Norsk Regnesentral
NORWEGIAN COMPUTING CENTER

Lynkurs i matte + Logistisk regresjon (del 2)

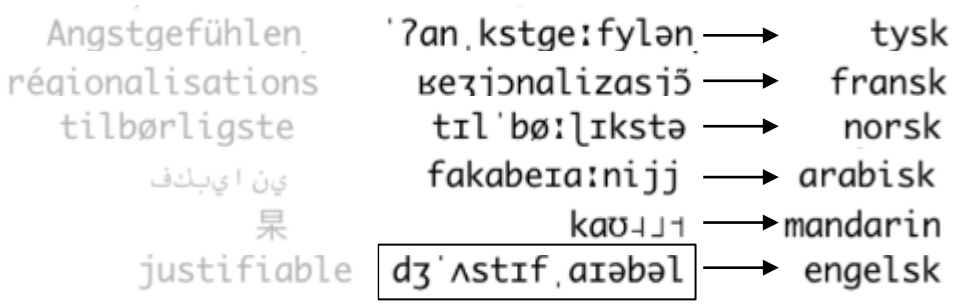
Pierre Lison
plison@nr.no

IN2110: Språkteknologiske metoder

9. mars 2022

Oblig 1b

- Hust Oblig 1b! **Innleveringsfrist**: 1. april, kl. 23:59
- To deler:
  1. Logistisk regresjon
  2. Sekvensmodeller



= Gitt den *fonetiske transkripsjonen* av et ord, modellen skal predikere *hvilket språk* ordet hører til

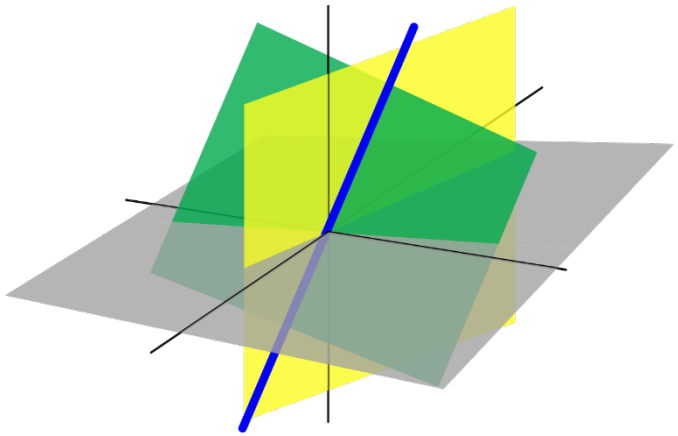
"Den formelle eieren av krokodillene er René Hedegaard i Krokodillezoo i Danmark"

↓

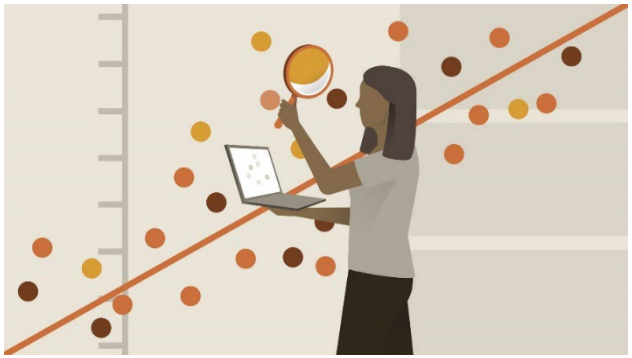
"Den formelle eieren av krokodillene er <PER>René Hedegaard</PER> i <ORG>Krokodillezoo</ORG> i <GPE>Danmark</GPE>"

= Gitt en tekst, modellen skal finne ut de *navngitte entitetene*

I dag skal vi snakke om 2 temaer:

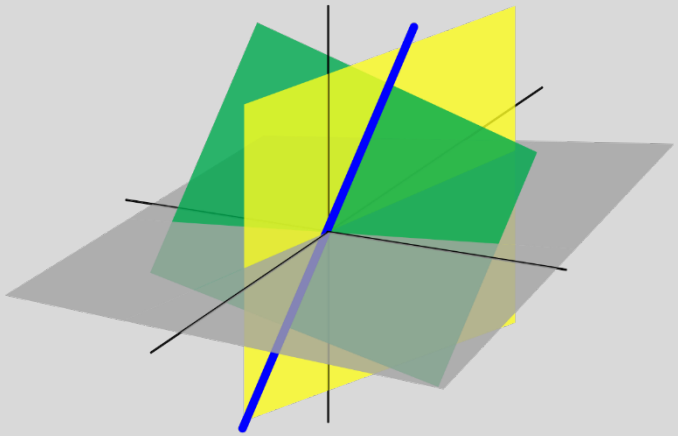


Lynkurs i lineær algebra og sannsynlighetsteori

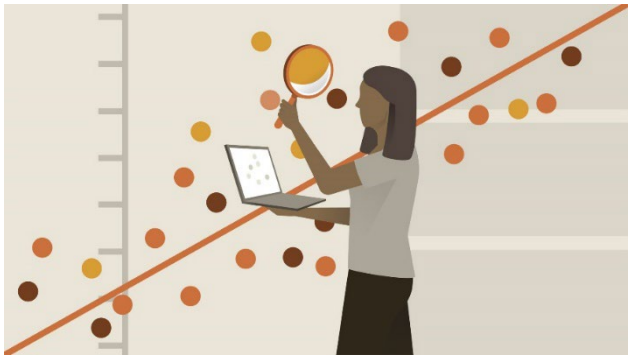


Logistisk regresjon (del 2): optimering, regularisering og forklarbarhet

I dag skal vi snakke om 2 temaer:



Lynkurs i lineær algebra og sannsynlighetsteori



Logistisk regresjon (del 2): optimering, regularisering og forklarbarhet

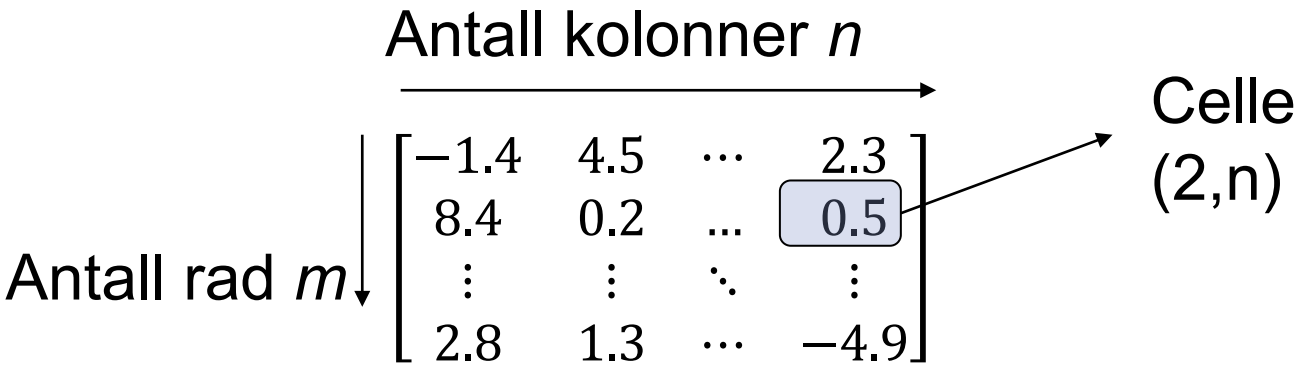
Lynkurs i lineær algebra

"Unfortunately, no one can be told what the Matrix is. You have to see it for yourself."

