## GD - OLS

Verdier for eta:

array([1.05131232e-04, 5.25656160e-04, 1.05131232e-03, 5.25656160e-03,

1.05131232e-02, 5.25656160e-02, 1.05131232e-01, 2.10262464e-01])

Niterations = 5000

conv=10\*\*(-4) #konvergenskriterium holder til dette formålet

#eta = 0.5/np.max(EigValues) #learning rate, denne må være mindre enn 2/(den største egenverdien til H) for å kunne konvergere

eta = [1.e-03, 5.e-03, 1.e-02, 5.e-02, 1.e-01, 5.e-01, 1.e+00, 2.e+00]

A graph with a blue line and a blue line

Description automatically generated

A graph with a line

Description automatically generated

A graph with numbers and a line

Description automatically generated

Endelig

Eta = 0.1.

Niterations = 3000

conv=10\*\*(-4) #konvergenskriterium holder til dette formålet

beta linreg

[[10.56140272]

[ 1.63887939]

[ 1.5810637 ]]

gradient

[[-3.00094887e-05]

[ 8.57383642e-05]

[-4.13799701e-05]]

beta GD

[[10.56046054]

[ 1.64157124]

[ 1.57976453]]

MSE\_GD

18.656638127036356

MSE\_OLS

18.656688485492715

R2\_GD

-0.8782605540902844

R2\_OLS

-0.8782656239373545

A red and blue dotted line

Description automatically generated

A graph of a function

Description automatically generated with medium confidence

## GDM-OLS

Niterations = 10000

conv=10\*\*(-4) #konvergenskriterium holder til dette formålet

#eta = 0.5/np.max(EigValues) #learning rate, denne må være mindre enn 2/(den største egenverdien til H) for å kunne konvergere

eta = [1.e-02,5.e-02, 1.e-01, 5.e-01, 1.e+00]

eta = eta / np.max(EigValues) #learning rate < 2/(den største egenverdien til H)

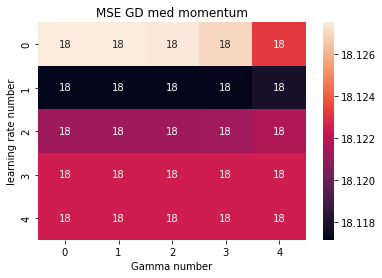
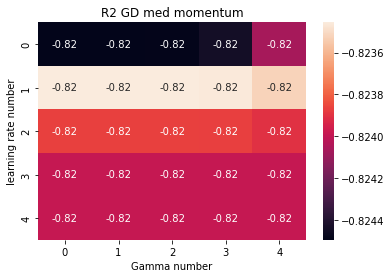
gamma = np.logspace(-5,-1,5) #parameter til GDM

gamma

Out[171]: array([1.e-05, 1.e-04, 1.e-03, 1.e-02, 1.e-01])

eta

Out[172]: array([0.00105131, 0.00525656, 0.01051312, 0.05256562, 0.10513123])

### Beste verdier

Beste verdier: Nest minste eta og nest støste gamma

Eta= 0.005

Gamma = 0.01

Da konvergerer det ikke på 10000 iterasjoner engang:

A red and blue dotted line

Description automatically generatedA graph of a number of numbers

Description automatically generated with medium confidence

beta linreg

[[10.56140272]

[ 1.63887939]

[ 1.5810637 ]]

gradient

[[-0.00340043]

[ 0.00971517]

[-0.00468884]]

beta GD

[[10.45464254]

[ 1.94389769]

[ 1.43385249]]

MSE\_GD

18.653086769444396

MSE\_OLS

18.656688485492715

R2\_GD

-0.8779030204964491

R2\_OLS

-0.8782656239373545

### Mens med

eta = 0.1 #

gamma = 0.1 #parameter til GDM

får vi konvergens ettet 2000 iterasjoner og svaret blir like bra!

beta linreg

[[10.56140272]

[ 1.63887939]

[ 1.5810637 ]]

gradient

[[-3.00507457e-05]

[ 8.58562371e-05]

[-4.14368592e-05]]

beta GD

[[10.56045924]

[ 1.64157494]

[ 1.57976275]]

MSE\_GD

18.656638058031344

MSE\_OLS

18.656688485492715

R2\_GD

-0.8782605471431919

R2\_OLS

-0.8782656239373545

## SGD

### Konstatnt eta = 0.1

*Ender opp med å vaka rundt løsningen til OLS*

m = 10 # andtall minibatcher i SGD

10 punkter i hver batch

[[10.56140272]

[ 1.63887939]

[ 1.5810637 ]]

beta GD

[[ Vi]]

