Solution to Assignment 3

MATH7502 2019, Semester 2

Assignment 3 questions (https://courses.smp.uq.edu.au/MATH7502/2019/ass3.pdf)

Solution to Question 1

(a) The weighted least squares problem can be writen as trying to approximatly solve the equations $Ax = b$ where each row (each i'th equation) is scaled by a strictly postive w_i. This can be written as minimization of

$$||DAx - Db||^2 = ||D(Ax - b)||^2,$$

where D is an $m \times m$ diagonal matrix with diagonal elements $\sqrt{w_1}, \ldots, \sqrt{w_m}$. Thus it is a the standard least squares problem minimizing $||Bx - d||^2$ where $B = DA$ and $d = Db$.

(b) Take an $\tilde{x}$ with $DA\tilde{x} = 0$ (we aim to show that $\tilde{x}$ must be the 0 n-vector). Since D is square and non-singular we can multiply both sides by D^{-1} to get, $A\tilde{x} = 0$. However, by assumption A has linearly independent collumns then the only solution to $Ax = 0$ is $x = 0$. This means that $\tilde{x} = 0$. Hence the only element in the null-space of DA is 0 and hence DA has linearly independent collumns.

(c) Using B and d we have

$$\hat{x} = (B^T B)^{-1} B^T d = (A^T D^T DA)^{-1} A^T D^T Db = (A^T WA)^{-1} A^T Wb,$$

where $W = D^T D = D^2$ is the diagonal matrix of weights.

Solution to Question 2

(a) We know that the $\hat{x}$ that solves the normal equations $A^T Ax = A^T b$, is given by

$$\hat{x} = A^\dagger b = R^{-1} Q^T b.$$

Now $A\hat{x}$ is the "predicated" value, closest to b within the collumn space of A. Using $A = QR$ it is given by,

$$A\hat{x} = QR\, R^{-1} Q^T b = QQ^T b.$$

(b) Using the above and $||u-v||^2 = u^T u - 2u^T v + v^T v$, we have,

$$||A\hat{x} - b||^2 = ||QQ^T b - b||^2 = (QQ^T b - b)^T(QQ^T b - b) = (QQ^T b)^T QQ^T b - 2b^T QQ^T b + b^T b.$$

This equals:

$$b^T QQ^T QQ^T b - 2(Q^T b)^T Q^T b + ||b||^2$$

and using $Q^T Q = I$ we have,

$$b^T QQ^T b - 2(Q^T b)^T Q^T b + ||b||^2 = (Q^T b)^T Q^T b - 2(Q^T b)^T Q^T b + ||b||^2 = -||Q^T b||^2 + ||b||^2,$$

as desired.

Solution to Question 3

For $x = (x_1, x_2)$, we aim to fit: $\hat{f}(x) = a + b_1 x_1 + b_2 x_2 + c_1 x_1^2 + c_2 x_2^2 + c_3 x_1 x_2$

(a) We have $f^{(1)}(x), \ldots, f^{(6)}(x)$ as follows:

$$\begin{aligned} f^{(1)}(x) &= 1 \\ f^{(2)}(x) &= x_1 \\ f^{(3)}(x) &= x_2 \\ f^{(4)}(x) &= x_1^2 \\ f^{(5)}(x) &= x_2^2 \\ f^{(6)}(x) &= x_1 x_2 \end{aligned}$$

Now the i'th row of the $n \times 6$ design matrix A has the form

$$[f^{(1)}(x)\ \ f^{(2)}(x)\ \ f^{(3)}(x)\ \ f^{(4)}(x)\ \ f^{(5)}(x)\ \ f^{(6)}(x)],$$

where x is the i'th data point.

