**Teori**

I nevrale nettverk brukes det **Aktiverings-funksjoner** til å skalere nodene, dette er forklart i mer detalj i teorien under i nevralt nettverk. Det finnes mange forskjellige aktiverings-funksjoner, med forskjellige egenskaper. Sigmoid funksjonen er illustrert i figur (??), denne har en myk aktivering, men gir kun positive output. To andre som ble brukt i dette eksperimentet er tanh og softsign, illustrert i figur (??) og figur (??). Disse ligner på hverandre, men tanh er brattere enn softsign rundt 0 og gir derfor en bråere overgang. Begge disse funksjonene kan gi negative output. Funksjonene og flere andre, med deres deriverte er listet i [5].

|  |  |  |
| --- | --- | --- |
| Figur 1 viser sigmoid funksjonen. | Figur 2 viser tanh funksjonen. | Figur 3 viser softsign funksjonen. |

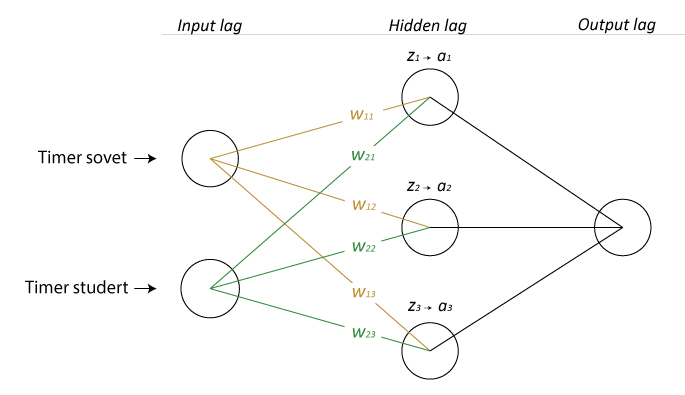
Når det forsøkes å tilpasse en måling eller et resultat ønsker man å minimere **Cost funksjonen**. Denne funksjonen forteller hvor stor feil man har gjort i forhold til målingen. Dette gjøres også i vanlig regresjon. En cost funksjon som brukes i denne oppgaven er f.eks.

(1)

Hvor er målingene som tilpasses i rad og er aktiverings-funksjonen i rad i output laget nummer .

**Nevralt nettverk** fungerer slik at nevroner eller noder sender et signal mellom lag av noder. Det første laget kalles et input lag, illustrert til venstre i figur (??). Dette laget har like mange noder som antall input. Hvis man ser på en modell hvor antall tier man sover er 5, antall timer man studerer er 7 og med dette fikk en person en prosent karakter på 83% [1]. Prøver vi å estimere karakteren gitt antall timer sovet og antall timer 7 er antall noder i input laget 2.

Til høyre i illustrasjonen er output laget. Det er kun 1 node i output laget i vårt tilfelle, siden vi kun vil estimere 83% med metoden. Mellom output og input laget er det ett eller flere hidden lag. Hvert hidden lag kan ha ønsket antall noder, trenger ikke være de samme som noen av de andre lagene, og det er opp til brukeren å definere antall lag. I dette eksempelet er det et hidden lag med 3 noder. Hver node er forbundet med en vekt . Input fra hver node multipliseres med hver sin vekt . Summen av begge multiplikasjonene summeres og kalles aktiviteten, . Aktiviteten sendes deretter gjennom en aktiverings-funksjon som skalerer aktiviteten og symboliseres ved , se videre i teorien.



Figur 4 viser basisen på hvordan et nevralt nettverk fungerer. Alle nodene blir multiplisert med en vekt, , videre til neste lag. Summen av dette, , sendes til en aktiveringsfunksjon , , se tekst for nærmere detaljer.

Det tilsettes også en liten bias verdi i hver node slik at hvis vektene er null gir ikke det kontinuerlig null verdier tilbake. Metoden kan beskrives når alle vekter, aktiveringer og aktiverings-funksjoner beskrives av matriser, matrisene er definert nærmere i [2]. For et hvert trenings input definer den første aktiveringsfunksjonen som . Hvor er laget, i den første aktiveringsfunksjonene er . Først utregnes alle -verdier og deretter -verdier **suksessivt forover**. For , er det output laget, regn ut [3]

for (2)

er en vilkårlig aktiveringsfunksjon. Deretter regnes feilen som gjøres ut i output laget [3]

(3)