MSE\_GD

18.60586801952863

MSE\_OLS

18.12252565626647

R2\_GD

-0.8726440672537072

R2\_OLS

-0.8239966078572818

A red and blue dotted line

Description automatically generatedA graph of a number of numbers and a line

Description automatically generated with medium confidence

### Med tuneable learning rate

m = 10 # andtall minibatcher i SGD

10 punkter i hver batch

*Denne konverterer etter ca 3000 steg. Til en bedre verdi en OLS. Her virker det å være en fordel med korte steg!*

[[10.56140272]

[ 1.63887939]

[ 1.5810637 ]]

beta GD

[[10.27225345]

[ 2.45116256]

[ 1.18927818]]

MSE\_GD

18.12037889468584

MSE\_OLS

18.12252565626647

R2\_GD

-0.8237805405212373

R2\_OLS

-0.8239966078572818

A red and blue dotted line

Description automatically generatedA graph of a function

Description automatically generated with medium confidence

## Med 5 minibatcher isteden:

20 punkter i hver batch

Eta = 0.1.

MSE\_GD

17.825350959466952

*Reproduserer ca ols.*

Med tunable learning rate bruker den lengere tdid på å konvergere i stad, og jgør det mindre godt.

## SGG med momentum

10 minibatches

Samme tall som i stad:

Niterations = 3000

conv=10\*\*(-4) #konvergenskriterium holder til dette formålet

eta = 0.1 #det var den verdien som fungerte for vandlig GD

gamma = 0.1 #parameter til GDM

beta\_prev = np.copy(beta) #til GDM

Konvergerer dårlig, til ca det samme som OLS.

A red and blue line graph

Description automatically generatedA graph of a number of objects

Description automatically generated with medium confidence

Prøver det samme med tuneable learning rate:

Da konvergerer den faktisk selv, men sakter enn den uten momentum:

A red and blue line graph

Description automatically generatedA graph of a number of objects

Description automatically generated with medium confidence

## SGD med Adagrad

10 minibatcher og etaverdier

array([0.00105131, 0.00525656, 0.01051312, 0.05256562, 0.10513123,

0.21026246])

Ser at den nest minste ser bra ut!

Men den er jo veldig lav da. Konvergerer dette?

A graph with numbers and a line

Description automatically generatedA graph of a graph

Description automatically generated with medium confidence

### Kjører den eta = 0.005

[[10.56140272]

[ 1.63887939]

[ 1.5810637 ]]

5000

0.008660253664661267

beta GD

[[10.37428131]

[ 2.26226068]

[ 1.19469438]]

MSE\_GD

17.574603785538876

MSE\_OLS

18.12252565626647

R2\_GD

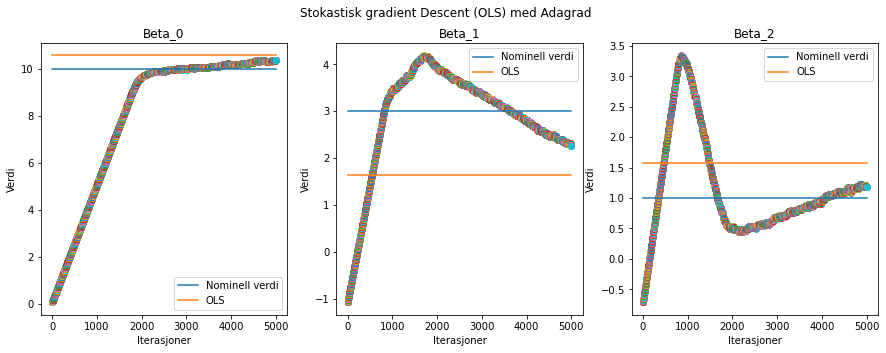
-0.7688493478928646

R2\_OLS

-0.8239966078572818

Men den har ikke konvergert, selv om den ser ut til å være på vei.

A red and blue line graph

Description automatically generated

### Setter learning raten til 0.01 isteden:

Det gikk dårligere.

## ble SGD med RMSprop

Brukte 10 minibatcher

eta

Out[270]: array([0.00525656, 0.01051312, 0.05256562, 0.10513123, 0.21026246])

rho

Out[271]: array([0.0001, 0.001 , 0.01 , 0.1 ])

Ser at midtre Verdi for eta og nest største Verdi for rho peker seg ut: eta = 0.05 og rho = 0.01

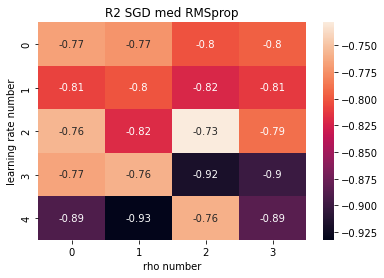
Den ser først ut til å konvergere, så divergerer den faktisk!

