

# ECE 133A HW 4

Lawrence Liu

November 10, 2022

## Exercise T13.3

(a)

With the following julia code we get:

```
using MAT
using LinearAlgebra
using PyPlot
# using Statistics

include("mooreslaw.m")
# println(T)
Years, Transistors=T[:,1],T[:,2]
# println(Years)
# println(Transistors)
A=transpose([reshape(ones(size(Years)),1,:); reshape(Years.-1970,1,:)])
# println(A)
log_Transistors=log10.(Transistors)
theta=A\log_Transistors
println("theta_1=",theta[1])
println("theta_2=",theta[2])
#plot out
plot(Years,log_Transistors,"o")
plot(Years,A*theta)
xlabel("Years")
ylabel("Transistors (log10)")
title("Moore's Law")
legend(["Data","Fit"])
savefig("Moore's Law.png")
close()
```

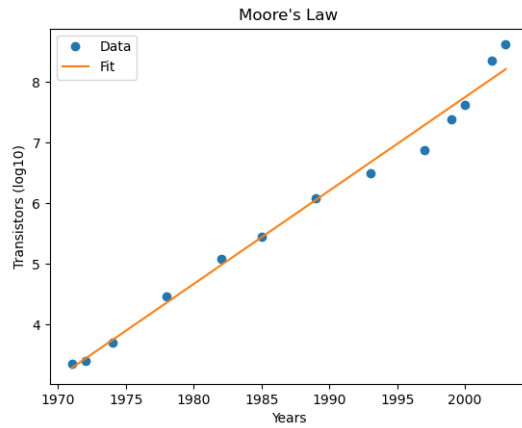
that

$$\theta_1 = 3.125592633829346$$

and

$$\theta_2 = 0.1540181798438225$$

which results in the following fit:



(b)

From our fit we expect the number of transistors to be:

$$10^{\theta_1 + \theta_2(2015-1970)} \approx 10^{10}$$

Which is more than the acutally number of  $4 \cdot 10^9$  transistors:

(c)

This is in line with Moore's law since  $2\theta_2 = 0.30803635968$  which is close to  $\log_{10}(2) = 0.30102999566$

## Exercise T12.12

(a)

Let  $p_{ik} = [u_{ik}, v_{ik}]$  then we have that

$$\begin{aligned} \|p_{i_1} - p_{j_1}\|^2 + \dots + \|p_{i_L} - p_{j_L}\|^2 &= (u_{i_1} - u_{j_1})^2 + \dots + (u_{i_L} - u_{j_L})^2 + (v_{i_1} - v_{j_1})^2 + \dots + (v_{i_L} - v_{j_L})^2 \\ \|p_{i_1} - p_{j_1}\|^2 + \dots + \|p_{i_L} - p_{j_L}\|^2 &= D(u) + D(v) \end{aligned}$$

(b)

Let  $C$  be the incident matrix of the graph we have that we want to minimize

$$\|C^T u\|^2 + \|C^T v\|^2$$

Letting the matrix made up of the first  $N - K$  rows of  $C$  be the matrix  $A$  and the last  $K$  rows of  $C$ , likewise let  $u_1$  and  $v_1$  be the first  $N - K$  rows of  $u$  and  $v$  and  $u_2$  and  $v_2$  be the last  $K$  rows of  $u$  and  $v$  respectively, then we have

$$\begin{aligned} \|B^T u\|^2 + \|B^T v\|^2 &= \|A^T u_1 + B^T u_2\|^2 + \|A^T v_1 + B^T v_2\|^2 \\ &= \left\| \begin{bmatrix} A^T & 0 \\ 0 & A^T \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \end{bmatrix} + \begin{bmatrix} B^T u_2 \\ B^T v_2 \end{bmatrix} \right\|^2 \end{aligned}$$

So then we have that in term of a least square problem we have  $\|Ax - b\|^2$

$$A = \begin{bmatrix} A^T & 0 \\ 0 & A^T \end{bmatrix}$$

And

$$b = \begin{bmatrix} B^T u_2 \\ B^T v_2 \end{bmatrix}$$

And

$$x = \begin{bmatrix} u_1 \\ v_1 \end{bmatrix}$$

(c)