Hvor er aktiveringsfunksjonen derivert, er Hadamard produktet [3] og er gradienten til cost-funksjonen i output noden. Dette kan brukes til å gjøre **backpropagation** av feilen man gjør [3]. For hver beregn [3]

(4)

For at det nevrale nettverket skal lære kan man bruke feilen man har funnet til å oppdatere vektene. En måte er å anvende **Gradient decent**. Dette er en metode hvor gradienten til cost funksjonen, , brukes til å bevege seg mot et mindre avvik. Denne metoden beskrives som [3]

og (5)

Når er raden og er kolonnen. Gradientene brukes til å oppdatere vektene hvor er læringsraten[4], som beskriver hvor mye vekten skal bevege seg i retningen til gradienten. De nye vektene er da [3]

(6)

Og de nye biasene er [3]

(7)

I tillegg blir det også ofte brukt **batch gradient decent**. Istedenfor å bruke hele datasettet kalkuleres gradienten i et subset kalt minibatch. Med et data sett med punkter og minibatcher blir antall batcher . Gradienten blir da [3] for

(8)

Siden denne gradienten legger et stokastisk element til, så er sjansen for å bli stående i et lokalt minima mye lavere. Samtidig kan dette øke hastigheten på utregningene.

Det blir også ofte lagt til en **regulerings** **parameter**, til cost funksjonen. Denne parameteren gjør det slik at vektene ikke kan spinne ut av kontroll, akkurat som for ridge regresjon. Denne beskrives som [2]

(9)

**Applikasjon**

**Nevrale nettverk ising modell regresjon**

Input dataen er den samme som i eksperimentet med regresjon. Dataen er en matrise som blir fordelt i et trenings set på 2/3 av størrelsen og test set på 1/3. Treningssettet er da en matrise som brukes til å trene modellen. Til å evaluere modellen brukes et test set som er en matrise. Det nevrale netverket pga. Aktiverings-funksjonene gir ikke output større enn 1 og mindre enn -1. Derfor normaliseres input før det brukes til beregningene.

I dette eksperimentet brukes nevrale nettverk med cost funksjonen i ligning (1), men med en regulerings parameter gitt ved ligning (9). Dette programmeres ved ligningene (2-7) i samme rekkefølge. Det brukes også batch gradient decent gitt ved ligning (8). Softsign-funksjonen blir brukt som aktiveringsfunksjon. Først plottes alle kombinasjoner av og , med en batch størrelse på 50, epoke størrelse på , dvs. Antall ganger man går igjennom det fulle trenings-settet, og med 3 hidden lag med 50, 75 og 30 noder. I denne rekkefølgen. Mean squared error, MSE, anvendes på test settet og plottes.

Det avgjøres så manuelt ut ifra den minimale MSE og fra MSE plottet hvilken 3 kombinasjoner som gir best resultat. Disse tre brukes til å finne MSE til alle kombinasjoner av antall hidden lag, hvor hidden lag størrelsene er 50, 75, 30, 85 og 110. I denne rekkefølgen. Den beste kombinasjonen brukes så til å plotte trenings-sett MSE og test-sett MSE med epoke størrelsen på alt gjort med batch størrelse på 50. Det testes så med de forskjellige aktiveringsfunksjonene nevnt i teoridelen.

Koden ble testet mot regresjons resultatene for å se om det ga et realistisk resultat.

**Resultat**

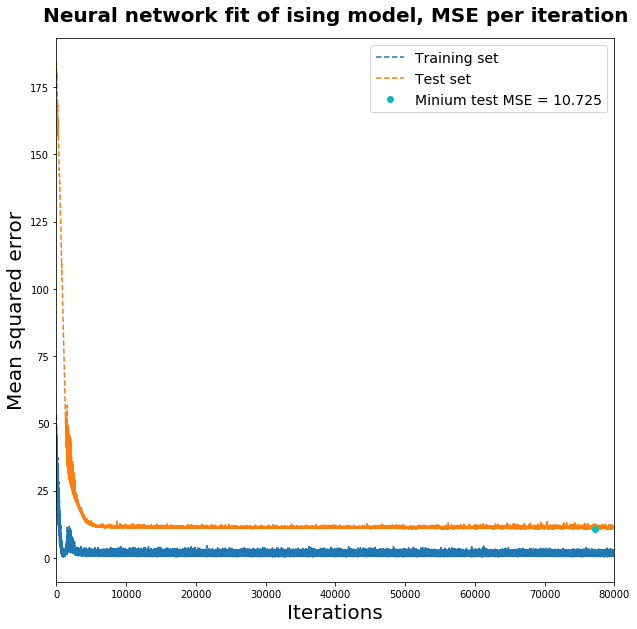
**Nevrale nettverk ising modell regresjon**