A red and blue line graph

Description automatically generatedA graph of a graph

Description automatically generated with medium confidence

A chart of different colors

Description automatically generated

## SGD med Adam:

eta

Out[294]: array([0.00525656, 0.01051312, 0.05256562, 0.10513123, 0.21026246])

rho1

Out[295]: array([0.0001, 0.001 , 0.01 , 0.1 ])

rho2

Out[296]: array([0.0001, 0.001 , 0.01 , 0.1 ])

Ser ut til at det er learning rate nummer 4 som er best og da rho 1 nummer 2 og rho 2 nummer 3

Eta = 0.1

Rho1 = 0.001

Rho2 = 0.01

A screenshot of a chart

Description automatically generated

Det ser ikke ut til å konvergere

A graph with red dots and blue lines

Description automatically generatedA graph of a graph

Description automatically generated with medium confidence

## SGD med tuneable learning rate og skalerte data

Med skalerte data:

Konvergerte etter noen hundre iterasjoner.på omtrent verdien til ols

[[0. ]

[0.32719967]

[0.63434815]]

beta GD

[[0.06428002]

[0.30131474]

[0.66146675]]

MSE\_GD

0.1300066014503099

MSE\_OLS

0.1290610517864677

R2\_GD

0.8699933985496902

R2\_OLS

0.8709389482135323

# Feed forward neural network

## Enkelt nevralt nettverk med stokastisk gradient descent.

100 punkter – 2. ordens polynom med en del støy

10 minbatcher

Stokastic gradient descent med momentum (det får holde!)

Learning rate og gamma = 0.1

Bruker et teststett med 50 punkter.

#setter opp det nevrale nettverket

network\_input\_size = np.shape(x)[1] # på en måte antall noder i inputlaget dvs antal features

layer\_output\_sizes = [3, 1] #må siste lag være 1, siden det jeg sammenligner med er ett tall

activation\_funcs = [sivNN.sigmoid,sivNN.linear\_act]

activation\_ders = [sivNN.sigmoid\_der,sivNN.linear\_act\_der]

Har prøvd med mange og få iterasjoner. Det går ikke så sykt bra.

1000 iterasjoner

MSE ved start = 204.2717

MSE = 50.5004

MSE = 1.0830

MSE = 0.8838

MSE = 0.7737

MSE = 0.7914

MSE = 1.1757

MSE = 0.9301

MSE = 0.8055

MSE = 0.7707

MSE = 0.7666

MSE for testsett = 0.9732

Med 10 000 iterasjoner:

Ingen betre konvergens:

MSE = 0.7767

MSE = 0.7630

MSE = 0.7653

MSE = 0.8260

MSE for testsett = 1.4813

1000 iterasjoner og mindre støy:

MSE ved start = 204.0922

MSE = 50.5043

MSE = 0.0849

MSE = 0.0586

MSE = 0.0359

MSE = 0.0303

MSE = 0.0643

MSE = 0.0241

MSE = 0.0549

MSE = 0.0145

MSE = 0.0154

MSE for testsett = 0.0183

Det virker som det er fordi det er så mye støy i dataene at det ikke går an å gjøre bedre.

### 3 skjulte lag med 3 noder i hver

Gikk skikkelig dårlig

MSE ved start = 100.9680

MSE = 46.8145

MSE = 8.9548

MSE = 8.8315

MSE = 8.8663

MSE = 9.0328

MSE = 8.9477

MSE = 8.7984

MSE = 9.2723

MSE = 9.5975

MSE = 9.8996

MSE for testsett = 9.3785

R^2 for testsett = -0.0508

A graph of red and blue dots

Description automatically generated

### 1 lag 10 noder:

Ikke veldig bra:

MSE ved start = 233.5676

MSE = 10.2963

MSE = 0.9590

MSE = 1.7443

MSE = 0.8406

MSE = 0.7816

MSE = 1.1457

MSE = 0.8031

MSE = 0.8844

MSE = 0.8628

MSE = 0.8406

MSE for testsett = 1.7623

R^2 for testsett = 0.8212

### 2 og 4 lag var heller ikk så verst. 3 var tilsynelatende best.

## Forsøk med andre aktiveringsfunksjoner:

### ReLU 1 skult lag og 3 noder:= ræva

MSE ved start = 240.9441

MSE = 630.3322

MSE = 8.9588

MSE = 8.8292

MSE = 8.7941

MSE = 8.7924

MSE = 8.8027

MSE = 8.7991

MSE = 8.9963

MSE = 8.7990

MSE = 8.8044

MSE for testsett = 10.9849

R^2 for testsett = -0.0515

Ble betre med fleire

Ett lag 10 noder ReLU

MSE = 31.9480

MSE = 1.6118

MSE = 1.5127

MSE = 3.1057

MSE = 2.7542

MSE = 1.4870

MSE = 3.1779

MSE = 1.2185

MSE = 1.5367

MSE = 12.0167

MSE for testsett = 2.6671

R^2 for testsett = 0.7294