# Studie Område Projekt

Matematik A / Programmering B

Victor Østergaard Nielsen
3di - H. C. Ørsted Gymnasiet Lyngby
15/12/2024

Vejledere:
Jan Strauss Hansen (Matematik A)
Kristian Krabbe Møller (Programmering B)

# Indhold

# 1 Opgaveformulering

**Machine-learning**

**Hovedspørgsmål**: Hvordan kan et simpelt neuralt netværk, uden unødige abstraktioner eller biblioteker, anvendes tilgenkendelse af håndskrevne tal i realtid i et tegneprogram på computeren, og hvad er matematikken bag?

**Opgaveformulering**:

- Redegør overordnet for begrebet neuralt netværk.
- Redegør for de grundlæggende matematiske principper bag neurale netværk herunder matriceregning.
- Redegør for valg af programmeringssprog ift. udvikling af programmer med neuralt netværk.
- Analysér hvordan et neutralt netværk kan programmeres, gerne uden unødvendige abstraktioner eller biblioteker, så det kan genkende håndskrevne tal i realtid i/fra et tegneprogram på computeren.
- Undersøg hvorledes programmet kan optimeres for at opnå lavest mulig fejlrate og evt. hvorledes støj i den analyserede data kan påvirke fejlraten.
- Diskuter og vurder hvorvidt neutrale netværk er den mest effektive tilgang til at genkende håndskrevne tal på en adaptiv og robust måde.

# 2 Indledning

# 3 Matricer og matrixregning

## 3.1 Matricer

En matrice er en tabel af tal, der er arrangeret i rækker og kolonner. En matrice kan repræsenteres med et stort bogstav, f.eks. $A$, og elementerne i matricen kan repræsenteres som $a_{ij}$, hvor $i$ er rækken og $j$ er kolonnen. En matrice med $m$ rækker og $n$ kolonner kaldes en $m \times n$ matrice. En matrice med lige mange rækker og kolonner kaldes en kvadratisk matrice. En matrice med kun én række kaldes en rækkevektor, og en matrice med kun én kolonne kaldes en søjlevektor. (Lauritzen & Bökstedt, 2019)

$$\underbrace{A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}}_{m \times n \text{ Matrice}} \qquad \underbrace{B = \begin{bmatrix} b_{11} \\ b_{21} \\ \vdots \\ b_{m1} \end{bmatrix}}_{\text{Søjlevektor}} \qquad \underbrace{C = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1n} \end{bmatrix}}_{\text{Rækkevektor}} \tag{1}$$

Mens der i denne opgave ekskulstivt vil blive fokusseret på "2d"matricer, så er det også muligt at have "3d"matricer, hvor der er en dybde dimension. Dette kunne f.eks. være en matrice, der repræsenterer et billede, hvor der er en række og en kolonne for hver pixel, og en dybde for hver farvekanal.

## 3.2 Matrixregning

Matrixregning er en vigtig del af matematikken bag neurale netværk og er derfor vigtig at forstå. Der er flere forskellige operationer, der kan udføres på matricer, herunder addition, subtraktion, skalarmultiplikation og matrix-multiplikation m.m.

### 3.2.1 Addition og subtraktion

For at addere eller subtrahere to matricer skal de have samme dimensioner, altså de skal have samme mængde rækker og søjler. Givet dette, så er matmatkiken ikke meget andlernedes fra normale tal, det betyder at: $A + B = B + A$. Da de 2 matricer har samme dimension udføres addition og subtraktion således:

$$A + B = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{bmatrix} \tag{2}$$

Hver position i $A$ matricen bliver altså adderet eller subtraheret med samme position i $B$ matricen.

### 3.2.2 Skalarmultiplikation af matricer

Givet tallet $k$ kan man gange $k$ på matricen $A$ således:

$$k \cdot A = k \cdot \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} k \cdot a_{11} & k \cdot a_{12} \\ k \cdot a_{21} & k \cdot a_{22} \end{bmatrix} \tag{3}$$

### 3.2.3 Elementvis anvendelse af en funktion på matricer

Givet en funktion $f(x)$ og en $m \times n$ matrice $A$, vil $f(A)$ være en $m \times n$ matrice, hvor $f(x)$ er blevet anvendt på hvert element i $A$:

$$f(A) = f\left(\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}\right) = \begin{bmatrix} f(a_{11}) & f(a_{12}) & \dots & f(a_{1n}) \\ f(a_{21}) & f(a_{22}) & \dots & f(a_{2n}) \\ \vdots & \vdots & \ddots & \vdots \\ f(a_{m1}) & f(a_{m2}) & \dots & f(a_{mn}) \end{bmatrix} \tag{4}$$

