

## Forelesning 2

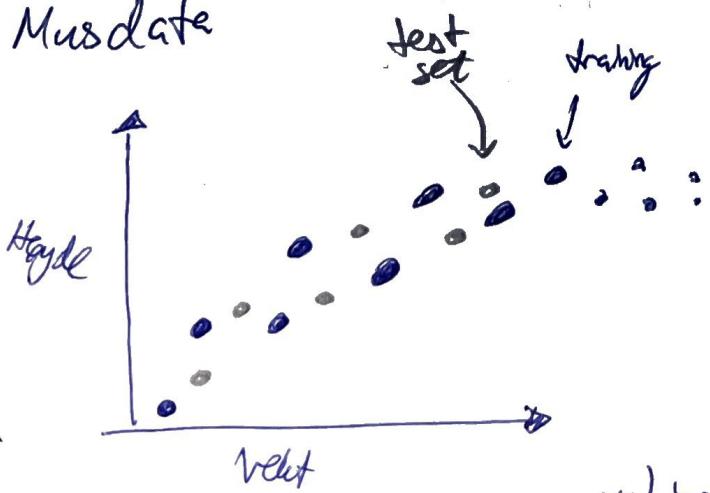
- Fra sist forelesning  
Date innturer, vektorer  
Tensorer  
Linear algebra  
Namn  
Kalkulus → Derivasjon  
Statistikk → Irreducible og Reducible error  
Maskinlæring  
Regresjon vs klasifikasjon  
Predisjon vs Infors  
Overfitting  
Trenings-, validitets-, test-sett

- PowerPoint i starten.

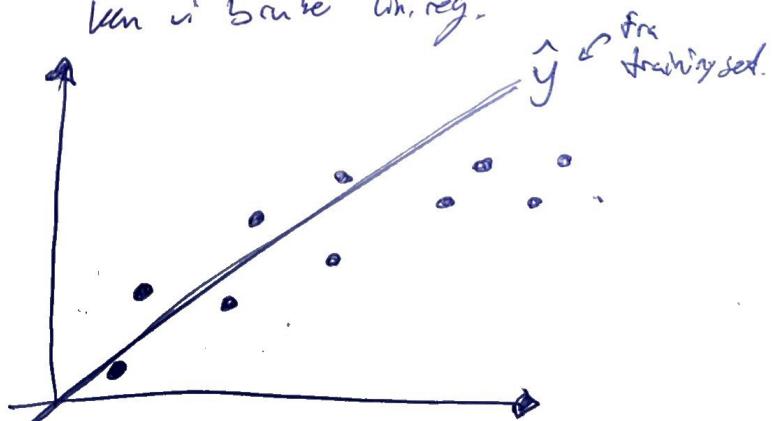
- Gå i gjennom ~~og~~ hvilke maskinlæringsteknikker vi har i kompendiet, som er selvstudium.

## Bias - Variance Tradeoff

- Musdata

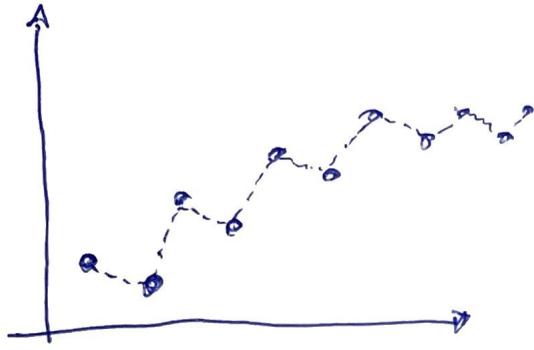


Før å lage en modell for dette  
kan vi bruke lin. reg.



\* Linear regresjon har ikke fleksibiliteten  
til passe kurver til datene.

\* Lin. reg. kan altså ikke fange opp det  
ekte forholdet mellom vekt/høyde. Denne  
mangelnde egensteigen er bias.



En annen ML teknikk kan "fritte" den kurvede datanen, fordi den er hyperflexibel!

↳ Denne har derfor lite bias. (Hvordan forskjell fra lin. reg.)

→ Hvor viser på MSE av treningsdatene, vil lin. reg. få en del MSE, mens den fleksible ML-teknikken vil få 0.

→ Regner ut MSE for test sett, vil fortlin. reg. være?

→ Denne forskjellen i "fits" (mellan training/test sett) er en konsekvens av varians. Statistisk sett, så refererer varians til <sup>den</sup> andelen prediksjonene vil endre seg, hvis vi "fitter" modellen til et annet treningssett.

lar bres og flexibel } kan være veldig bra  
høy varians } kan være overfitted

høy bias og lite flexibel } kan gi gode prediksjoner  
lav varians } men ikke driftbar.  
↳ mer konstant!

Ideelt i ML:  
- En modell som har:  
    - lav bias, og kan være en modell  
        for det faktiske forholdet  
    - lav varians, ved å gi konstante prediksjoner  
        på ulike datasett.