Further,

$$\beta = [a\ \ b_1\ \ b_2\ \ c_1\ \ c_2\ \ c_3]^T.$$

Then given data x and y we wish to minimize,

$$||A\beta - y||^2.$$

(b)

In [1]:

```
f1(x) = 1
f2(x) = x[1]
f3(x) = x[2]
f4(x) = x[1]^2
f5(x) = x[2]^2
f6(x) = x[1]*x[2]
fs = [f1,f2,f3,f4,f5,f6] #an array of functions
f(x,j) = fs[j](x)
```

Out[1]:

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

In [2]:

```
using LinearAlgebra
x1Grid = -5:0.1:5
x2Grid = -5:0.1:5
n = length(x1Grid)*length(x2Grid)
xvals = [[x1,x2] for x1 in x1Grid, x2 in x2Grid];
xflat = reshape(xvals,n)
A = [f(x,j) for  x in xflat, j in 1:6];

AA = [8 3; 3 11]; d = [1,1];
yOfX(x) = (x-d)'*AA*(x-d) + 30cos(10x[1])*cos(10x[2])
y = yOfX.(xflat);
betaHat = pinv(A)*y

println("Estimates of (a,b1,b2,c1,c2,c3):  ", betaHat)
```

```
Estimates of (a,b1,b2,c1,c2,c3):  [24.9965, -22.0, -28.0, 8.00025, 11.0
002, 6.0]
```

(c) Observe that A is symmetric and consider $(x-d)^T A(x-d)$. Expanding it equals,

$$x^T Ax - 2x^T Ad + d^T Ad.$$

Now treating the entries of A and d in the standard manner $(a_{11}, a_{12}, a_{22}, d_1, d_2)$ this equals,

$$a_{11}x_1^2 + 2a_{12}x_1x_2 + a_{22}x_2^2 - 2(a_{11}d_1 + a_{12}d_2)x_1 - 2(a_{12}d_1 + a_{22}d_2)x_2 + d_1(a_{11}d_1 + a_{12}d_2) + d_2(a_{12}d_1 +$$

These coefficients may now be equated with the estimated a, b and c values. For simplificty let's round the estimated values: to (25,-22,-28,8,11,6). This also kills the 'noise' from 30cos(10x[1])*cos(10x[2]).

We now have:

$$a_{11} = c_1 = 8.$$

$$a_{22} = c_2 = 11.$$

$$2a_{12} = c_3 = 6 \quad \Rightarrow \quad a_{12} = 3.$$

Hence we see the matrix A is exactly reconstructed (this works because it is symmetric).

Further:

$$-2(a_{11}d_1 + a_{12}d_2) = b_1 = -22 \quad \Rightarrow \quad 8d_1 + 3d_2 = 11$$

Similarly,

$$-2(a_{12}d_1 + a_{22}d_2) = b_2 = -28 \quad \Rightarrow \quad 3d_1 + 11d_2 = 14.$$

Solving these linear equations for d_1 and d_2 we get $d_1 = 1$ and $d_2 = 1$.

Hence we see that by removing the noise we are **exactly** able to reconstruct the matrix A and vector d!

Solution to Question 4

We first experiment numerically to determine the eigenvalues and then show analytically that these hold.

In [3]:
```
using LinearAlgebra
eigvals(ones(1,1))
```

Out[3]:
```
1-element Array{Float64,1}:
 1.0
```

In [4]:
```
eigvals(ones(2,2))
```

Out[4]:
```
2-element Array{Float64,1}:
 0.0
 2.0
```

In [5]:
```
eigvals(ones(3,3))
```

Out[5]:
```
3-element Array{Float64,1}:
 -5.624168597199657e-16
  7.305347407387203e-18
  2.9999999999999996
```

In [6]:
```
eigvals(ones(4,4))
```

Out[6]:
```
4-element Array{Float64,1}:
 -5.660001591138239e-16
 -1.2325951644078312e-32
  1.0888646801245488e-17
  3.999999999999999
```

In [7]:
```
eigvecs(ones(4,4))
```

Out[7]:
```
4×4 Array{Float64,2}:
 -0.408248   0.707107     -0.288675  -0.5
 -0.408248  -0.707107     -0.288675  -0.5
  0.816497  -8.75605e-17  -0.288675  -0.5
  0.0        0.0           0.866025  -0.5
```

Hence we believe:

Set $A = \mathbf{1}\mathbf{1}^T$ an $n \times n$ matrix of all 1's. Then the eigenvalues are $n, 0, 0, \ldots, 0$. The fact that $n-1$ of the eigenvalues are 0 is not surprising. This is because the rank of the matrix is 1 and hence the rank of the null-space is $n-1$. This means that to get,

$$Ax = 0x = 0$$

we can take $x \neq 0$ as any of the vectors from an $n-1$ dimensional sub-space which is the null-space.