- Morpheus, "The Matrix"

Matematiske objekter

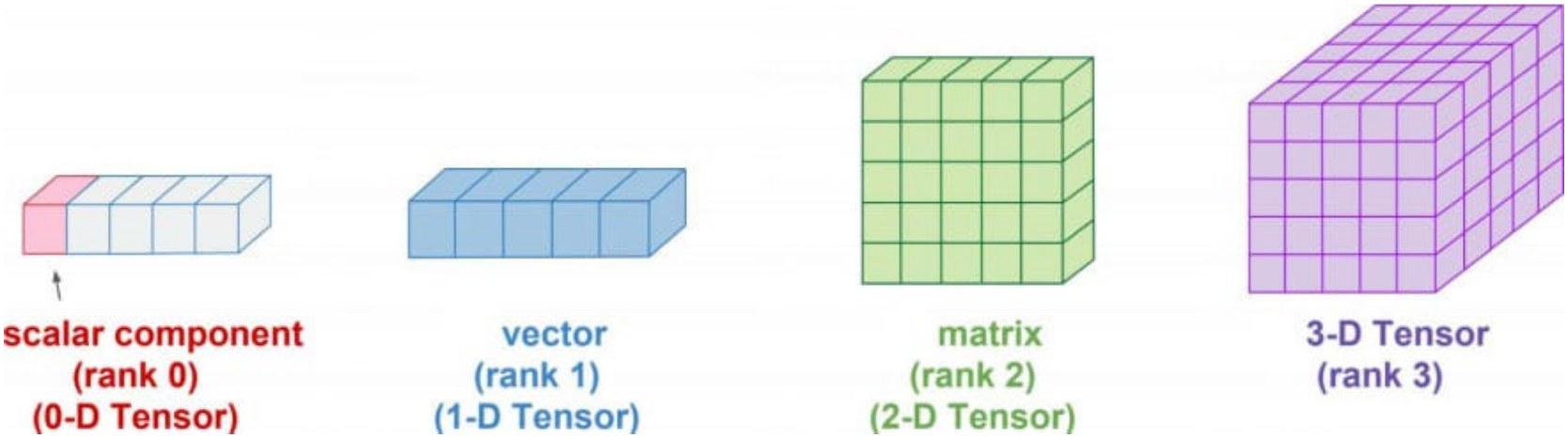
- **Skalar**: et enkelt tall, f.eks. -3.56
- **Vektor**: en rekke tall (av lengde *m*), f.eks. $\begin{bmatrix} -3.4 \\ 6.2 \\ \dots \\ 0.3 \end{bmatrix}$ Lengde *m*
- **Matrise**: en tabell (med *m* rad og *n* kolonner):



Matematiske objekter

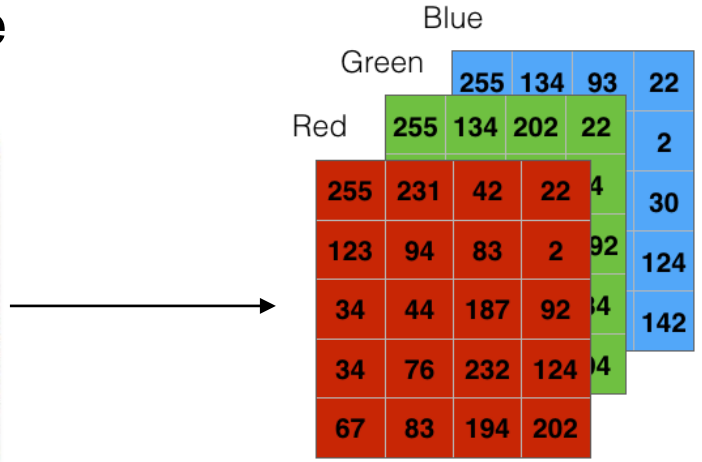
Man kan generalisere disse til **tensorer**:

- En skalar er en tensor med 0 dimensjon
- En vektor er en tensor med 1 dimensjon
- En matrise er en tensor med 2 dimensjoner
- Osv.



Matematiske objekter

- I maskinlæring bruker man tensorer (vektorer, matriser, osv.) for å representere både **data** og **modelparametre**
- **Eksempel**: en video (uten lyd) = en 4-dimensjonal tensor :
  - Dimensjon 1 er tid (med "sampling" på en viss frekvens, f.eks. 50 frames/sek)
  - Dimensjon 2 og 3 er X- Y-akse i bildene
  - Dimensjon 4 er RGB verdiene



Operasjoner

- Addisjon: $\begin{bmatrix} -3.4 \\ 6.2 \\ \dots \\ 0.3 \end{bmatrix} + \begin{bmatrix} -0.4 \\ -1.2 \\ \dots \\ 2.4 \end{bmatrix} = \begin{bmatrix} -3.8 \\ 5.0 \\ \dots \\ 2.7 \end{bmatrix}$
- Multiplikasjon med skalar: $2 * \begin{bmatrix} -3.4 \\ 6.2 \\ \dots \\ 0.3 \end{bmatrix} = \begin{bmatrix} -6.8 \\ 12.4 \\ \dots \\ 0.6 \end{bmatrix}$
- Elementvis produkt: $\begin{bmatrix} -2 \\ 1 \\ \dots \\ 0.3 \end{bmatrix} \circ \begin{bmatrix} -0.4 \\ -1.2 \\ \dots \\ 2.4 \end{bmatrix} = \begin{bmatrix} 0.8 \\ -1.2 \\ \dots \\ 0.72 \end{bmatrix}$

Elementvis produkt ≠ *matriseprodukt*! (mer om det i noen minutter)