With the following code in julia

```
using PyPlot

A=[[1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 1. 1. 1. 0. 0. 0. 0. 0.]
 [1. 0. 0. 1. 0. 0. 0. 1. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 1. 0. 1. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1.]]
B=[[0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]

println(A)
println(B)

A=vcat(hcat(transpose(A), zeros(size(A)[2], size(A)[1])),
 hcat(zeros(size(A)[2], size(A)[1]), transpose(A)))

println(size(A))

u2=transpose([[0] [0] [1] [1]])
v2=transpose([[0] [1] [1] [0]])
println(size(u2))
b=vcat(transpose(B)*u2, transpose(B)*v2)
println(size(transpose(B)*u2))
println(size(b))

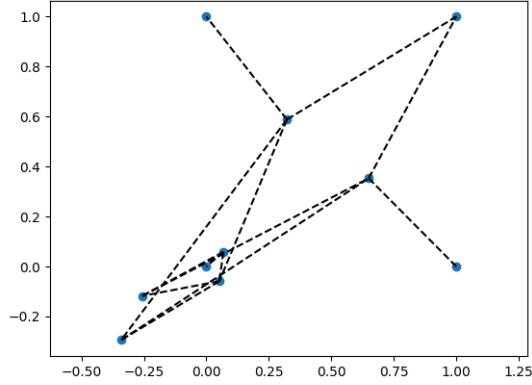
x=A\b
println(x)
u=reshape(vcat(x[1:6], u2), (10))
v=reshape(vcat(x[7:12], v2), (10))

plot(u, v, "o")
axis("equal")
edges=[[ 1 3]
 [ 1 4]
 [ 1 7]
 [ 2 3]
 [ 2 5]
 [ 2 8]
 [ 2 9]
 [ 3 4]
 [ 3 5]
 [ 4 6]
 [ 5 6]
 [ 6 9]
 [ 6 10]]

for i=1:size(edges)[1]
    plot(u[edges[i, :]], v[edges[i, :]], "--", color="black")
end

savefig("fig1.png")
close()
```

We get the following plot:



### Exercise A8.3

We can get that

$$\alpha t_i + \beta = \ln\left(\frac{y_i}{1 - y_i}\right)$$

So therefore we can have a least squares problem, with

$$A = \begin{bmatrix} t_1 & 1 \\ t_2 & 1 \\ \vdots & \vdots \\ t_n & 1 \end{bmatrix}$$

and

$$b = [\ln(\frac{y_1}{1 - y_1}), \ln(\frac{y_2}{1 - y_2}), \dots, \ln(\frac{y_n}{1 - y_n})]^T$$

and

$$x = [\alpha, \beta]^T$$

Then we have a least squares problem of

$$||Ax - b||^2$$

To find  $\alpha$  and  $\beta$  I used the following code:

```

using PyPlot
include("logistic_fit.jl")

t,y=logistic_fit()

A=ones(length(t),2)
A[:,1]=t
b=log.(y./(ones(length(t)).-y))

x=A\b
# println(x)
# println(x[1].*t)
# println(x[2])
t1=LinRange(-1,5,50)
y1=exp.(x[2].+(x[1].*t1))./(ones(length(t1)).+exp.(x[2].+(x[1].*t1)))
println("alpha=",x[1])
println("beta=",x[2])
plot(t,y,"o",label="data")
plot(t1,y1,label="fit")
xlabel("t")
ylabel("y")
title("Logistic Fit")
legend()
savefig("fig2.png")
close()

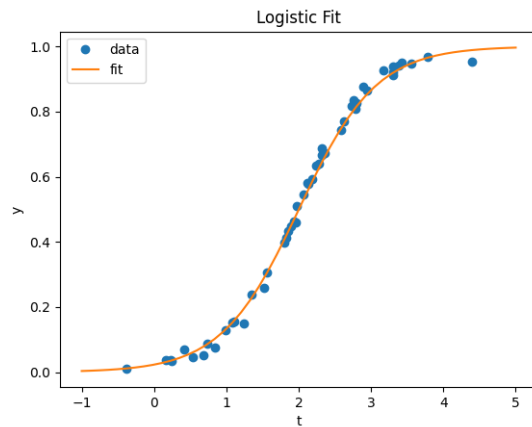
```

which results in

$$\alpha = 1.8676293241597044$$

$$\beta = -3.739673238861126$$

and the following fit:



## Exercise A5.11

(a)

we have that

$$x = A^\dagger b$$

$$x = (A^T A)^{-1} A^T b$$

$$x = \left( \begin{bmatrix} 1 & 10^{-k} & 0 \\ 1 & 0 & 10^{-k} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 10^{-k} & 0 \\ 0 & 10^{-k} \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 10^{-k} & 0 \\ 1 & 0 & 10^{-k} \end{bmatrix} \begin{bmatrix} -10^{-k} \\ 1 + 10^{-k} \\ 1 - 10^{-k} \end{bmatrix}$$

$$x = \left( \begin{bmatrix} 1 + 10^{-2k} & 1 \\ 1 & 1 + 10^{-2k} \end{bmatrix} \right)^{-1} \begin{bmatrix} 10^{-2k} \\ -10^{-2k} \end{bmatrix}$$

$$x = \begin{bmatrix} \frac{1+10^{-2k}}{10^{-4k}+2 \cdot 10^{-2k}} & -\frac{1}{10^{-4k}+2 \cdot 10^{-2k}} \\ -\frac{1}{10^{-4k}+2 \cdot 10^{-2k}} & \frac{1+10^{-2k}}{10^{-4k}+2 \cdot 10^{-2k}} \end{bmatrix} \begin{bmatrix} 10^{-2k} \\ -10^{-2k} \end{bmatrix}$$

$$x = \frac{10^{-2k}}{10^{-4k} + 2 \cdot 10^{-2k}} \begin{bmatrix} 2 + 10^{-2k} \\ -2 - 10^{-2k} \end{bmatrix}$$

thus we have for  $k = 6$ :

$$x = \frac{10^{-12}}{10^{-24} + 2 \cdot 10^{-12}} \begin{bmatrix} 2 + 10^{-12} \\ -2 - 10^{-12} \end{bmatrix}$$