### 3.2.4 Matrixmultiplikation

At gange to matricer sammen er lidt mere kompliceret, det indebærer først og fremmest at de 2 matricer er af kompatibel størrelse. Hvis $A$ er en $m \times p$ matrice og $B$ er en $p \times r$ matrice, så er $C = A \cdot B$ en $m \times r$ matrice. Bemærk at antallet af kolonner i $A$ matricen skal være lig antallet af rækker i $B$ matricen. For at finde elementet $c_{ij}$ i $C$ matricen ganges række $i$ i $A$ matricen med kolonne $j$ i $B$ matricen. Dette gøres ved at gange elementerne i række $i$ i $A$ matricen med elementerne i kolonne $j$ i $B$ matricen og summere dem. F.eks. hvis $A$ er en $3 \times 2$ matrice og $B$ er en $2 \times 3$ matrice, så er $C$ en $3 \times 3$ matrice, og elementet $c_{11}$ i $C$ matricen findes således (Simonson, 2015):

$$c_{11} = a_{11} \cdot b_{11} + a_{12} \cdot b_{21} \tag{5}$$

Intuitivt kan dette visualiseres ved at tegne $A$ og $B$ matricerne således (Simonson, 2015):

$$\begin{bmatrix} & & \\ a_{11} & a_{12} & \\ a_{21} & a_{22} & \\ a_{31} & a_{32} & \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix} \begin{bmatrix} & & \\ & & \end{bmatrix} \tag{6}$$

For at finde elementet $c_{11}$ i $C$ matricen, ganges række 1 i $A$ matricen med kolonne 1 i $B$ matricen, dette er visualiseret herunder (Simonson, 2015):

$$\begin{bmatrix} & & \\ a_{11} & a_{12} & \\ a_{21} & a_{22} & \\ a_{31} & a_{32} & \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix} \begin{bmatrix} c_{11} & & \\ & & \end{bmatrix} \qquad c_{11} = a_{11} \cdot b_{11} + a_{12} \cdot b_{21} \tag{7}$$

Samme operation gentages for resten af positionerne i $C$ matricen:

$$\begin{bmatrix} & & \\ a_{11} & a_{12} & \\ a_{21} & a_{22} & \\ a_{31} & a_{32} & \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix} \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \tag{8}$$

Det ses nu visuelt at den resulterende matrice $C$ er en $3 \times 3$ matrice når $A$ er en $3 \times 2$ matrice og $B$ er en $2 \times 3$ matrice. Det skal dog bemærkes at matrixmultiplikation ikke er kommutativ, altså $A \cdot B \neq B \cdot A$. Dette kan også ses visuelt ved at bytte om på $A$ og $B$ matricerne:

$$\begin{bmatrix} & & \\ a_{11} & a_{12} & \\ a_{21} & a_{22} & \\ a_{31} & a_{32} & \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix} \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \qquad \begin{bmatrix} & & \\ b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \tag{9}$$

Det ses at $A \cdot B$ og $B \cdot A$ ikke er ens, og derfor er matrixmultiplikation ikke kommutativ. (Lauritzen & Bökstedt, 2019) Selv med to kvadratiske matricer af samme dimension er matrixmultiplikation ikke nødvendigvis kommutativ. Dette kan betragtes i følgende eksempel:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \tag{10}$$

$$A \cdot B = \begin{bmatrix} 1 \cdot 5 + 2 \cdot 7 & 1 \cdot 6 + 2 \cdot 8 \\ 3 \cdot 5 + 4 \cdot 7 & 3 \cdot 6 + 4 \cdot 8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix} \qquad B \cdot A = \begin{bmatrix} 5 \cdot 1 + 6 \cdot 3 & 5 \cdot 2 + 6 \cdot 4 \\ 7 \cdot 1 + 8 \cdot 3 & 7 \cdot 2 + 8 \cdot 4 \end{bmatrix} = \begin{bmatrix} 23 & 34 \\ 31 & 46 \end{bmatrix} \tag{11}$$