Prikkprodukt

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i$$

- Regneprosedyre:
  - vi tar to vektorer (av samme lengde)

$$\mathbf{a} = \begin{bmatrix} -0.4 \\ -1.2 \\ 2.4 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} -3.4 \\ 6.2 \\ 0.3 \end{bmatrix}$$

-0.4*-3.4 = 1.36
-1.2*6.2 = 5.0
2.4*0.3 = 0.72

  - For hver posisjon *i* **ganger vi** a_i og b_i
  - Og til slutt **legger vi sammen** alle tallene:

  = (-0.4*-3.4) + (-1.2*6.2) + (2.4*0.3) = -5.36

- Prikkprodukten er nyttig for å bl.a.:
  - Sammenligne vektorer (se Erik sine forelesninger)
  - Representere vektede summer (som i logistisk regresjon)

Matriseprodukt

- Prikkprodukt er mellom vektorer, men det finnes en lignende operasjon for matriser eller tensorer
- Matriseprodukten er definert mellom to matriser A og B hvor A har dimensjon (m,n) og B er (n,p)

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Antall kolonner i A må være lik antall rader i B!

$$B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \dots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix}$$

n

Regneprosedyre for matriseprodukt

- Man tar en rad i A og en kolonne i B (altså to vektorer), og beregne **prikkprodukten** mellom de to
- Vi gjentar operasjon for alle kolonner 1,...p i B
- Og for alle radene 1,...m i A

A = $\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$

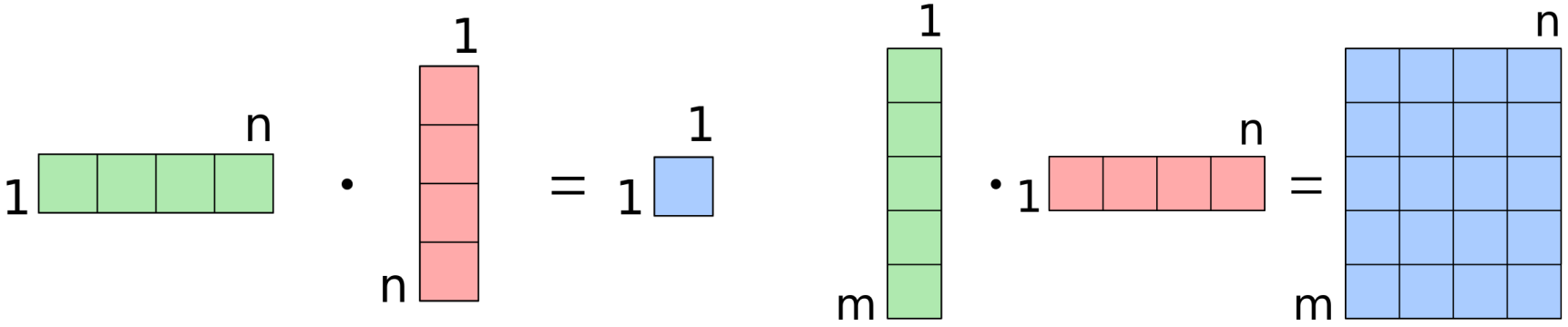
B = $\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix}$

➔ C = $\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{bmatrix}$

hvor hver $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$
(altså prikkprodukten mellom raden $a_{i\cdot}$ og kolonnen $b_{\cdot j}$)

Matriseprodukt

- Prikkprodukt er en spesielt tilfelle av matriseprodukt (med kun én dimensjon)
- Merk at matriseprodukten *ikke* er kommutativ, dvs. at rekkefølgen av matrisen er viktig: AB ≠ BA
- Retningen (vertikal eller horisontal) er også viktig:



Øvelser

- Gitt to matriser A og B:

$$A=\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \qquad B=\begin{pmatrix} 5 & -1 \\ 1 & 0 \\ -2 & 3 \end{pmatrix}$$

- Spørsmål 1: Hva er dimensjon til C = AB ?
- Spørsmål 2: Beregn C

Øvelser

Beregn verdien for:

T står for "transpose", og forvandler rader til kolonner (og vice versa)

$$\left(\begin{bmatrix}3\\2\\1\end{bmatrix}-\begin{bmatrix}2\\-2\\3\end{bmatrix}\right)^{\mathrm{T}}\begin{bmatrix}1\\0\\2\end{bmatrix}= ?$$

$$=\begin{bmatrix}1 & 4 & -2\end{bmatrix}\begin{bmatrix}1\\0\\2\end{bmatrix}=(1*1) + (4*0) + (-2*2) = -3$$

Eller med NumPy:

```
In [1]: import numpy as np

In [2]: (np.array([3,2,1]) - [2,-2,3]).dot([1,0,2])
Out[2]: -3
```

Litt lineær algebra

skjæringspunktet

Hvorfor er det nyttig?

- Vi kan skrive formelen for logistisk regresjon slik:

$$P(y=1) = \boldsymbol{\sigma}(w_1 x_1 + w_2 x_2 + \ ... + w_n x_n + b)$$

Sannsynlighet for klassen y=1

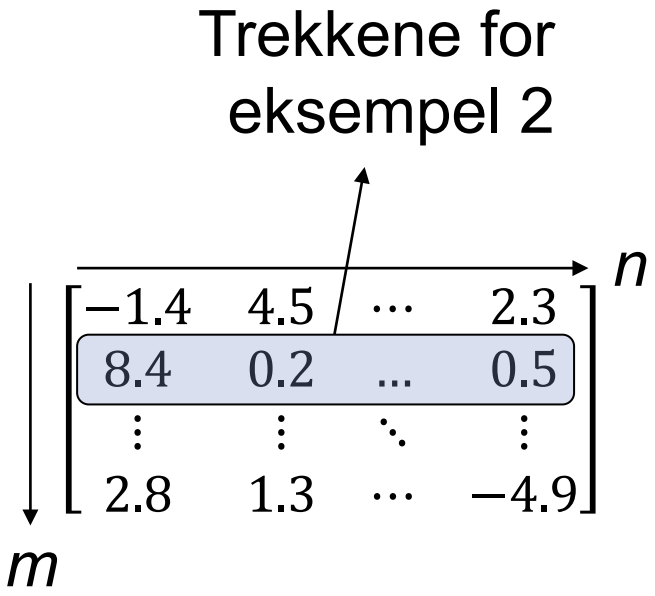
$$\boldsymbol{\sigma}(z) = \frac{1}{1+e^{-z}}$$

Input **x** med *n* (numeriske) features

- Men det er lettere slik:

$$P(y=1) = \boldsymbol{\sigma}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b) \quad \text{hvor } \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ ... \\ w_n \end{bmatrix} \text{ and } \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix}$$

