - E^2(X) + a^2[E(Y^2) - E^2(Y)] = var(X) + a^2 var(Y)$

1.2 1D Gaussian Densities

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

Answer:

Yes. Since the value of the probability density function is not the probability.

There is no constraint on the range. The value is the height of pdf when $X = x$, so the values can be greater than 1.

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:

$$f(x) = \frac{1}{\sqrt{(2\pi\sigma^2)}} e^{\frac{-1}{2}(\frac{x-\mu}{\sigma})^2}$$

[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*variance*π))).*exp((-1/2)*((x.-mean)./sqrt(variance))^2)
  end
```

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

Test your implementation against a standard implementation from a library:

```
# Test answers
        using Test
        using Random
        using Distributions: pdf, Normal # Note Normal uses N(mean, stddev) for
parameters
        @testset "Implementation of Gaussian pdf" begin
          x = randn()
          @test 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))
        end;
```

```
Test Summary:                   | Pass  Total
Implementation of Gaussian pdf |    2      2
```

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

When $\mu = 0$ and $\sigma = 0.1$, the value of the pdf of continuous random variable X at $x = 0$ is $f(X = 0) = \frac{1}{0.1\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{0}{0.1})^2} = 3.99$ where $f(x)$ is pdf of X.

So the value of the pdf at $x = 0$ is the value of pdf $f(x)$ when $x = 0$.

The probability that $x = 0$ is $P(X = 0) = 0$.

```
gaussian_pdf(0)
```

```
3.989422804014327
```

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

$$x \sim \mathcal{N}(0., 1.)$$

By normalization we get $\frac{x-\mu}{\sigma} = z$ and $z \sim \mathcal{N}(\mu, \sigma^2)$

So we can write it as $z = \mu + x\sigma$

[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

```
function sample_gaussian(n; mean=0, variance=0.01)
            # n samples from standard gaussian mean = 0, variance = 1
            x = randn(n)
            # transform x to sample z from N(mean,variance)
            z = (sqrt(variance).*x).+mean
            return z
            end;
```

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

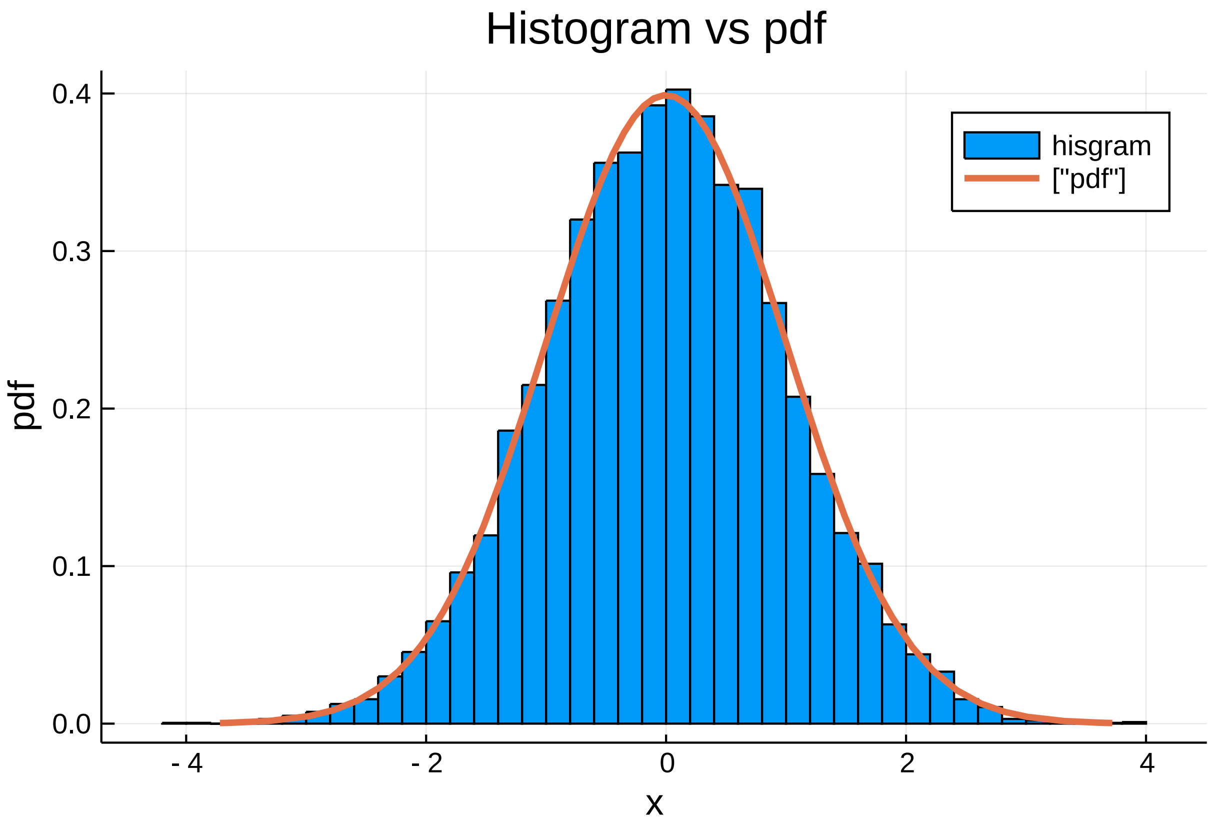
```
using Test
            using Statistics: mean, var
            @testset "Numerically testing Gaussian Sample Statistics" begin
            @test isapprox(mean(sample_gaussian(100000)),0;atol=1e-2)
            @test isapprox(var(sample_gaussian(100000)),0.01;atol=1e-2)
            end;
```

```
Test Summary:                                  | Pass  Total
Numerically testing Gaussian Sample Statistics |    2      2
```

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. (Note: with `Plots.jl` the function `plot!` will add to the existing axes.)