- Dette lykkes vi med om vi finner sweet spotet mellom en enkel modell og kompleks modell

= mer data  
= mer regulering

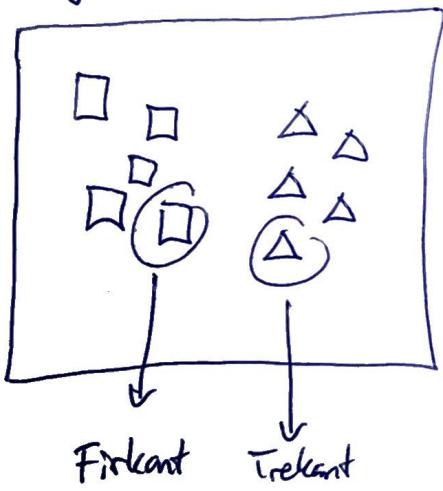
- annen modell: overfitted...

## Maskinlæring

Vi deler hovedsakelig maskinlæring inn i to.

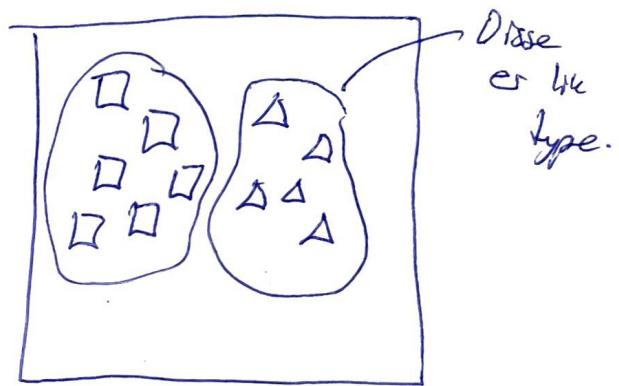
### Veiledet læring

- Merket data
- Bruker treningsdata
- Brukes for prediksjon
- Klassifikasjon eller regression

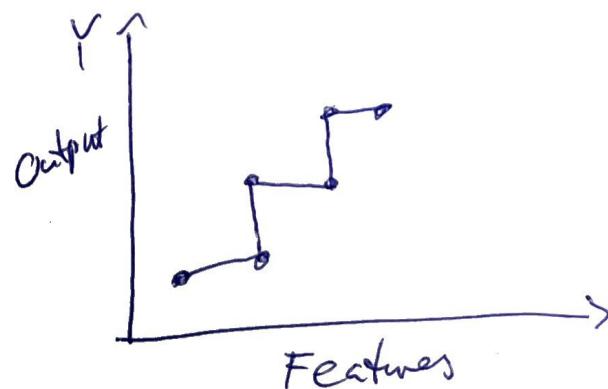
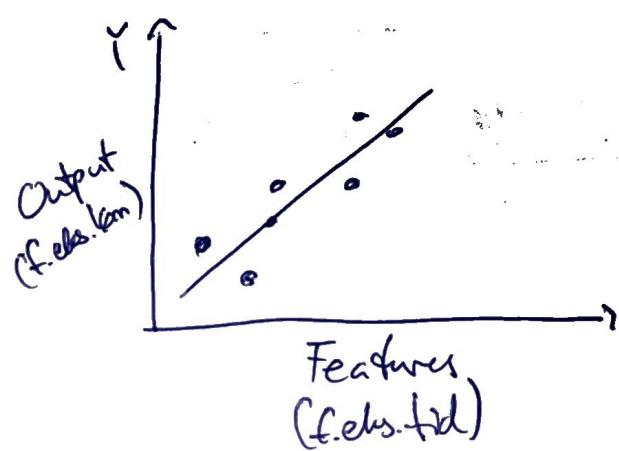


### Ikke veiledet læring

- Umerket data
- Bruker all data som input
- Klustering



## Overfitting



"Når vi skal drive med prediksjon ønsker vi å ha en modell som kan (prøve å) forutsi noe om fremtida."

For å gjøre det, så trenger vi data, masse data!