Now for the eigenvalue equalling n we can guess than an eigenvector is $\mathbf{1}$. Observe:

$$\mathbf{1}\mathbf{1}^T\mathbf{1} = n\mathbf{1}.$$

Hence n is an eigenvalue.

Note that an alternative way to solve this problem is to directly consider $\det(\mathbf{1}\mathbf{1}^T - \lambda I)$. Using determinant operations it can be shown to be,

$$\lambda^{n-1}(n-\lambda).$$

Here is a crude computational check:

In [8]:
```
characteristicPolynomial1(λ,n) = det(ones(n,n) - λ*I)
characteristicPolynomial2(λ,n) = λ^(n-1)*(n-λ)
n = 5
lamGrid = -2n:0.1:2n
maximum(abs.(characteristicPolynomial1.(lamGrid,n) .- characteristicPoly
nomial2.(lamGrid,n)))
```

Out[8]:
```
2.9103830456733704e-11
```

Solution to Question 5

By trying out several attempts it appears that the mean spectral radius is $n/2$. That is:

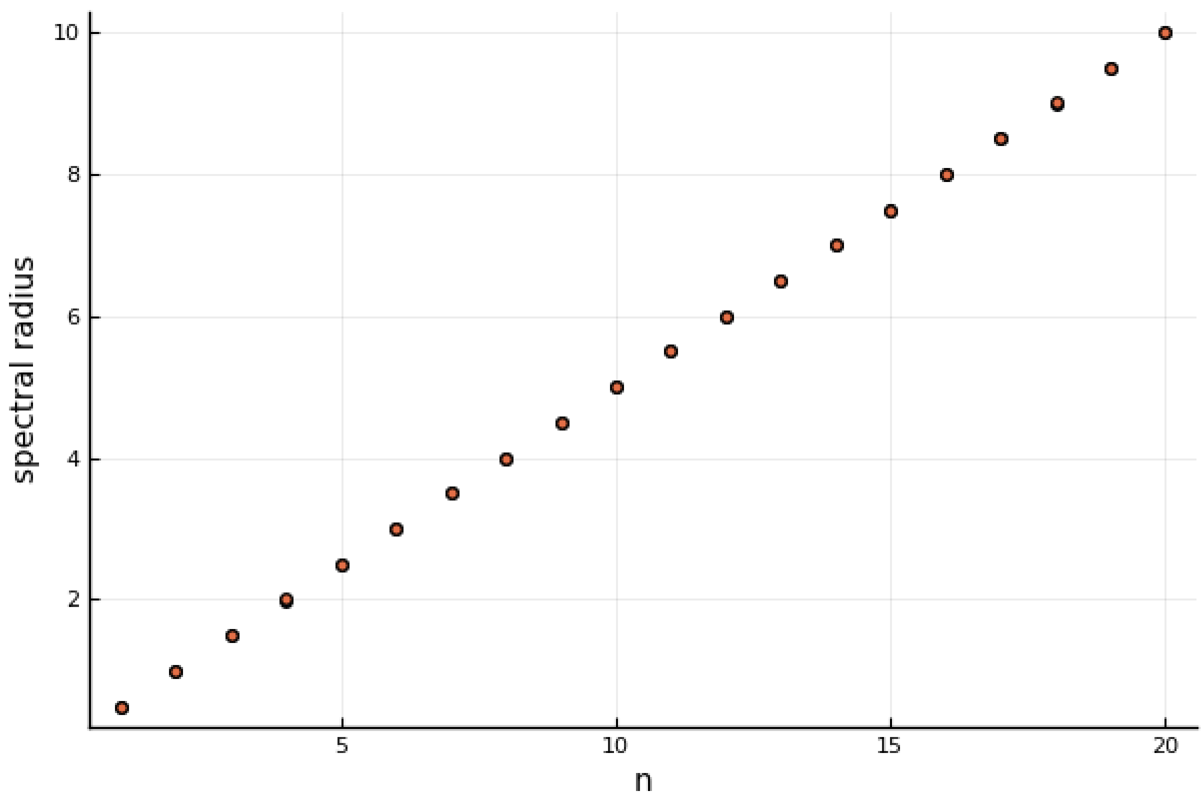
Conjecture: Take an $n \times n$ matrix with entries that are i.i.d. uniform(0,1) values. Denote the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$. And denote