```
using Plots
            using Distributions
            using StatsPlots
            x=sample_gaussian(10000;mean=10.,variance=2)
            z=(x.-10.)./sqrt(2)
            histogram(z,normalize=:pdf,label="hisgram")
            plot!(Normal(0,1),lw=3,title="Histogram vs pdf",label=["pdf"])
            xlabel!("x")
            ylabel!("pdf")
```



2 Calculus

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:

$$x'y = \Sigma_{i=1}^{m} x_i y_i$$
$$\frac{\partial x'y}{\partial x_k} = y_k, \forall k = 1, 2, ..., m$$
$$\frac{\partial x'y}{\partial x} = y'$$

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

Answer:

$$x'x = \Sigma_{i=1}^{m} x_i^2$$

$$\frac{\partial x'x}{\partial x_k} = 2x_k, \forall k = 1, 2, ..., m$$

$$\frac{\partial x'x}{\partial x} = 2x'$$

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

Answer:

Let $z' = x'A$ and a_j be jth column of A, then $z_i' = x'a_j$

$$\frac{\partial z_i}{\partial x} = a_j{}'$$

$$\frac{\partial z'}{\partial x} = A'$$

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

Answer:

$$x'Bx = \Sigma_{j=1}^m \Sigma_{i=1}^m b_i j x_i x_j$$

$$\frac{\partial x'Bx}{\partial x_k} = \Sigma_{j=1}^m a_{kj} x_j + \Sigma_{i=1}^m a_{ik} x_i, \forall k = 1, 2, ..., m$$

$$\frac{\partial x'Bx}{\partial x} = x'B' + x'B = x'(B' + B)$$

2.2 Automatic Differentiation (AD)

Use one of the accepted AD library (Zygote.jl (julia), JAX (python), PyTorch (python)) to implement and test your answers above.

2.2.1 [1pts] Create Toy Data

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

# Make random toy data with correct dimensions
x = rand(m)
y = rand(m)
A = rand(m,n)
B = rand(m,m)
```

```
4×4 Array{Float64,2}:
 0.0858934  0.761271  0.1776     0.659769
 0.998544   0.371376  0.0843238  0.909156
 0.502962   0.780813  0.675547   0.814322
 0.348606   0.413332  0.187497   0.971796
```

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

```julia
using Test
# Make sure your toy data is the size you expect!
@testset "Sizes of Toy Data" begin
  @test size(x) == (m,)
  @test size(y) == (m,)
  @test size(A) == (m,n)
  @test size(B) == (m,m)
end;
```

```
Test Summary:      | Pass  Total
Sizes of Toy Data |    4      4
```

2.2.2 Automatic Differentiation

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

```julia
# Use AD Tool
using Zygote: gradient
# note: `Zygote.gradient` returns a tuple of gradients, one for each argument.
# if you want just the first element you will need to index into the tuple with [1]
f1(x) = x'y
df1dx = gradient(f1, x)[1]'
```

```
1×4 LinearAlgebra.Adjoint{Float64,Array{Float64,1}}:
 0.691187  0.716974  0.90877  0.664992
```

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]'
```

```
1×4 LinearAlgebra.Adjoint{Float64,Array{Float64,1}}:
 1.37122  0.39939  1.32518  0.792152
```

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

If you try the usual `gradient` fucntion to compute the whole Jacobian it would give an error. You can use the following code to compute the Jacobian instead.

```julia
function jacobian(f, x)
    y = f(x)
    n = length(y) #3
    m = length(x) #4
    T = eltype(y) #Int64
    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)
```

```julia
f3(x) = transpose(x)*A
df3dx = jacobian(f3,x)
```

```
3×4 Array{Float64,2}:
 0.457949  0.283911  0.34086   0.266517
 0.745013  0.308664  0.755715  0.141643
 0.108956  0.896636  0.908215  0.737167
```

[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:

The very important takeaway here is that, with AD, `gradient`s are cheap but full `jacobian`s are expensive.

Since x' is a vector of $1 \times m$ and A is a matrix of $m \times n$ so the output $x'A$ is a vector of $1 \times n$ instead of a 1×1 vector.

So if we want to calculate the differentation of it, we need to calculate the jacobian matrix of it.

Therefore, we need to take derivative of each row of $x'A$ with respect to x. From the code, we know we need to take n times of gradient to compute a whole jacobian for $f : \mathbb{R}^m \to \mathbb{R}^n$.

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

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

```
1×4 LinearAlgebra.Adjoint{Float64,Array{Float64,1}}:
 1.31953  2.45191  1.93138  2.38905
```

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

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

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