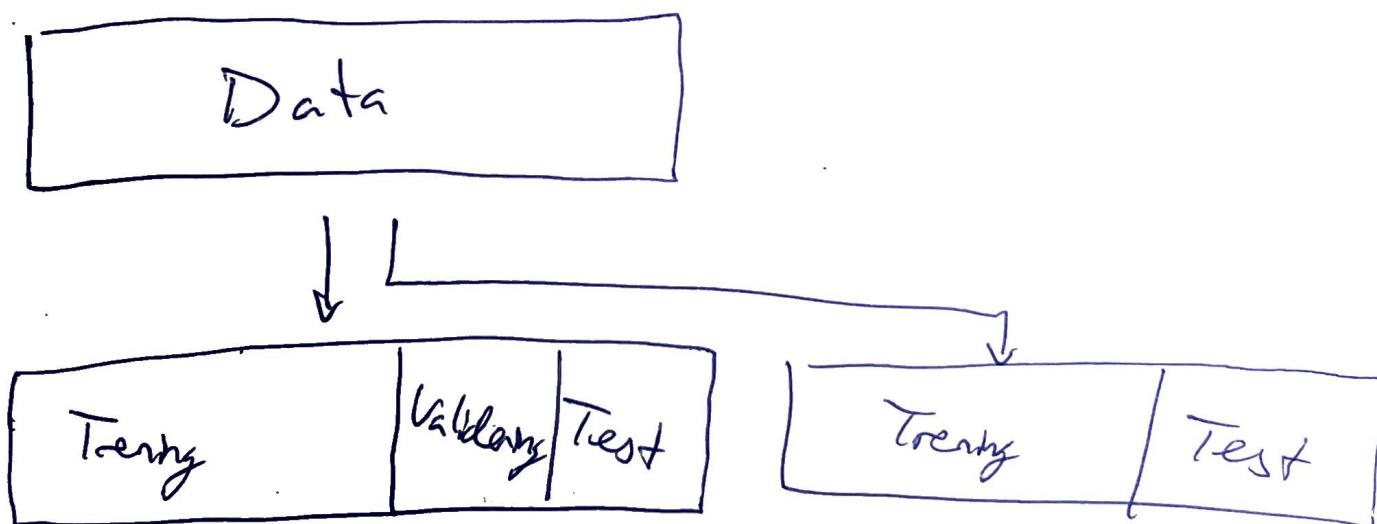
## Data

Universell oppnåelighetskorenmet:

Hvis vi har en eller annen funksjon  $f(x)$  som er kontinuert, kan vi bruke et neuralnettverk til å representere/etterligne denne funksjonen.

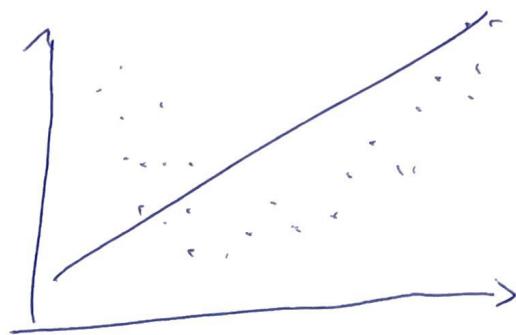
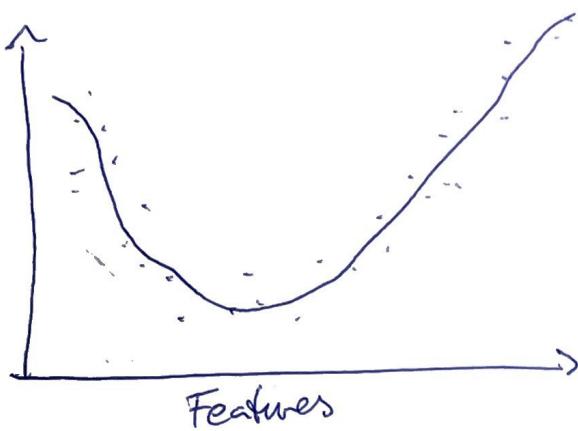
Men, vi er avhengig av data!

"Slik vi gjør i maskinlæring, når vi ønsker å predikere noe er:"



"Snakke litt om hvorfor"

## Overfitting vs Underfitting



"Temperatur dag til dag, da  
er det lurt på nikk"

"Dette handler om og viser hvorfor vi deler opp dataene  
vært i treningsdata og test data."

Spm: Hva skjer hvis vi bruker all data til trenings?