$$\sigma = \max_{i=1,\ldots,n} |\lambda_i|.$$

The $E[\sigma] = n/2$.

In [9]:
```
using Random, LinearAlgebra, Plots, Statistics
pyplot()
Random.seed!(0)

N = 10^4
nRange = 1:20

eigR(n) = maximum(abs.(eigvals(rand(n,n))))
meanEst(n) = mean([eigR(n) for _ in 1:N])

ests = [meanEst(n) for n in nRange]
conj(n) = n/2

scatter(nRange,[ests,conj.(nRange)],legend = false, xlabel = "n", ylabel
= "spectral radius")
```

Out[9]:



Solution to Question 6

(a) To compute the eigenvalues consider the characteristic polynomial and equate to 0 (this should hold for every $\theta):

$$(\cos\theta - \lambda)^2 + \sin^2\theta = 0$$

or,

$$\cos^2\theta - 2\lambda\cos\theta + \lambda^2 + \sin^2\theta = 0$$

or

$$\lambda^2 - 2(\cos\theta)\lambda + 1 = 0$$

Hence,

$$\lambda_{1,2} = \cos\theta \pm \frac{1}{2}\sqrt{4\cos^2\theta - 4} = \cos\theta \pm \sqrt{\cos^2\theta - 1} = \cos\theta \pm \sqrt{-\sin^2\theta} = \cos\theta \pm i\left|\sin\theta\right| = \cos\theta \pm$$

Remember $e^{i\theta} = \cos\theta + i\sin\theta$.

Let A_θ be the rotation matrix. To find eigenvectors consider:

$$A_\theta x = e^{i\theta}x.$$

Set the second coordinate of x to be 1 hence the first equation from the above reads:

$$(\cos\theta)x_1 - \sin\theta = (\cos\theta)x_1 + i(\sin\theta)x_1.$$

Hence $x_1 = -1/i = i/(-ii) = i$. Thus an eigenvector corresponding to $e^{i\theta}$ is $x = [i, 1]^T$ and thus a normalized one is $(1/\sqrt{2})[i, 1]^T$.

Similarly, a noramlized eigenvector corresponding to $e^{-i\theta}$ is $(1/\sqrt{2})[1, i]^T$.

Here is a test...

In [10]:
```
A(θ) = [cos(θ) -sin(θ); sin(θ) cos(θ)]
θ = pi/6
eigvals(A(θ))
```

```
cannot define function A; it already has a value

Stacktrace:
 [1] top-level scope at In[10]:1
```

In [11]:
```
exp(im*θ),exp(-im*θ)
```

```
UndefVarError: θ not defined

Stacktrace:
 [1] top-level scope at In[11]:1
```

In [12]:
```
x1 = [im,1]/sqrt(2);
A(θ)*x1 - exp(im*θ)*x1
```

```
UndefVarError: θ not defined

Stacktrace:
 [1] top-level scope at In[12]:2
```

In [13]:
```
x2 = [1,im]/sqrt(2);
A(θ)*x2 - exp(-im*θ)*x2
```

```
UndefVarError: θ not defined

Stacktrace:
 [1] top-level scope at In[13]:2
```

(b) Trace $= 2\cos\theta$. Sum of eigenvalues $= \cos\theta + i\sin\theta + \cos\theta - i\sin\theta = 2\cos\theta$.

