ECE 133A HW 1

Lawrence Liu

October 5, 2022

Exercise T2.4

ϕ is not linear, since point 2 is equal to the negative of point 3, but the output of the function at point 2 is not the negative of the output at point 3.

Exercise T2.8

(a)

$$\int p(x)dx = \sum_{i=1}^{n} c_i \frac{1}{i} x^i$$
$$\int_{\alpha}^{\beta} = \sum_{i=1}^{n} c_i \frac{1}{i} (\beta^i - \alpha^i)$$

therefore we have

$$a = \boxed{\left((\beta - \alpha)), ..., \frac{1}{n}(\beta^n - \alpha^n)\right)}$$

(b)

we have

$$p'(\alpha) = \sum_{i=1}^{n} (i-1) c_i \alpha^{i-2}$$

thus we get

$$b = \boxed{\left(0, 1, ..., (n-1)\alpha^{n-2}\right)}$$

Exercise A1.2

let $u = (\sqrt{x_1}, ..., \sqrt{x_n})$ and $v = \left(\sqrt{\frac{1}{x_1}}, ..., \sqrt{\frac{1}{x_n}}\right)$, from Cauchy-Schwarz we have

$$\begin{aligned}
\|u^T v\| &\leq \|u\|\|v\| \\
\|u^T v\|^2 &\leq \|u\|^2\|v\|^2 \\
n^2 &\leq \left(\sum_{k=1}^{n} x_n\right)\left(\sum_{k=1}^{n} \frac{1}{x_k}\right) \\
\frac{1}{n}\sum_{k=1}^{n} x_k &\geq n \left(\sum_{k=1}^{n} \frac{1}{x_k}\right)^{-1} \\
\frac{1}{n}\sum_{k=1}^{n} x_k &\geq \left(\frac{1}{n}\sum_{k=1}^{n} \frac{1}{x_k}\right)^{-1}
\end{aligned}$$

Exercise T3.25

(a)

$$
\begin{aligned}
E[p] &= \theta \text{avg}(r) + (1-\theta)\text{avg}(\mu^{\text{rf}}\mathbf{1}) \\
E[p] &= \boxed{\theta\mu + (1-\theta)\mu^{\text{rf}}}
\end{aligned}
$$

$$
\begin{aligned}
\text{std}(p) &= \sqrt{\frac{(p_1 - E[p])^2 + ... + (p_T - E[p])^2}{T}} \\
\text{std}(p) &= \sqrt{\theta^2(\text{std}(r))} \\
\sqrt{Var(p)} &= \boxed{|\theta|\sigma}
\end{aligned}
$$

(b)

Therefore we have that

$$\theta_{optimal} = \pm\frac{\sigma^{\text{tar}}}{\sigma}$$

Therefore if our risky asset has a rate of return greater than the risk free asset ie: $\mu > \mu^{rf}$, we will pick

$$\theta_{optimal} = \boxed{\frac{\sigma^{\text{tar}}}{\sigma}}$$

And if our risky asset has a rate of return less than the risk free asset, ie: $\mu < \mu^{rf}$ we will choose

$$\theta_{optimal} = \boxed{-\frac{\sigma^{\text{tar}}}{\sigma}}$$

(c)

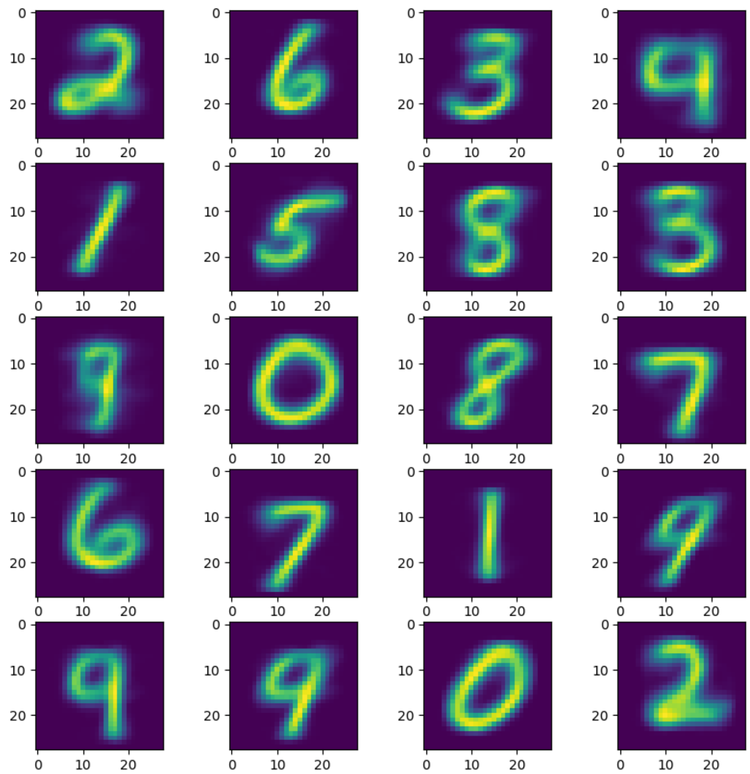
If we want our portfolio to have a lower risk level than the asset, ie $\sigma^{\text{tar}} < \sigma$, but if the expected return rate of the asset is still positive, then we will hedge, since $\theta_{optimal} = \frac{\sigma^{\text{tar}}}{\sigma} < 1$.