Det ses at $A \cdot B \neq B \cdot A$, matrixmultiplikation er altså ikke kommutativ hverken i den resulterende størrelse i ikke kvadratiske matricer eller i kvadratiske matricer af samme størrelse. (Lauritzen & Bökstedt, 2019)

### 3.2.5 Transponering af matricer

Transponering af en matrice betyder at bytte om på rækker og kolonner. Hvis $A$ er en $m \times n$ matrice, så er transponeringen af $A$ en $n \times m$ matrice, og elementet $a_{ij}$ i $A$ matricen bliver til elementet $a_{ji}$ i $A^T$ matricen. Dette kan visualiseres ved at tegne $A$ matricen og $A^T$ matricen:

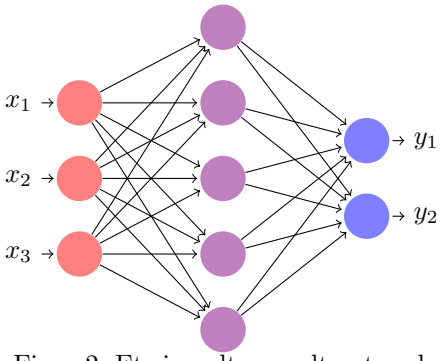
$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \qquad A^T = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \tag{12}$$

# 4 Neurale netværk

## 4.1 Introduktion

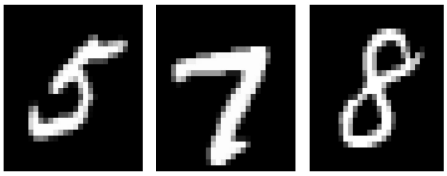
Et neuralt netværk er en matematisk model, der er inspireret af de biologiske neuroner i menneskehjernen. Et neuralt netværk består af en række lag, hvor hvert lag består af neuroner. Dette kan ses på Figur 2. Hvert neuron i et lag er forbundet til alle neuroner i det forrige lag og det næste lag. Hver forbindelse mellem neuronerne har en vægt, der bestemmer, hvor meget signalet fra det ene neuron påvirker det næste neuron. Hvert neuron har også en bias, der bestemmer, hvor let det er for neuronet at sende et signal. Et neuralt netværk består af et inputlag, et eller flere skjulte lag og et outputlag. Dette er illustreret på Figur 2. Det er altså denne model, der er inspireret af de mange sammenkoblede neuroner i menneskehjernen. Denne lighed er ikke tilfældig, da neurale netværk er designet til at efterligne hjernens evne til at lære.

Figur 1: Neuroner i menneskehjernen fra (St. Clair, 2021)




Figur 2: Et simpelt neuralt netværk

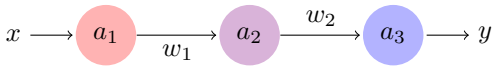
Bemærk Figur 3. Du ved udmærket godt, hvilke tal der er på billedet, selvom du aldrig før har set netop dette 5-, 7- og 8-tal før. Dette er, fordi din hjerne er trænet til at genkende tal. Denne opgave er enormt udfordrende for et alment computerprogram, idet det ikke har nogen intuitiv forståelse af, hvad et tal er, og det skal derfor programmeres med specifikke instruktioner for at kunne genkende tal. Denne tilgang er ikke optimal, da den kræver, at programmøren har en dyb forståelse af, hvordan tal ser ud, og hvordan de kan genkendes. Denne metode er ikke holdbar i længden, idet der er uendeligt mange måder at skrive et 7-tal på. For at kunne generalisere talgenkendelse og gøre metoden mere robust overfor nye skrivemåder af tal kan neurale netværk anvendes. Antag et neuralt netværk med et tal som input og de helt rigtige vægte og biases. Denne model vil i teorien kunne genkende tal, også selvom modellen aldrig før har set netop dette tal. Håbet er, at modellen har "lært"at generalisere træningsdataen til en mere generel forståelse af tal. Modellens output vil derfor være en søjlevektor med sandsynligheder for, at inputtet er et tal fra 0 til 9. Modellens prediktering vil være det tal, der har den højeste sandsynlighed ifølge modellen.