Litt lineær algebra



- Hvis vi har en datasett **X** med *m* eksempler, og hvert eksempel har *n* trekk, kan vi representere den som en matrise (*m*,*n*):

- I obligen 1b skal dere trene en språkklassifikator, og treningsettet vil da representeres som et par (**X**, Y) hvor:
  - **X** er en inputmatrise (*m*,*n*), hvor *m* er antall eksempler (ord), og *n* antall (fonetiske) trekk for hvert eksempel
  - Y er en vektor av lengde *m* som fastsetter outputklassen for hvert eksempel

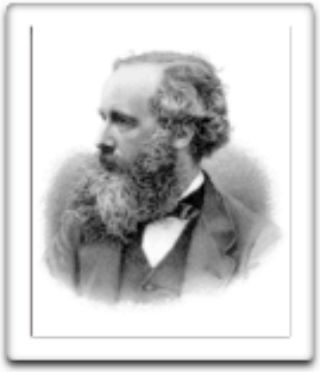
Lynkurs i sannsynlighetsteori

- Hvorfor bruker vi sannsynligheter?
- Fordi de gjør det mulig å **utrykke usikkerheter** og **resonnere seg fram** basert på usikker kunnskap
- Og usikkerhet er overalt i språkteknologi (og AI generelt)!

Pierre-Simon de Laplace
(1749-1827)

«Probability is nothing but common sense reduced to calculation»

«The true logic for this world is the calculus of probabilities»



James Clerk Maxwell
(1831-1879)

Sannsynligheter

- En *sannsynlighetsmassefunksjon P* (probability mass function, eller PMF) gir oss sannsynligheten knyttet til en bestemt verdi eller hendelse:
  - P(terning=4) = 1/6
  - P(ord="kvantemekanikk") = 1.2E-12
- Aksiomer:
  - $0 \leq P(X = x) \leq 1 \quad \forall \text{ mulige verdier } x \in X$
  - $\sum_{x \in X} P(X = x) = 1$

Dette her en veldig uformell innføring! En grundigere gjennomgang burde starte med **utfallrom** og hvordan **tilfeldige variabler** kan defineres basert på disse

Felles og betingede sannsynligheter

- Vi må ofte jobbe med flere variabler samtidig, som for eksempel P(ord_{t-1}="en", ord_t="maler)
  - Det kalles en **felles sannsynlighet** (joint probability), og skrives P(X, Y)
  - Hvis de to hendelser er uavhengig, P(X,Y) = P(X) P(Y)
- Et annet begrep er den **betingede sannsynligheten** P(*X* | Y) for X gitt en annen informasjon *Y* ("evidensen")
- Felles og betingede sannsynligheter er tett koblet:
  $P(x, y) = P(x \mid y)\, P(y)$, eller omvendt: $\mathrm{P(x \mid y)} = \frac{P(x,y)}{P(y)}$

Marginalisering

- La oss si at vi har felles sannsynligheter P(X,Y), og vi ønsker å beregne P(X). Hvordan gjør vi det?
- Vi kan bruke **marginalisering** (også kalt "summing out"):

$$P(X) = \sum_{y \in Y} P(X, Y = y)$$

- **Eksempel**: Vi har bigramsannsynligheter P(ord_{t-1}, ord_t) og vi ønsker unigramsannsynligheter P(ord_t):

$$P(ord_t = x) = \sum_y P(ord_{t-1} = y, ord_t = x)$$

Øvelser

Sannsynligheten for regnvær i dag er 55% hvis man er i Bergen, og 30% hvis man er et annet sted i Norge

Hva er sansynnlighet for at det skal regne, gitt at det er 15% sjanse for at jeg befinner meg i Bergen i dag?

Bruk formler for å komme til svaret

Husk:

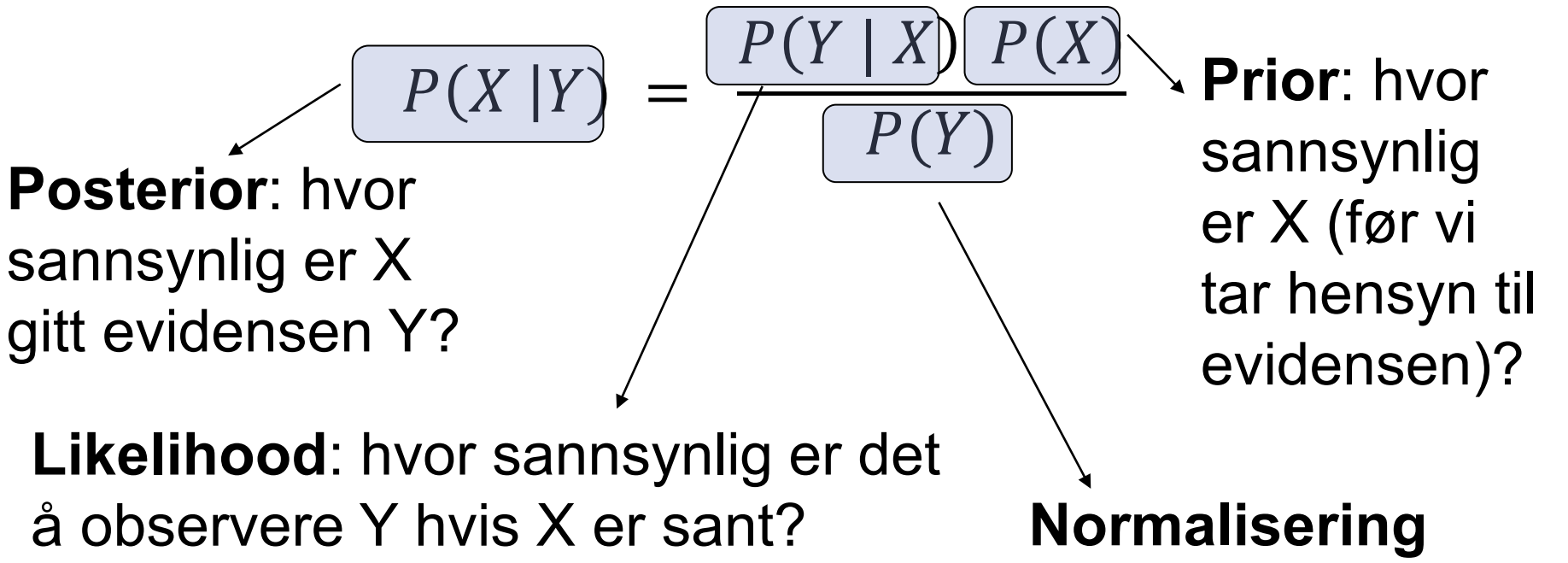
$$P(X,Y) = P(X\,|Y)\,P(Y)$$

Og:

$$P(X) = \sum_{y \in Y} P(X, Y = y)$$

Bayes' setningen

En siste operasjon som er sentralt i statistikk er Bayes' setning, som gjør det mulig å "snu" en betinget sannsynlighet:



Øvelser

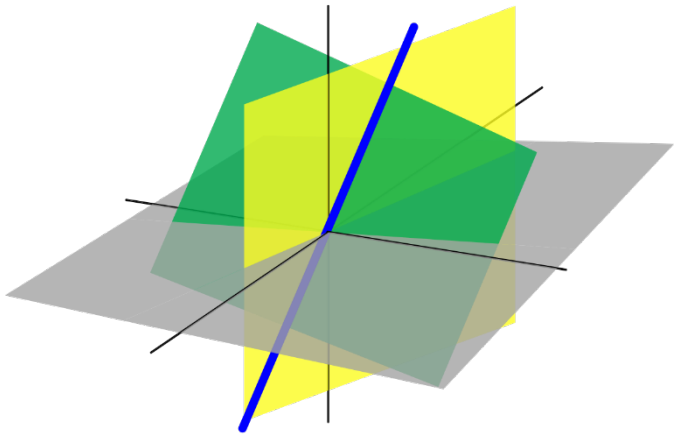
Husk Bayes: $P(X \mid Y) = \frac{P(Y \mid X)\ P(X)}{P(Y)}$

Vi ønsker å vite om en medisin fungerer. Vi gjennomfører en klinisk prøve, som gir oss gode resultater i forhold til en plasebo.

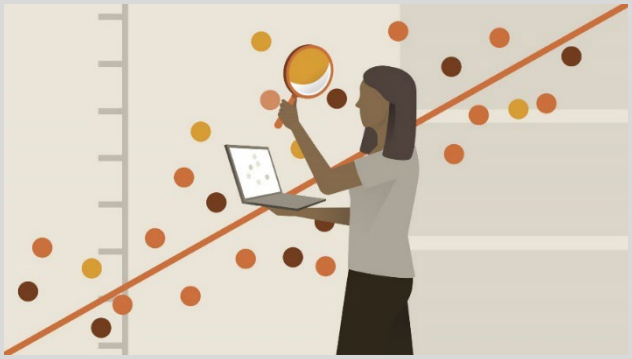
Hva er sannsynlighet for at vi har funnet en medisin som fungerer, gitt at:

- I utgangspunktet har en nylig utviklet medisin en 5% sannsynlighet til å fungere
- En medisin som faktisk fungerer har en 90% sannsynlighet til å gi gode resultater i en klinisk prøve
- En medisin som ikke fungerer har en 5% sannsynlighet til å gi gode resultater i en klinisk prøve

I dag skal vi snakke om 2 temaer:



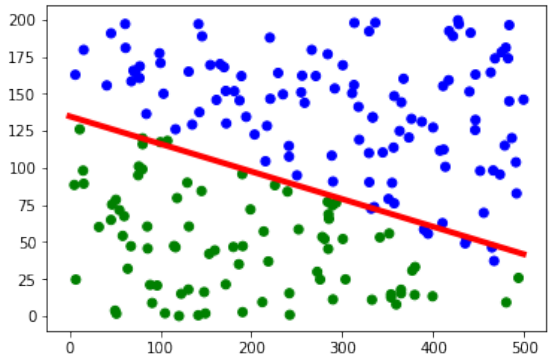
Lynkurs i lineær algebra og sannsynlighetsteori



Logistisk regresjon (del 2): optimering, regularisering og forklarbarhet

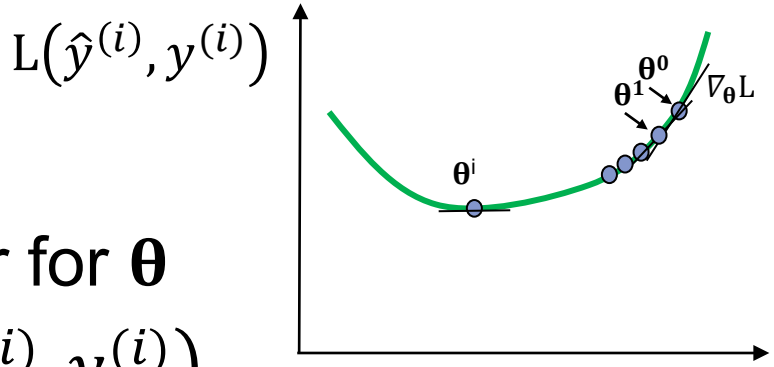
Trening

$$y = \boldsymbol{\sigma}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b)$$



- Hvordan velger man vektene **w** og skjæringspunktet *b*?
- *Intuisjon*: vi ønsker å finne verdiene for **w** og *b* som minimere "feiler" på treningsett, basert på:
  1. En **tapsfunksjon** $\mathrm{L}(\hat{y}, y)$ som angir hvor mye "feil" vi gjør hvis modellen gir sansynnlighet $\hat{y}$ som output mens den "ekte" verdi er y
  2. En **optimeringsalgoritme** som søker på verdiene for **w** og *b* som minimerer tapsfunksjonen på hele treningsett