If we are comfortable with a greater level of risk than the asset, ie $\sigma^{\text{tar}} > \sigma$, and if the expected return rate of the asset is still positive, we will Leverage since $\theta_{optimal} = \frac{\sigma^{\text{tar}}}{\sigma} > 1$.

And, if the expected return rate of the asset is negative, we will short the asset.

Exercise A1.10

After 29 iterations, I get a $J^{\text{cluster}} = 34.63$ and clusters which look like this



This was acomplished with the following code in Julia

```
using MAT
#using Plots
using Random
using LinearAlgebra
using Statistics
using PyPlot
#using Distributions
#using Printf


function random_assign(data,n_clusters)
    n = size(data,2)
    assignments = rand(1:n_clusters,n)
    return assignments
end
```

```
function update_centroids(data,assignments,n_clusters)
    n = size(data,2)
    d = size(data,1)
    centroids = zeros(n_clusters,d)
    #println(size(centroids))
    for i = 1:n_clusters
        centroids[i,:] = transpose(mean(data[:,assignments .== i],dims=2))
    end
    # print(centroids[10,:],"\n")
    return centroids
end

function assign(data,centroids)
    n = size(data,2)
    #println("n=",n)
    n_clusters = size(centroids,1)
    #make an array
    assignments = zeros(Int64,n)
    for i = 1:n
            #println(size(data[:,i]))
            #println(size(centroids))
            #println(size(transpose(centroids).-data[:,i]))
            dists = mean((transpose(centroids).-data[:,i]).^2,dims=1)
            #println(size(dists))
            assignments[i]=argmin(dists)[2]
    end
    # println(unique!(assignments))
    #println(size(assignments))
    return assignments
end

function calculate_J(data,assignments,centroids)
    n = size(data,2)
    J = 0
    # println(size(assignments))
    for i = 1:n
        J += sum((data[:,i] .- centroids[assignments[i],:]).^2)
    end
    J=J/n
    return J
end

function main()
    file = matopen("mnist_train.mat")
    digits = read(file, "digits")[:,1:10000]
    #print(size(digits))
    close(file)
    println("loaded digits, running K-means")
    d = size(digits,1)
    n_clusters = 20
    assignments = random_assign(digits,n_clusters)
    centroids = zeros(n_clusters,d)
    J_change_threshold = 10^-5
    #print(J_change_threshold)
    J_old=Inf
    Js=Vector{Float64}()
    i=0
    while true
        #print("updating centroids\n")
        centroids = update_centroids(digits,assignments,n_clusters)
        #print("assigning\n")
        assignments = assign(digits,centroids)
        #print("calculating J\n")
        J=calculate_J(digits,assignments,centroids)
        #print("J=",J,"\n")
        #print("---------------------------\n")
        append!(Js,J)
        if abs(J-J_old)<=J*J_change_threshold
            break
        end
        J_old=J
        i+=1
    end
    println("achived a J of ", round(Js[length(Js)],digits=2), " after ", i, " iterations")
    #plot out Js
    plot(Js)
```

```
    savefig("Js.png")

    #plot out an image
    #println(size(reshape(centroids[1,:,:],(28,28))))
    fig=figure("testfig",figsize=(10,10))
    for i=1:n_clusters
            subplot(5,4,i)
            imshow(reshape(centroids[i,:,:],(28,28)))
    end
    savefig("digits.png")
    close(fig)
end

main()
```