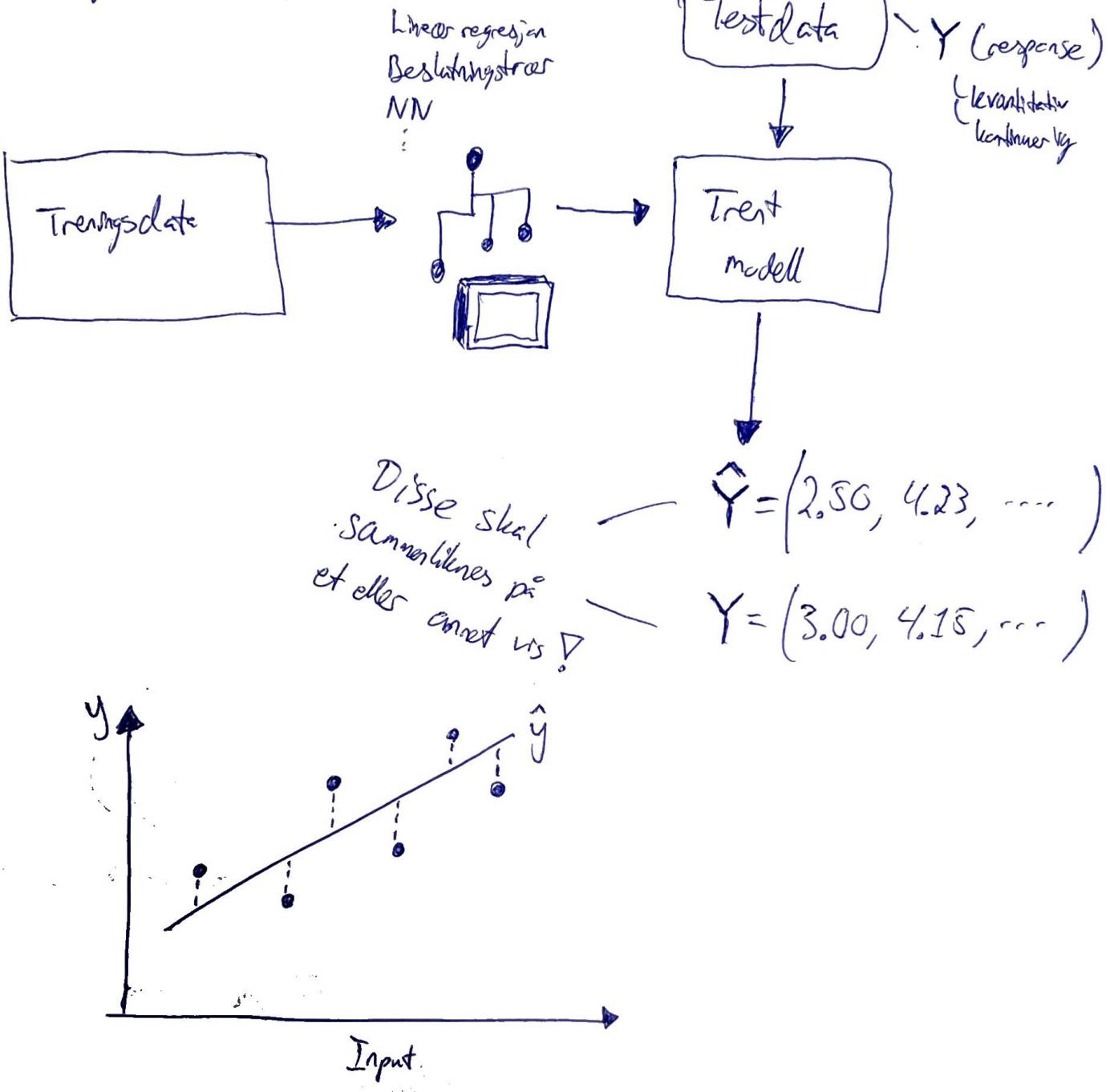
Hva kan vi hindre over- eller underfitting?

- Tidlig stopp under treninger av neurale nettverk.
- Kryss validering
- Dropout teknikker (kan forhindre ≈ plukke opp støy)
- Bruke enklere modeller (det enkle er ofte det beste)
- Ensemble metoder.
- Endre forhold i trenings/Test data. Færre treningsdata.)  
- Vanskelys i fine perfekt balanse.

## Evaluering av modeller

"Hordan vi evaluerer maskinlæringsmodellene våre avhenger av om vi har et klassifikasjonsproblem eller regressjonsproblem."

## Evaluering av regressjonsmodeller



$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

$$RMSE = \sqrt{MSE}$$

## MSE

## MAE

### Fokus

- | Straffer større feil mer (pga "i andre")

Behandler alle feil likt.

### Sensitivitet for outliers

- | Veldig sensitiv for outliers (pga "i andre")

Mindre sensitiv for outliers

### Tolkning

- | Verden er kubert, og gjør tolking vanskelig

Verdene er i samme enhet som måleværdi, og lettere å tolke.

### Optimalisering

- | Lettere å optimisere matematisk (den deriverte er glatt og kontinuerlig)

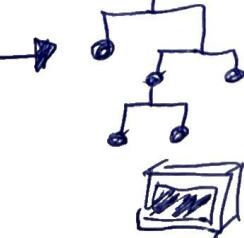
Kan føre til utfordringer (ikke derivbar i 0)

(etter når predikent verdier ikke samme verdier)

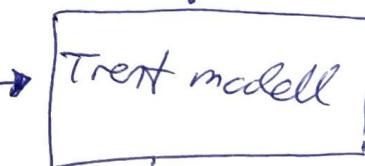
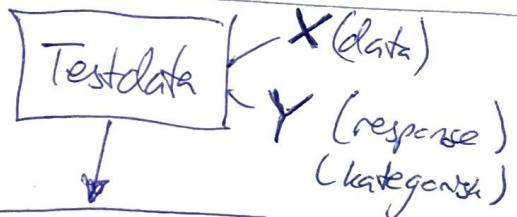
## Evaluering av klassifikasjonsmodeller



Logistisk reg.  
NN  
Beslutningskroer



Disse skal sammenlignes på et eller annet vis



$$\begin{aligned} \hat{Y} &= (0, 0, 0, 1, 1, 0, 1, \dots) \\ Y &= (1, 0, 0, 1, 1, 0, 1, \dots) \end{aligned}$$

ekse:

- $X$  - økonomiske/bankdata
- $Y$  - responser - 0 = ikke-svindel  
1 = svindel

Når vi skal sammenligne predikerte mot faktiske output av klassifikasjonsproblemer, må vi introdusere confusion matrix!

		0	1
Faktisk	0	True negative	False positive
	1	False Negative	True positive

Bhørt tilfelle!  
Viktig: 

- Som er positiv, og har som er negativ!

Predikert

"Ut fra matrisen over er det en relativ metode å evaluere hvor god en modell er."