Gradient descent



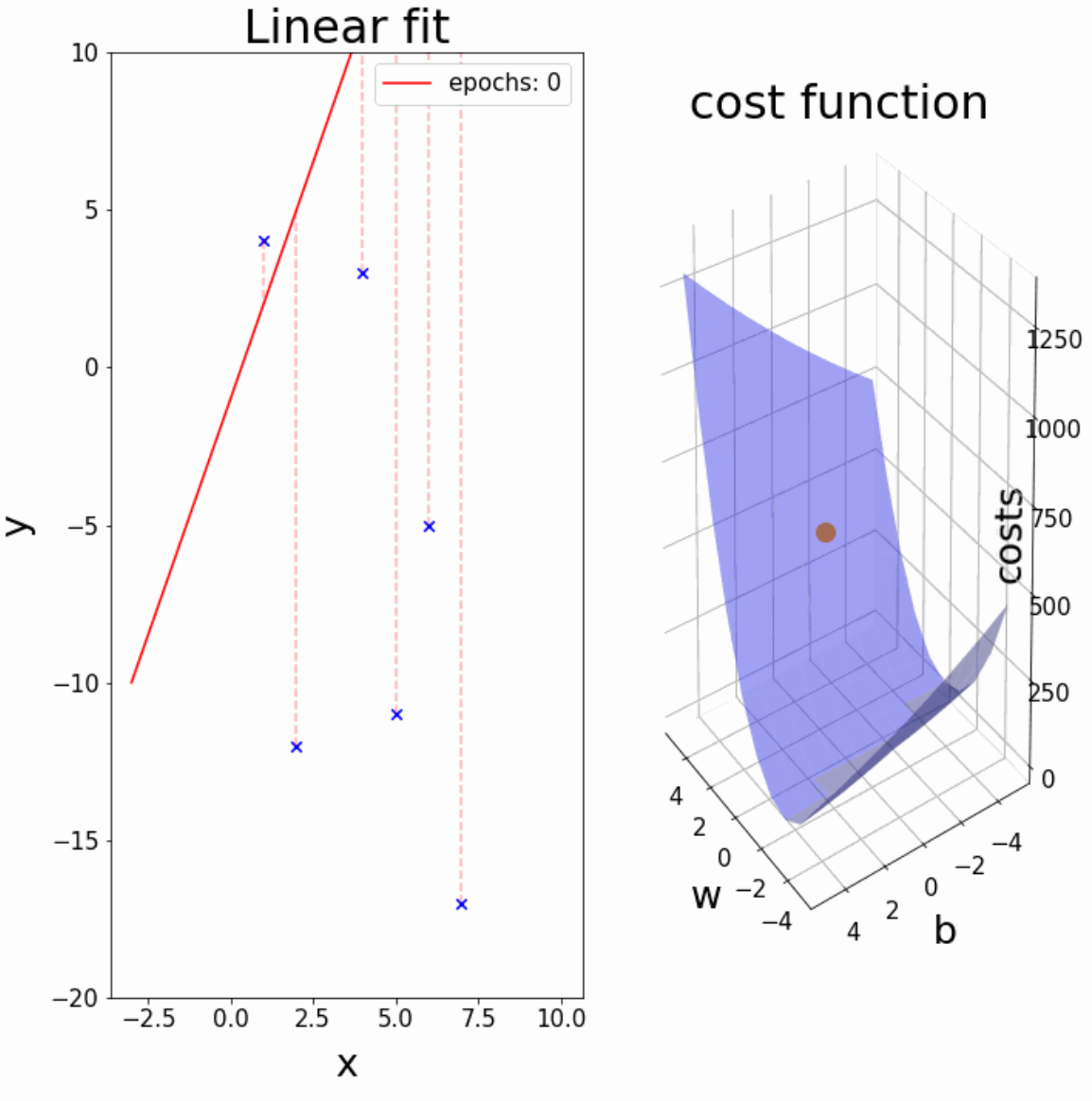
1. Vi starter med tilferdige verdier for $\boldsymbol{\theta}$
2. Vi tar en treningseksempel $\left(x^{(i)}, y^{(i)}\right)$
3. Vi beregner gradienten $\nabla_{\boldsymbol{\theta}} L\left(\hat{y}^{(i)}, y^{(i)}\right)$
4. Vi endrer deretter $\boldsymbol{\theta}$ verdiene med en liten inkrement i den motsatte retningen av gradienten:

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} - \eta \, \nabla_{\boldsymbol{\theta}} L\left(\hat{y}^{(i)}, y^{(i)}\right)$$

5. Vi gjentar for alle treningseksemplene, ofte flere ganger (epochs) inntil et minimum er nådd

η er *læringsraten*, og bestemmer størrelsen på "trinnene" ved hver endring

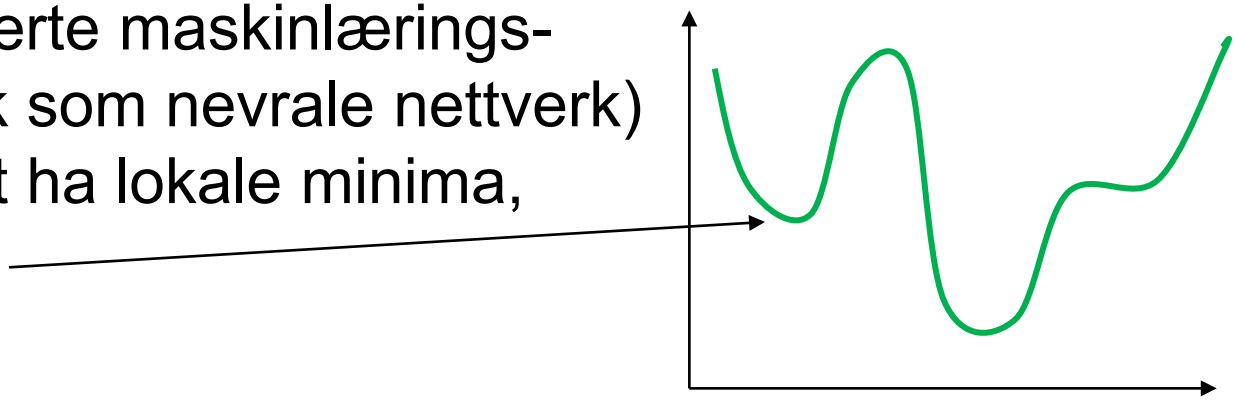
Eksempel (animert)



Vi justerer parameterne (her w og b) stegvis med *gradient descent*, slik at vi minimerer tapet på treningsett.

Gradient descent

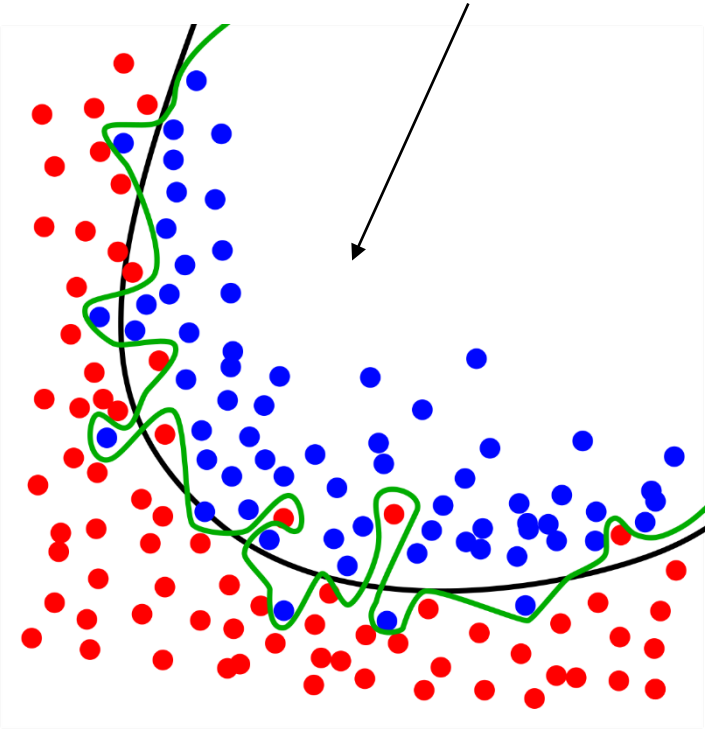
- I logistisk regresjon med vanlige tapsfunksjoner som cross-entropy vet vi at det finnes kun én global minimum
  - Matematisk sett sier vi at funksjonen er **konveks**
- I mer kompliserte maskinlærings--modeller (slik som nevrale nettverk) kan vi derimot ha lokale minima, slik som her:



- Spørsmål (på Mentimeter): hvorfor er lokale minima et problem for Gradient Descent?