And for  $k = 7$  we have

$$x = \frac{10^{-14}}{10^{-28} + 2 \cdot 10^{-14}} \begin{bmatrix} 2 + 10^{-14} \\ -2 - 10^{-14} \end{bmatrix}$$

And for  $k = 8$  we have

$$x = \frac{10^{-16}}{10^{-32} + 2 \cdot 10^{-16}} \begin{bmatrix} 2 + 10^{-16} \\ -2 - 10^{-16} \end{bmatrix}$$

(b)

We have that with the following code:

```

for k = [6,7,8]

    A=transpose([1,1] [10.0^-k,0] [0,10.0^-k]))
    b=[-(10.0^-k),1+10.0^-k, 1-10.0^-k]
    println("k=",k)
    println("x=",A\b)
    println("_____")
end

```

we get that

```

k=6
x=[0.9999999999176988, -0.999999999917699]
-----
k=7
x=[1.00000000027436178, -1.00000000027436176]
-----
k=8
x=[1.00000000043137076, -1.00000000043137076]
-----

```

(c)

We have that with the following code:

```

for k = [6,7,8]

    A=transpose([1,1] [10.0^-k,0] [0,10.0^-k]))
    b=[-(10.0^-k),1+10.0^-k, 1-10.0^-k]
    println("k=",k)
    println("x=", (A'*A) \ (A'*b))
    println("_____")
end

```

we get that



```

k=7
x=[1.00000000027436178, -1.00000000027436176]
-----
k=8
x=[1.00000000043137076, -1.00000000043137076]
-----

```

And for  $k = 8$  we get a singular exception error since  $\mathbf{A}' * \mathbf{A}$  gets rounded to a matrix  $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$  that is singular

## Exercise A5.12

(a)

$$f(y) = \|Ay - b\|^2 + (c^T y - d)^2$$

To minimize we take the derivative of it with respect to  $y_i$  for all  $1 \leq n \leq N$  and set it to zero have

$$\frac{\partial}{\partial y_i} f(y) = 2(A^T(Ay - b))_i + 2(c^T y - d)c_i = 0$$

Thus we have

$$\nabla f(y) = 2(A^T(Ay - b) + c(c^T y - d)) = 0$$

which gives us

$$\begin{aligned} \nabla f(y) &= 0 \\ 2(A^T(Ay - b) + c(c^T y - d)) &= 0 \\ A^T(Ay - b) + c(c^T y - d) &= 0 \end{aligned}$$

if  $\hat{y}$  is a solution then we must have that

$$A^T(A\hat{y} - b) + c(c^T \hat{y} - d) = 0$$

we can confirm this, since

$$\hat{y} = \hat{x} + \frac{d - c^T \hat{x}}{1 + c^T (A^T A)^{-1} c} (A^T A)^{-1} c$$

we have:

$$\begin{aligned} A^T (A \hat{y} - b) + c(c^T \hat{y} - d) &= 0 \\ A^T A \frac{d - c^T \hat{x}}{1 + c^T (A^T A)^{-1} c} (A^T A)^{-1} c + c c^T \hat{x} + c(c^T \frac{d - c^T \hat{x}}{1 + c^T (A^T A)^{-1} c} (A^T A)^{-1} c - d) &= 0 \\ d c - c^T \hat{x} c + c c^T (d - c^T \hat{x}) (A^T A)^{-1} c - c(d - c^T \hat{x})(1 + c^T (A^T A)^{-1} c) &= 0 \\ c c^T (d - c^T \hat{x}) (A^T A)^{-1} c - c(d - c^T \hat{x})(c^T (A^T A)^{-1} c) &= 0 \\ c c^T (d - c^T \hat{x}) (A^T A)^{-1} c - c c^T (d - c^T \hat{x}) (A^T A)^{-1} c &= 0 \end{aligned}$$

(b)

We first compute the QR factorization of  $A$ , which will cost us  $2mn^2$  flops, then we can compute  $\hat{x}$  with an additional  $2mn + n^2$  flops. Likewise, since we can rewrite  $(A^T A)^{-1} c$  as  $(R^T Q^T Q R)^{-1} c = (R^T R)^{-1} c$ , which we can solve in  $2n^2$  flops. then computing  $c^T \hat{x}$  and  $c^T (A^T A)^{-1} c$  will each cost us an additional  $2n - 1$  flops, then computing  $\frac{d - c^T \hat{x}}{1 + c^T (A^T A)^{-1} c}$  will cost us 3 flops. Then computing  $\hat{x} + \frac{d - c^T \hat{x}}{1 + c^T (A^T A)^{-1} c} (A^T A)^{-1} c$  will cost us  $2n$  flops, so in total this algorithm will cost us  $\boxed{2mn^2 + 2mn + 3n^2 + 6n - 1}$  flops.