Accuracy:

$$\frac{TN + TP}{TN + FP + FN + TP}$$

( "Andel korrekte predikasjoner av totale datapunkter" )

Precision:

$$\frac{TP}{TP + FP}$$

( "av alle pos. predikasjoner, hvor mange er faktisk positive? " )

Recall (sensitivitet):

$$\frac{TP}{TP + FN}$$

( "Andelen riktige klassifiserte positive, av alle faktiske positive" )

F1-Score:

$$2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

( "Harmonisk gjennomsnitt av Precision og Recall. Dette balanserer de to metrikkene, spesielt ved ubalanserte datasett" )

## Eksempel:

Binært problem, sunde/døde

Data: 0 (sunde sunde) : 9900 samples (99%)  
1 (sunde) : 100 samples (1%)

## Confusion matrix:

		0	1
		0	1
Faktisk	0	9850	50
	1	95	5

Prediktert

Accuracy:  $\frac{9850 + 5}{9850 + 50 + 95 + 5} = \frac{9855}{10000} = 98,55\%$  Bra!

Precision:  $\frac{5}{5 + 50} = \frac{5}{55} = 9,09\%$  Ikke bra!

Recall:  $\frac{5}{5 + 95} = \frac{5}{100} = 5\%$  Ikke bra!

F1:  $2 \cdot \frac{0,0909 \cdot 0,05}{0,0909 + 0,05} \approx 6,45\%$  Ikke bra!

Ubalansert datasett for koseknuser!

- Times teknisker med døde,
- oversampling (SMOTE)
- weighted loss function
- undersampling

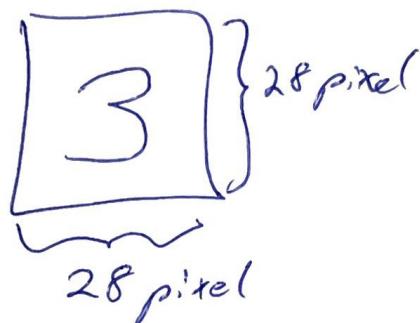
## Forelesning 2 - Maskinlæring for elektromer

- Hverfer er

$\boxed{3}$ ,  $\boxed{3}$  og  $\boxed{3}$

alle det samme tall?

- Hvordan kan



et dataprogram klare  $\Rightarrow$  si at dette er 3?!