Overtrening?

- Overtrening ("overfitting") er en av de største farene når man trene en maskinlæringsmodell
  - Man kan ha "perfekt" klassifisering på treningsett, men elendig ytelse på nye, ukjente datapunkter!

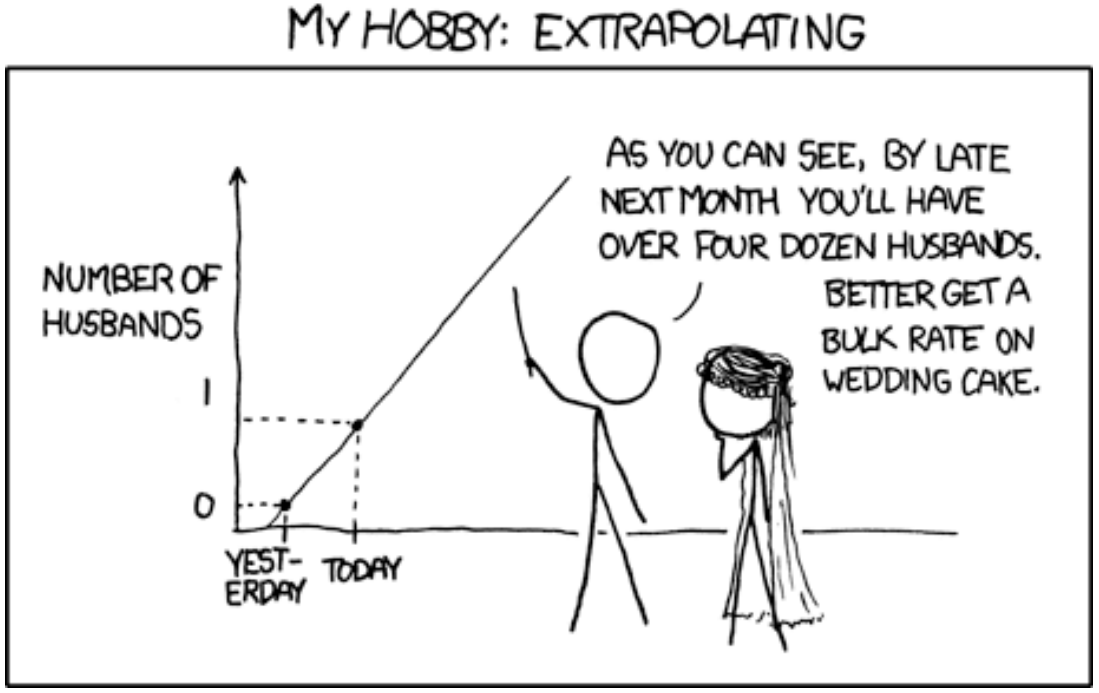


- Logistisk regresjon er relativ robust, men kan allikevel overtrenes når antall features er veldig høy
- Man kan redusere faren for overtrening ved å bruke *regularisering*

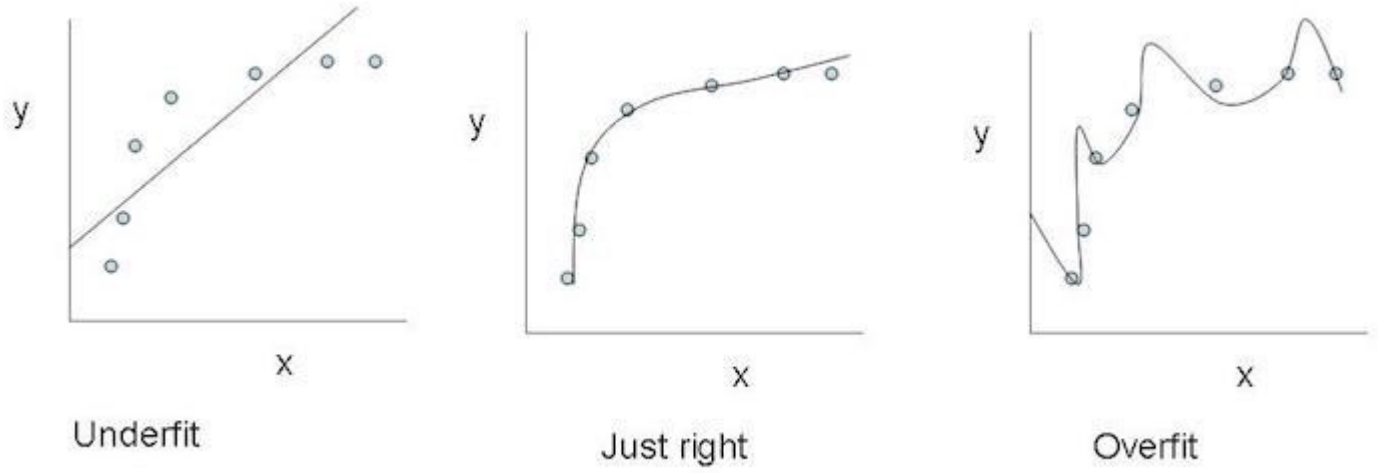
Overtrening?

Overtrening er særlig problematisk når man jobber med små datamengder

- Modellen må ekstrapolere ut fra få observasjoner



Overtrening?