I dette eksperimentet ble det brukt de samme input av 1D ising modellen som for regresjons eksperimentet. Det beregnes først MSE for alle mulige kombinasjoner av og . Test MSE blir plottet, hvor epoke størrelsen er , batch størrelse på 50, og hidden lagene er 50, 75, 30 noder. Softsign-funksjonen brukes som aktiveringsfunksjon. Plottet er veldig stort og det må zoomes inn for å få en god oversikt, plottet ligger på github [6] med navn *Eta\_lambda\_testing.png*. Y-aksen venstre er mean squared error av test-settet, x-aksen er antall iterasjoner som er blitt gjort og tegnforklaringen er den minste MSE til test settet av alle iterasjonene.

De beste kombinasjonene blir vurdert til å være rad 3 kolonne 4 , rad 4 kolonne 3 og rad 5 kolonne 2 . Disse hadde de laveste MSE verdiene og var relativt stabile, altså ikke fullstendig støy.

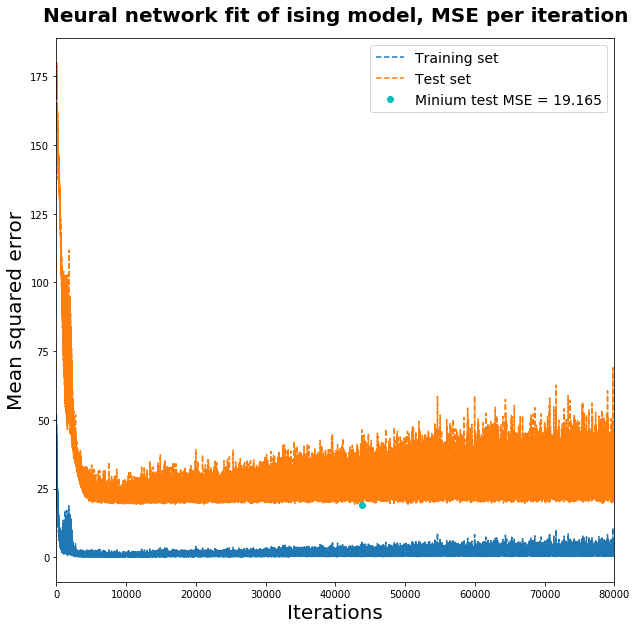
Kombinasjonene blir videre brukt til å beregne MSE som funksjon av antall hidden lag, fra 1-5 lag med antall noder 50, 75, 30, 85 og 110. Dette ble plottet, men dette plottet er også ganske stort og ligger på github [6] med navn *Layer\_size\_testing.png*. I plottet er de forskjellige kombinasjonene plottet på i hver kolonne, hvor det legges til et ekstra hidden lag for hver rad. Y-aksen venstre er mean squared error av test-settet, x-aksen er antall iterasjoner som er blitt gjort og tegnforklaringen er den minste MSE til test settet av alle iterasjonene. Den beste kombinasjonen blir vurdert til å være rad 4 kolonne 1, , antall hidden lag er 4 med antall noder fra 1-4 er 50, 75, 30, 85. Dette var ikke resultatet med lavest MSE, men det var blant de laveste og det var litt støy men ikke for mye.

Dette resultatet ble brukt til å plotte MSE til test- og trenings-settet med epoke størrelse på , vist i figur (??). Det samme ble beregnet med epoke størrelse på , hvor det viser seg at MSE støyen øker rundt iterasjoner. Plottet ligger på github [6] med navn *Best\_hyperparameter\_e\_800000.png*.