- Neurale netværk kommer i mange former

$\hookrightarrow$  CNN  $\rightarrow$  Bræ for bølger

$\hookrightarrow$  LSTM  $\rightarrow$  Bræ for tale, tidsserier

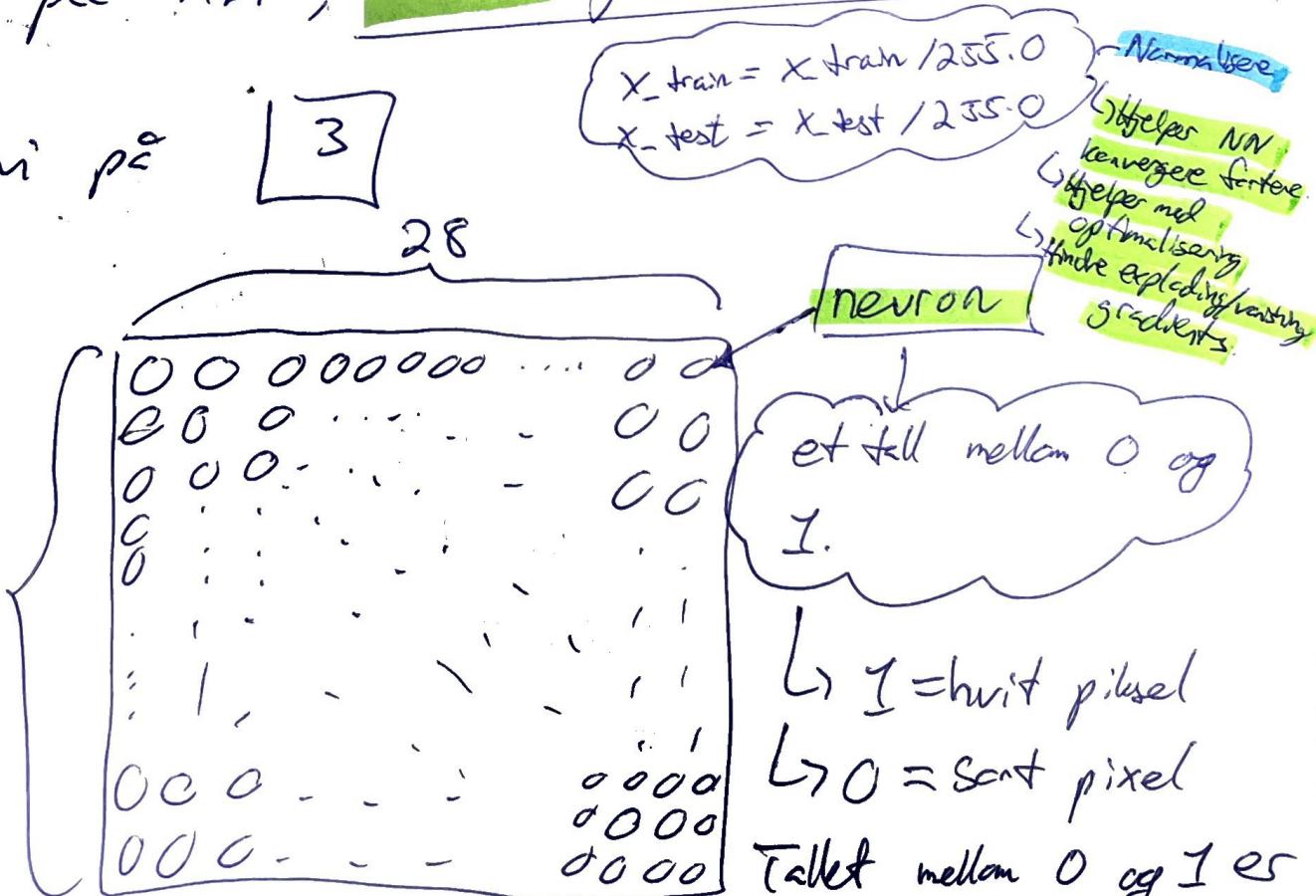
Viser på MLP; Multi layered Perceptron

Tænk vi på

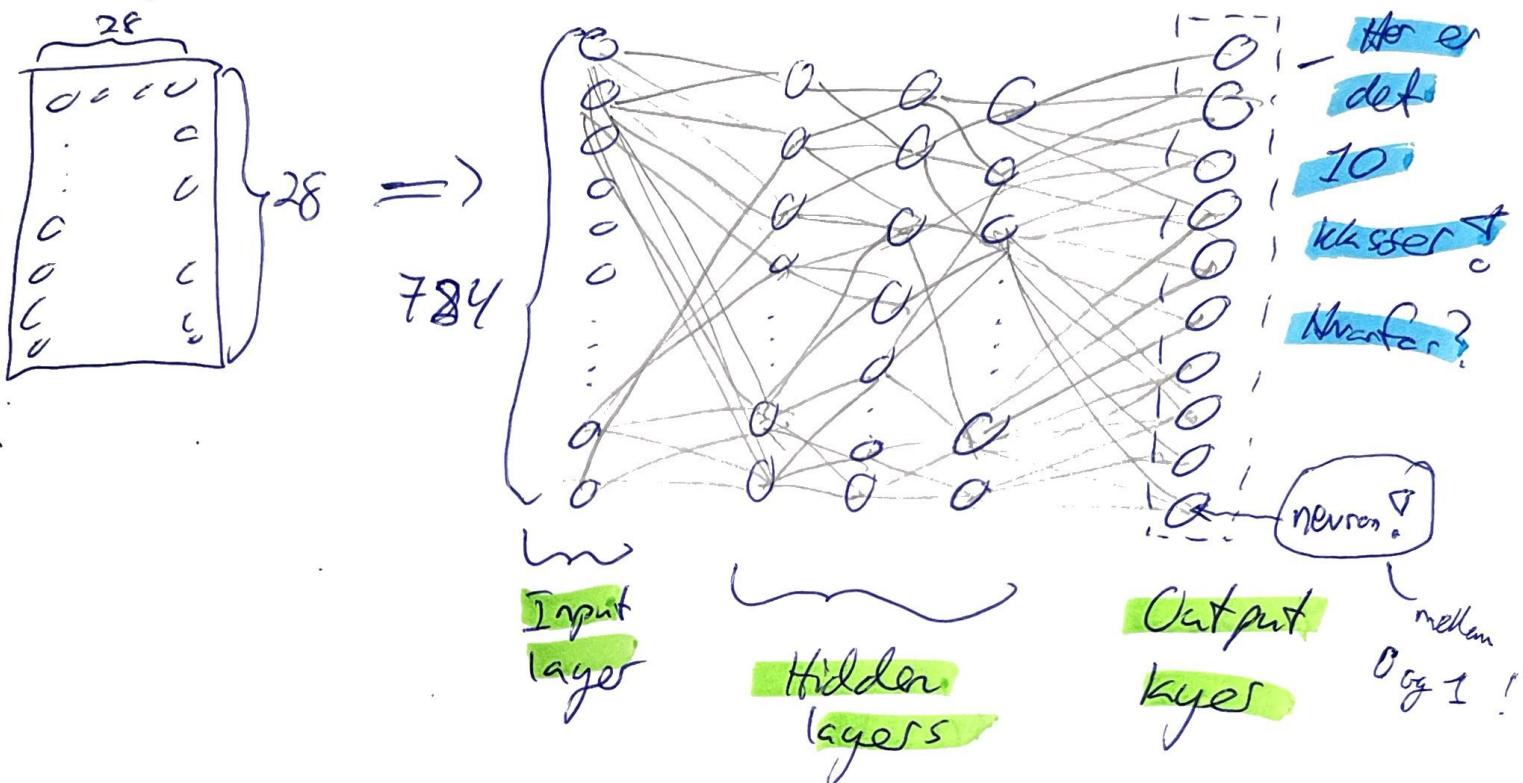


28

28



- Derfor er analogien til hjernen en ing, med neuroner som har et aktionspotensial med en aktivitet!  
 ↳ I hjernen, blir det enten full aktivitet (1) eller ingen aktivitet (0). AI → kan ha  $[0, 1]$ !