Figur 3: Eksempel fra MNIST datasættet (LeCun m.fl., 1994)

### 4.1.1 Feedforward

Når modellen skal prediktere, altså gå fra input til output, kaldes dette for *feedforward*. Her sendes inputtet gennem alle lagene i modellen, og bliver bliver påvirket af vægtene mellem neuronerne og biaset i hvert neuron. Herunder er et simpelt neuralt netværk med kun 1 neuron i hvert lag og et skjult lag og hvor $a_n$ er aktiveringen af neuronen i lag $n$ og $w_n$ er vægten mellem neuronen i lag $n$ og lag $n+1$:



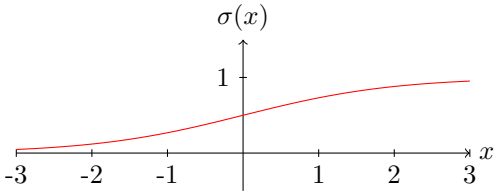

Figur 4: Et simpelt neuralt netværk

Så hvis vi ønsker at prediktere outputtet $y$ givet inputtet $x$, så kan vi gøre dette ved at følge disse trin:

$$a_1 = x \tag{13}$$
$$a_2 = \sigma(w_1 \cdot a_1 + b_1) \tag{14}$$
$$a_3 = \sigma(w_2 \cdot a_2 + b_2) \tag{15}$$
$$y = a_3 \tag{16}$$

Her er $\sigma(x)$ en aktiveringsfunktion, der tager inputtet $x$ og returnerer et output. Denne funktion er essentiel for at modellen kan lære mere komplekse femomener, da den introducerer ikke-linearitet i modellen. En af de mest brugte aktiveringsfunktioner er ReLU, der tager inputtet $x$ og returnerer $x$ hvis $x > 0$ og 0 ellers. (Sanderson, 2017) Hvis outputtet skal betragtes som en sandsynlighed, er sigmoid funktionen en god aktiveringsfunktion, da den tager inputtet $x$ og returnerer en værdi mellem 0 og 1, som kan tolkes som en sandsynlighed. Sigmoid funktionens definition samt et plot af funktionen er vist herunder: (Nielsen, 2019)

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{17}$$



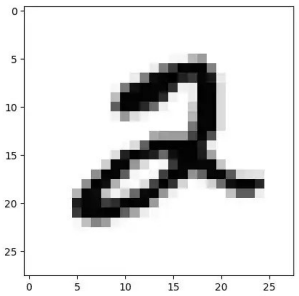

Figur 5: Sigmoid funktionen

Typisk har et neuralt netværk flere neuroner i hvert lag, og antallet af vægte og biases er derfor meget større. $x$, $y$, samt alle de forskellige $a_n$ og $b_n$ for hvert lag er søjlevektorer, og $w_n$ i alle lag er matricer. og derfor skal der bruges matrixmultiplikation for at kunne beregne outputtet. Dette er ikke et problem, da de pågældende regneoperationer er defineret tidligere i afsnittet.

### 4.1.2 Træning af neurale netværk

Når et neuralt netværk initialiseres, er alle vægtene og biases tilfældige. (Sanderson, 2017) Dette betyder, at modellen ikke kan genkende noget, ligesom et barn, der skal lære noget for første gang. For at lære modellen at genkende tal, skal modellen trænes. Dette kræver en anstændig mængde træningsdata og associerede labels, der fortæller modellen, hvad hvert billede er. For at træne modellen skal man vide hvor god modellen er til at prediktere datasættet, en primitiv måde at gøre det på er at angive procenten af rigtigt predikterede tal. Denne metode er ikke velfungerende, da en lille ændring i vægtene og biases vil ændre modellens præstation minimalt og er derfor ikke målbar. For at kunne træne modellen effekt skal vi vide præcist hvordan en lille ændring i vægtene og biases påvirker modellens præstation. Denne funktion kaldes en *loss funktion*, og er essential for at kunne træne modellen. Loss funktionen er en funktion, der tager modellens prediktion og sammenligner den med det rigtige svar. Loss funktionen er en måde at kvantificere, hvor god modellen er til at prediktere. Hvis loss funktionen er høj, er modellen dårlig til at prediktere, og hvis loss funktionen er lav, er modellen god til at prediktere. herunder ses et eksempel på en loss funktion på et enkelt datapunkt $i$ (Sanderson, 2017):

$$\hat{y}_i = \underbrace{\begin{bmatrix} 0.12 \\ 0.03 \\ 0.25 \\ 0.07 \\ 0.18 \\ 0.09 \\ 0.04 \\ 0.11 \\ 0.06 \\ 0.05 \end{bmatrix}}_{\text{Modellens prediktion}} \qquad y_i = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\text{Det rigtige svar}} \tag{18}$$