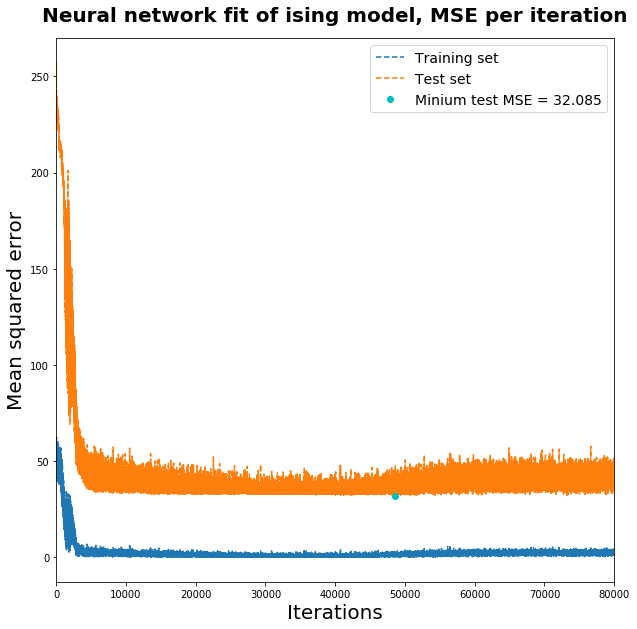
Figur 5 viser MSE til trening- og test-settet med de beste hyperparameterene funnet fra tidligere testingen. Se tekst for detaljer. Aktiveringsfunksjonen brukt er softsign-funksjonen.

Det beregnes også MSE med kombinasjonen fra rad 4 kolonne 2 fra github [6] med navn *Layer\_size\_testing.png.* , med samme antall hidden lag. Denne har mer MSE støy enn de beste parameteren. Plottet er vist i figur (??).



Figur 6 viser MSE til trening- og test-settet med . og funnet fra tidligere testingen. Se tekst for detaljer. Aktiveringsfunksjonen brukt er softsign-funksjonen

Det samme blir så beregnet, men med tanh-funksjonen som aktiveringsfunksjon. Vist i figur (??).



Figur 7 viser MSE til trening- og test-settet med de beste hyperparameterene funnet fra tidligere testingen. Se tekst for detaljer. Aktiveringsfunksjonen brukt er tanh-funksjonen.

**Diskusjon**

**Nevrale nettverk ising modell regresjon**

Det ble først avlet frem gode - og -kombinasjoner og , og , og . Hvis man ser på graden av ser vi at den er konstant. Det viser at for funksjonen som det tilpasses så er dette den gyldne kombinasjonen. Med en lav, men ikke for lav, -verdi går gradienten til vektene mot et den sanne verdien sakte men sikkert. Derimot når verdien økes så hindres vektene å bli for store. Måten det gjøres på er, ligning (9), å skyte gradienten mer i samme retning ved økende . Dette observerer vi i plottet på github [6] med navn *Eta\_lambda\_testing.png* som økende støy, fordi vektene fluktuerer hardere frem og tilbake. Dette kan virke positivt ved at MSEen ikke blir stående i et lokalt minimum, men fluktuerer nok til å gå forbi barrieren. I tillegg til at -verdien bidrar til dette, så gjør også det at implementeres batch gradient decent at sjansen for å bli stående i et lokalt minimum mye mindre.

Hyperparameter kombinasjonene brukes deretter til å finne antall hidden lag som gir best resultat. Dette viser seg å være 4, hvor den valgte kombinasjonen var og . Dette var ikke resultatet med lavest MSE, alle var ganske like med små variasjoner. Grunnen til at denne ble valgt var at MSE kjapt går mot et minimum og at den ikke har for mye støy. Hvis man velger å stoppe et sted for å bruke vektene får man da et stabilt resultat hver gang. Men det er derimot ikke for lite støy slik at den kan overvinne eventuelle lokale minimum. Plottet i figur (??).

For undersøke om det var et lokalt minimum, ble det beregnet MSE med en gode hyperparametere men som har litt mer støy vist i figur (??).

Det viser at de beste parameterene faktisk er de beste. For å undersøke om det kan hjelpe med en funksjon som har brattere overgang brukes tanh-funksjonen som aktiverings-funksjon. Denne funksjonen går fra -1 til 1, softsign funksjonen går derimot kun til 0.7. Det kunne tenkes at siden input har verdier fra -1 til 1 så vil dette kunne forbedre resultatet. Plottet i figur (7) viser derimot at minimum MSE er høyere enn med softsign og det gir økt støy. Dette kan komme av at energiene/verdiene som tilpasses inneholder 0. Hverken softsign eller tanh klarer å sette verdier til 0 fordi det er en bias og en helning rundt null, som betyr at de vil fluktuere frem og tilbake. Med tanh vil de fluktuere hardere siden helningen er brattere og kanskje er grunnen til at softsign-funksjonen fungerer bedre for denne funksjonen.

**Konklusjon**

Refereanser

[1] video om timer sovet osv.

[2] Slides Morten

[3] Slides backprop artikkelen

[4] Gradient decent Morten

[5] <https://en.wikipedia.org/wiki/Activation_function>

[6] Bilde på github