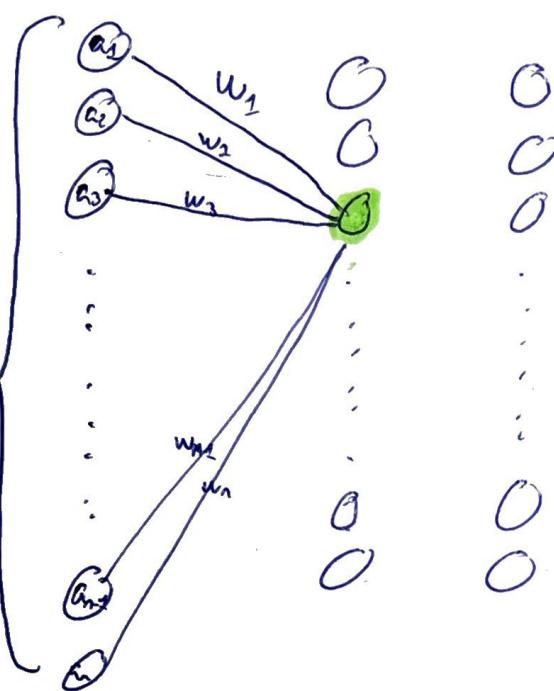


- Aktiveringen i ett lag bestemmer aktiveringen i det neste laget!
- Det neuronet : output layer med høyest verdi, er det sett nethvert fra 5! (det er !)

Dyp læring: Bruker ANN til å lære fra data. Det har også skjulte lag.  
 Til forskjell fra f.eks. beslutninger.

Så hvordan får vi informasjoner fra ett lag til det neste?!

784



Output lag (0-9)	
0	0
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1

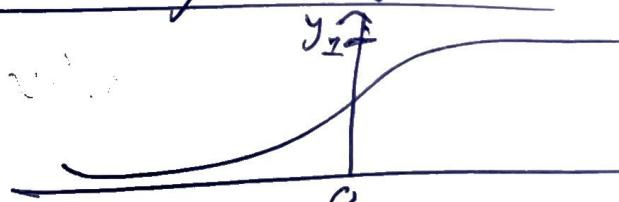
- Hvert neuron har en aktivitet (mellan 0 og 1), vi regner ut en vektet sum.

$$(w_1 a_1 + w_2 a_2 + w_3 a_3 + \dots + w_n a_n)$$

→ Dette blir et tall mellom "-∞" og "∞". Vi vil ikke mellom 0 og 1!

→ Derfor har vi aktivitetsfunksjoner!

F.eks. Sigmoid



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

Vis kode for ulike aktivitetsfunksjoner.

$$\sigma(w_1 a_1 + \dots + w_n a_n)$$

- Noen ganger vil man
- Da vet vi "hva posisjon" den velteide summen er! Gjennom en akt. funksjon.

↳ Men kanskje vil du ken at neuronet blir aktivert når den velteide summen er over 20.

↳ Da trenger vi en bias for å ta hensyn til dette og gjøre den aktiv!

$$\sigma(w_1a_1 + w_2a_2 + \dots + w_na_n - b)$$

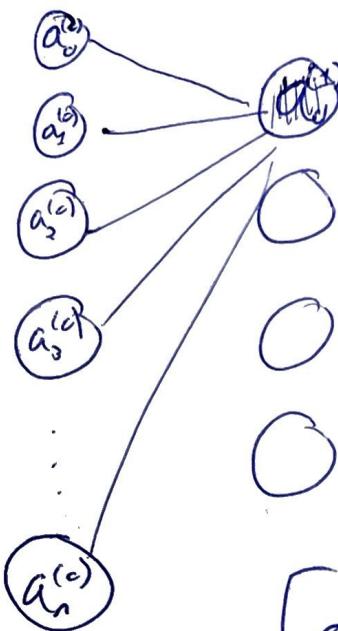
bias, freks. 20

• Nå takle vi føre len ett neuron, så det er 784 weights/neuron! I tillegg her de hversh bias!

$$\begin{cases} (\text{Input}) & (\text{Første Hidden}) & (\text{Andre Hidden}) \\ 784 \cdot 16 + 16 \cdot 16 + 16 \cdot 10 = 12\,960 \text{ weights} \\ \text{bias} & \text{bias} & \text{bias} \\ 16 + 16 + 10 = 42 \text{ bias'er} \end{cases}$$

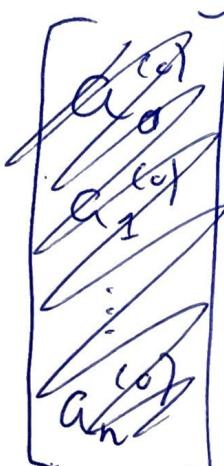
→ Totalt. 13002

"Læring" betyr å fine (at computeren) finner <sup>verdier</sup> for alle disse verdiene og biasene!