(c) Det = $\cos^2\theta + \sin^2\theta = 1$. Product of eigenvalues = $e^{i\theta}e^{-i\theta} = e^0 = 1$.

Solution to Question 7

Looking at AB and BA we have,

$$AB = \left(X\Lambda_1X^{-1}\right)\left(X\Lambda_2X^{-1}\right) = X\Lambda_1\Lambda_2X^{-1}.$$

$$BA = \left(X\Lambda_2X^{-1}\right)\left(X\Lambda_1X^{-1}\right) = X\Lambda_2\Lambda_1X^{-1}.$$

Hower $\Lambda_1\Lambda_2 = \Lambda_2\Lambda_1$ because these matrices are diagonal. Hence $AB = BA$.

Solution to Question 8

(a) The rank of A is 1 with

$$A = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix}.$$

Now look at

$$W = A^T A = \begin{bmatrix} 5 & 10 \\ 10 & 20 \end{bmatrix}.$$

The characteristic polynomial of W is $(5 - \lambda)(20 - \lambda) - 100 = \lambda^2 - 25\lambda = \lambda(\lambda - 25)$.

Hence eigenvalues are $\lambda = 0$ and $\lambda = 25$. Hence the singular value is $\sigma_1 = \sqrt{25} = 5$.

We can now guess that the SVD has to be of the form,

$$A = U \times 5 \times V^T$$

and hence given the structure of A we can set $U = [2/\sqrt{5} \quad 1/\sqrt{5}]^T$ and V = $U = [1/\sqrt{5} \quad 2/\sqrt{5}]^T$ which happen to be normed vectors.

Here is a sanity check with Julia:

In [14]:
```
using LinearAlgebra
A = [2 4; 1 2]
F = svd(A)
println("Singular value: ", F.S[1])
#it turns out that svd() in Julia chooses the negative of it
println("U:",F.U[:,1],"  or   ", [2/sqrt(5),1/sqrt(5)])
println("V:",F.U[:,1],"  or   ", [1/sqrt(5),2/sqrt(5)])
```

```
Singular value: 5.000000000000001
U:[-0.894427, -0.447214]  or   [0.894427, 0.447214]
V:[-0.894427, -0.447214]  or   [0.447214, 0.894427]
```

(b) To explore, let's take a different approach for the rank 1 matrix B. We know the sum of the eigenvalues of $B^T B$ is its trace. The $(1, 1)$ element of $B^T B$ is $2 \times 2 + 8 \times 8 = 68$. The $(2, 2)$ element is $(-1) \times (-1) + (-4) \times (-4) = 17$. Hence the trace is $68 + 17 = 85$. Now since the matrix $B^T B$ is of rank 1 one of the eigenvalues is 0 and the other must be 85. Hence the singular value is $\sqrt{85} \approx 9.21954$.

Here is a sanity check:

In [15]:
```
B = [2 -1; 8 -4];
svdvals(B)
```

Out[15]:
```
2-element Array{Float64,1}:
 9.219544457292887
 7.944109290391273e-16
```

Now we compute matching normalized eigenvectors for $B^T B$ to get

$$V = \frac{1}{\sqrt{5}} \begin{bmatrix} -2 & 1 \\ 1 & 2 \end{bmatrix}.$$

Now look at BB^T and get matching eigenvectors:

$$U = \frac{1}{\sqrt{17}} \begin{bmatrix} -1 & -4 \\ -4 & 1 \end{bmatrix}.$$

In [16]:
```
B
```

Out[16]:
```
2×2 Array{Int64,2}:
 2  -1
 8  -4
```

In [17]:
```
V = [-2 1 ;1 2]/sqrt(5);
U = [-1 -4; -4 1]/sqrt(17);
Σ = [sqrt(85) 0 ; 0 0];

U*Σ*V'
```

Out[17]:
```
2×2 Array{Float64,2}:
 2.0  -1.0
 8.0  -4.0
```

(c) The explicit computation of SVD for $A + B$ is messy.