Figur 6: Eksempel på et 2-tal fra MNIST datasættet (LeCun m.fl., 1994)

Loss funktionen for dette datapunkt er den kvadrerede forskel mellem modellens prediktion og det rigtige svar:

$$L_i = (y_i - \hat{y}_i)^2 = \begin{bmatrix} 0 - 0.12 \\ 0 - 0.03 \\ 1 - 0.25 \\ \vdots \\ 0 - 0.05 \end{bmatrix}^2 = \begin{bmatrix} -0.12^2 \\ -0.03^2 \\ 0.75^2 \\ \vdots \\ -0.05^2 \end{bmatrix} \tag{19}$$

Nu tages summen af den kvadrerede forskel for at få loss funktionen for dette datapunkt:

$$L_i = \sum_{i=1}^{10} (y_i - \hat{y}_i)^2 \tag{20}$$

$$= (-0.12^2) + (-0.03^2) + (0.75^2) + \ldots + (-0.05^2) \tag{21}$$

Denne operation gentages for alle datapunkter i træningssættet, og summen tages, derefter divideres summen med antallet af datapunkter for at få gennemsnittet af loss funktionen for hele træningssættet (Sanderson, 2017):

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{10} (y_{ij} - \hat{y}_{ij})^2 \tag{22}$$

Nu har vi en loss funktion, der kvantificerer, hvor god modellen er til at prediktere, og hvor selv en lille ændring i vægtene og biases vil ændre loss funktionen. Nu er det muligt at træne modellen ved at minimere loss funktionen, da man kan sige at når loss funktionen er lav, er modellen god til at prediktere. Desuden er denne loss funktion kontinuert og differentiabel, hvilket er essentielt for at kunne træne modellen. For at minimere loss funktionen bruges en algoritme kaldet *gradient descent*.

## 4.2 Gradient descent

Gradient descent er en algoritme, der tager loss funktionen og beregner gradienten af loss funktionen i forhold til alle modellens parameter (vægtene og biases). Gradienten er en vektor, der peger i retningen af den største stigning af loss funktionen. For at minimere loss funktionen skal vi derfor gå i den modsatte retning af gradienten. Dette gøres ved at opdatere vægtene og biases i modellen med gradienten ganget med en konstant, kaldet *learning rate*. Dette gentages indtil loss funktionen er minimeret tilstrækkeligt. (IBM, 1994; Nielsen, 2019; Sanderson, 2017)

Antag at vi organiserer modellens vægte og biases i en søjlevektor $\vec{W}$ og at loss funktionen er $L(\vec{W})$. Gradienten af loss funktionen er $\nabla L(\vec{W})$, derfor beskriver søjlevektoren $-\nabla L(\vec{W})$ hvordan vi kan opdatere $\vec{W}$ for at minimere loss funktionen og dermed øge modellens præstation. Algorimen som finder gradienten på baggrund af modelles parametre og loss funktionen kaldes *backpropagation*. (Nielsen, 2019; Sanderson, 2017) Det vil ses nærmere på, hvordan backpropagation fungerer i et kommende afsnit. Indtil videre antager vi blot, at den fungerer som påstået og returnerer det passende gradient for modellens parametre.

## 4.3 Differensiabilitet

# Litteratur


IBM. (1994). What is Gradient Descent? — IBM. https://www.ibm.com/topics/gradient-descent

Lauritzen, N., & Bökstedt, M. (2019). Tal og Lineær Algebra 2019 4 Matricer. https://data.math.au.dk/interactive/lintrans/Chapters/vektorerogmatricer.html

LeCun, Y., Cortes, C., & Burges, C. J. (1994). MNIST handwritten digit database. https://yann.lecun.com/exdb/mnist/

Nielsen, M. (2019). Using neural nets to recognize handwritten digits. http://neuralnetworksanddeeplearning.com/chap1.html

Sanderson, G. (2017). Gradient descent, how neural networks learn. https://www.3blue1brown.com/lessons/gradient-descent

Simonson, M. (2015 oktober). Matrix Multiplication Made Easy. https://blogs.ams.org/mathgradblog/2015/10/19/matrix-multiplication-easy/

St. Clair, B. (2021 april). Explainer: What is a neuron? https://www.snexplores.org/article/explainer-what-is-a-neuron