$$a_0^{(1)} = \sigma(w_{0,0}a_0^{(0)} + w_{0,1}a_1^{(0)} + \dots + w_{0,n}a_n^{(0)} + b_0)$$

Dette kan skrives som



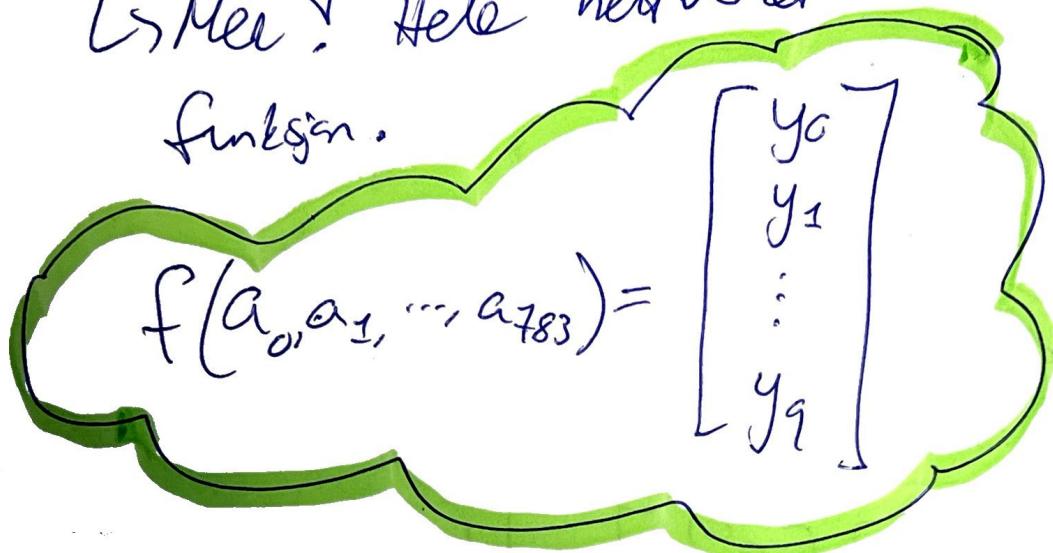
$$\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} \neq \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}$$

$$a^{(1)} = \sigma(Wa^{(0)} + b)$$

Fra input layer til første  
hidden layer!

Neuron kan ses ut som en funksjon, som tar output til alle neuroner fra det tidligere laget (akt., weight, bias) og spytter ut et tall mellom 0 og 1.

↳ Mer? Hele nettverket kan sees på som en funksjon.



## Kodelab #2

{ - oppgaver lagt ut  
{ - prosjekt  
{  
-----

selvstudium (?)

- Tensor operasjoner → TensorFlow dokumentasjon

- Struktur av ANN → scikit learn  
↳ Akt. funksjoner

$2^8 = 256$  = Sant/Mitt bilder normalisert

- MNIST → ↳ En b.t  $\rightarrow$  0  
↳ En byte, er en bitgruppe av 8 bit

↳ Mulige komb.  $2^8 = 256$

- Gå gjennom Sequential model.

↳ Logits → softmax før seg selv  
↳ Inne i Sequential

↳ Optimizer = 'adam' → "lært"  
↳ Finn beste veldt/loss.

↳ Loss-funksjon → TensorFlow dokumentasjon

"En epoch betyr et mønster"

↳ Epoch: har sett alle eksemplene i treningsdelen én gang, og gjort en predikjon per eksempel. Etter hver epoch, oppdaterer modellen sine parametere, basert på error/loss'et som det fikk fra predikjonene.

↳ 1875: Treningssettet er ca 60000 bilder

784	10
0	0
0	0
0	0
0	0
:	:
0	0
0	0
0	0

veldt loss

En batch er en delmenge av treindata som brukes i ett steg av treningen. Batch size er antall samples i en batch.

Standard i modell.fit ( $x\_train, y\_train, epochs=10$ ) er batch\_size=32.

$$\frac{60000}{32} = 1875$$

Større batch size = flere iterasjoner kan være mindre presis  
mindre batch size = kortere tid, men også mindre presis

## Nøkkelkonsept

Epoch: En komplett gennfang av hele datasettet i treninger.

Batch: En delmenge av datasettet brukt i en iterasjon.

Batch size: Antall samples i en batch.

Iterasjoner: Antall ganger modellen oppdaterer verktene sine per epoch.

## Scenario :

- 60000 trådby samples
- batchsize = 32 og epoch = 10

⇒ Hver batch har 32 samples

$$\text{Iterasjoner per epoch} = \frac{60.000}{32} = 1875$$

⇒ Epoch = 10, modellen trenes 10 fulle passeringer av datane.

## Oppsummering

- En batch er brukt i en epoch
- Hele datasettet er brukt én gang, per epoch.
- Hvis batch\_size = 32, så prosesserer modellen 32 samples av ganger, før den oppdaterer verktene.
- Hvis epoch = 10, så repeterer modellen denne prosessen 10 ganger over hele datasettet.