In [18]:
```
A+B
```

Out[18]:
```
2×2 Array{Int64,2}:
 4   3
 9  -2
```

In [19]:
```
F = svd(A+B)
F.U
```

Out[19]:
```
2×2 Array{Float64,2}:
 -0.382683  -0.92388
 -0.92388    0.382683
```

In [20]:
```
Diagonal(F.S)
```

Out[20]:
```
2×2 Diagonal{Float64,Array{Float64,1}}:
 9.87048   ⋅
  ⋅       3.54593
```

In [21]:
```
F.V
```

Out[21]:
```
2×2 Adjoint{Float64,Array{Float64,2}}:
 -0.997484  -0.070889
  0.070889  -0.997484
```

In [22]:
```
F.U*Diagonal(F.S)*F.V'
```

Out[22]:
```
2×2 Array{Float64,2}:
 4.0   3.0
 9.0  -2.0
```

Solution to Question 9

(a) Since A is non-singular we have that $\Sigma = AA^T$ is non-singular. Hence Σ^{-1} exists and $|\Sigma| \neq 0$.

(b) The derivation is based on the explicit inverse of Σ (sometimes called the precision matrix). Note that $|\Sigma| = \sigma_1^2\sigma_2^2(1-\rho)$ and, $$

\Sigma^{-1}

\frac{1}{\sigma_1^2\sigma_2^2(1-\rho)} \left[

$$\begin{matrix} \sigma_2^2 & -\sigma_1\sigma_2\rho \\ -\sigma_1\sigma_2\rho & \sigma_1^2 \end{matrix}$$

\right]. $$

After some manipulation the standard expression can be obtained:

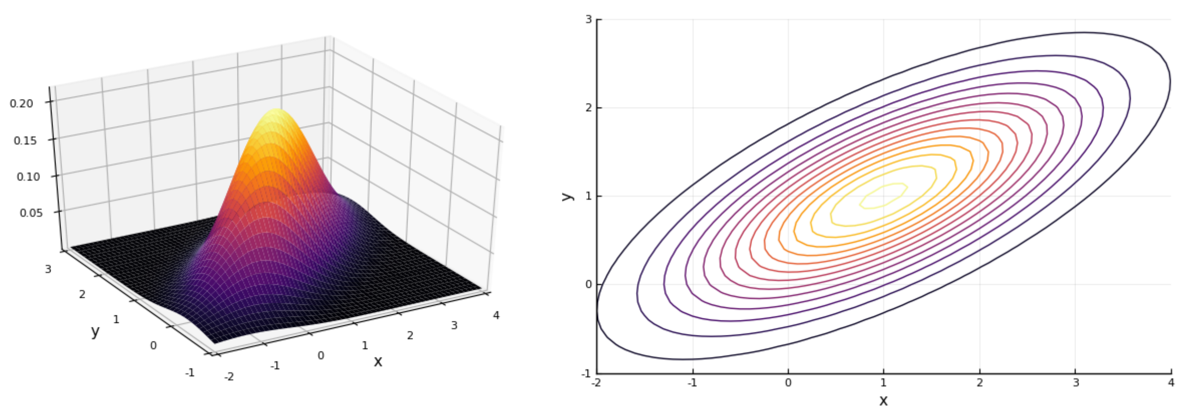
$$f(x,y) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \times \exp\left\{ \frac{-1}{2\left(1-\rho^2\right)} \left[\frac{(x-\mu_1)^2}{\sigma_1^2} - \frac{2\rho\,(x-\mu_1)\,(y-\mu_2)}{\sigma_1\sigma_2} + \frac{(y-\mu_2)^2}{\sigma_2^2} \right] \right\}$$

(c) Here are plots:

In [23]:
```
using Plots, LinearAlgebra
pyplot()
μ1, μ2 = 1, 1
σ1, σ2 = 1.3, 0.8
ϱ = 0.7


#direct implementation
fa(x) = (2π*σ1*σ2*sqrt(1-ϱ^2))^-1 *
    exp(-(2*(1-ϱ^2))^-1 * ((x[1]-μ1)^2/σ1^2 - 2ϱ*(x[1]-μ1)*(x[2]-μ2)/(σ1*
σ2) + (x[2]-μ2)^2/σ2^2 ))

μ = [μ1,μ2]
Σ = [σ1^2 ϱ*σ1*σ2;
     ϱ*σ1*σ2 σ2^2]

fb(x) = (2π)^-1 * det(Σ)^-0.5 * exp(-0.5*(x-μ)'*inv(Σ)*(x-μ))

println("Sanity check that both functions fa() and fb() are the same:")
println(fa([0,0]),"\t",fb([0,0]))
println(fa([1,0]),"\t",fb([1,0]))


xGrid = -2:0.1:4
yGrid = -1:0.1:3
p1 = surface(xGrid,yGrid,(x1,x2)->fa([x1,x2]),
            legend=false,xlabel="x", ylabel="y",camera=(-30,35),size=(50
0,400))
p2 = contour(xGrid,yGrid,(x1,x2)->fa([x1,x2]),
            legend=false,xlabel="x", ylabel="y",size=(500,400))
plot(p1,p2,size=(1200,400))
```

```
Sanity check that both functions fa() and fb() are the same:
0.09703890701860532     0.0970389070186053
0.04631505318110716     0.04631505318110719
```

Out[23]:



(d) Calculating/estimating $P(X < 0, Y < 0)$:

In [24]:
```
#Using a crude Riemann sum:
δ = 0.001
M = 5 #approximates infinity
grid = -M:δ:0
sum([fa([x,y])*δ^2 for x in grid, y in grid ])
```

Out[24]: 0.07555712725292603

In [25]:
```
#Using Monte-Carlo:
using Distributions
N = 10^7
length(filter((x)->(x[1]<0 && x[2]<0), [rand(MvNormal(μ,Σ)) for _ in 1:N
]))/N
```

Out[25]: 0.0754657

Solution to Question 10

The code below is a modification of the code in lecture 1 (also appearing in the [SWJ] book). Note the use of the modulo (%) operator for obtaining the parity of an integer (0 for even and 1 for odd).

In [26]:

```
using Flux.Data.MNIST, LinearAlgebra
using Flux: onehotbatch

imgs = MNIST.images()
labels = MNIST.labels()

nTrain = length(imgs)
trainData = vcat([hcat(float.(imgs[i])...) for i in 1:nTrain]...)
trainLabels = labels[1:nTrain]

testImgs = MNIST.images(:test)
testLabels = MNIST.labels(:test)
testParity = testLabels .% 2      #has 0 for even and 1 for odd

nTest = length(testImgs)
testData = vcat([hcat(float.(testImgs[i])...) for i in 1:nTest]...)

A = [ones(nTrain) trainData]
Adag = pinv(A)
tfPM(x) = x ? +1 : -1
yDatExplicit(k) = tfPM.(onehotbatch(trainLabels,0:9)'[:,k+1])
bets = [Adag*yDatExplicit(k) for k in 0:9]
classifyExplicitDigit(input) = findmax([([1 ; input])'*bets[k] for k in 
1:10])[2]-1

#### This is possibility I
classifyParityI(input) = classifyExplicitDigit(input) % 2
predictions = [classifyParityI(testData[k,:]) for k in 1:nTest]
accuracyI = sum(predictions .== testParity)/nTest
println("Accuracy with method I:", accuracyI)

#### This is possibility II
yDatParity = tfPM.((trainLabels .% 2) .== 1 )
beta = Adag*yDatParity
classifyParityII(input) = [1 ; input]'*beta > 0 ? 1 : 0
predictions = [classifyParityII(testData[k,:]) for k in 1:nTest]
accuracyII = sum(predictions .== testParity)/nTest
println("Accuracy with method II:", accuracyII)
```

```
Accuracy with method I:0.9283
Accuracy with method II:0.894
```

As can be seen, method I obtains 92.83% accuracy while method II (directly training on images labeled as "odd" or "even" obtains 89.4% accuracy. Hence it appears that method I is superior.