Regularisering

Det finnes mange regulariseringsmetoder, men de fungerer ofte ved å "straffe" modeller som gir for mye vekt (positiv eller negativ) til enkelte features:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \mathrm{L}\left(\hat{y}^{(i)}, y^{(i)}; \boldsymbol{\theta}\right) + \alpha \sum_{j}^{n} \theta_j^2$$

Regulariseringsstyrke

Vi "straffer" modeller med høye parameterverdier

Metoden ovenfor er kalt L2-regularisering

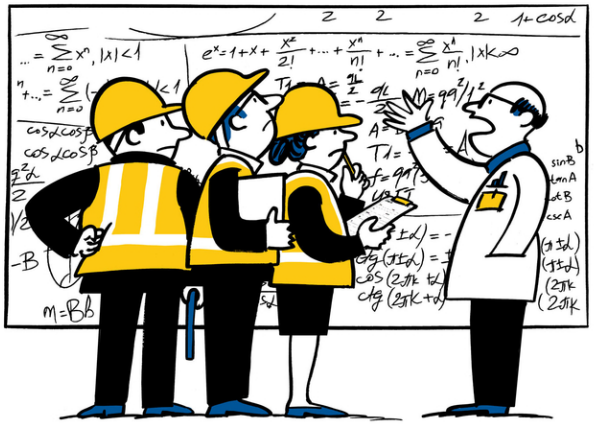
Regularisering

Ved å ta i bruk regularisering kan vi ta hensyn til *andre faktorer* enn bare å minimere tapsfunksjonen på treningsdata

- For eksempel å foretrekke enklere modeller
- Eller modeller som ikke stoler for mye på individuelle trekk (= hvis de gir høye parameterverdier til noen trekk)
- Eller modeller som er "sparse" og er i stand til å ignorere trekk som er lite relevante for klassifiseringen

→ Regularisering er et viktig "designvalg" i trening av maskinlæringsmodeller!

Forklarbarhet



- Tilbake til hovedformelen:

$$y = \boldsymbol{\sigma}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b)$$

- La oss anta vi har trent modellen på et dataset (=estimert gode verdier for $\mathbf{w}$ og b)
- Vi ønsker å **forstå** hva modellen har "lært"
- Stadig viktigere å utvikle "transparente" AI-modeller
  - GDPR: retten til forklaring ved automatiserte avgjørelser
  - Avdekke mulige skjevheter (*bias*) i modellen

Forklarbarhet

Eksempel: forskere utviklet en maskinlæringsmodell som skiller mellom ulv og husky:

Den fungerte bra på treningsett, men de oppdaget noen rare feilklassifiseringer

(a) Husky classified as wolf (b) Explanation

Når de prøvde å "tolke" modellen oppdaget de at modellen ignorerte dyret fullstendig og fokuserte kun på forekomst av snø I bakgrunnen!

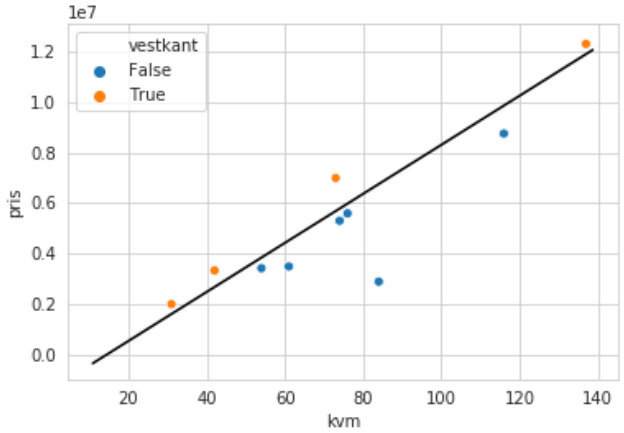
[Ribeiro et al. (2016). " Why should i trust you?" Explaining the predictions of any classifier. *In Proceedings of SIGKDD*.]

Forklarbarhet

$$y = \boldsymbol{\sigma}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b)$$

- Logistisk regresjon er heldigvis (relativt) lett å tolke
- For hver feature $x_i \in \mathbf{x}$ og tilsvarende vekt w_i:
  - Hvis w_i er positiv → *positiv* effect av x_i på y (dvs. at P(y=1|x) øker i samsvar med x_i)
  - Hvis w_i er negativ → *negativ* effekt av x_i på y (dvs. at P(y=1|x) minsker i samsvar med x_i)
  - Hvis w_i er null → ingen effect av x_i (kan forkastes)
- Jo *større* verdien (+ eller -), jo *større* effekten

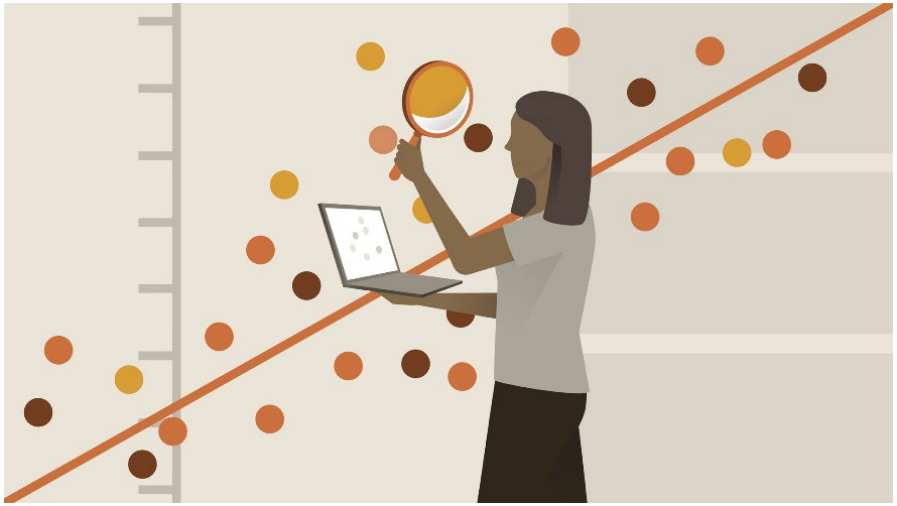
Forklarbarhet



- Ta igjen vår logistisk regresjonsmode som predikerte om en bolig var på østkant eller vestkant ut fra antall kvm og pris
- Utfra vår treningsdata (og etter omskalering) finner vi at w_{kvm}=-0.79, w_{pris}=0.83 og b=-0.49
- **Spørsmål**: i følge modellen, hvordan påvirker antall kvm og pris sannsynlighet til å holde på østkant eller vestkant?

Vi må omskalere de to variablene siden de har veldig forskjellige verdiområde (30-150 for kvm, 2.5-13 M for pris)
Viktig tips: det er alltid en god idé å omskalere variabler
(i scikit-learn: `StandardScaler` eller `QuantileTransformer`)

Oppsummering



- **Logistisk regresjon**: en lineær, probabilistisk klassifikasjonsmodell
  - Binær (med sigmoid) eller multi-klasse (med softmax)
- Hører til de viktiste verktøyene i språkteknologi
  - Pålitelig, rask å trene, og forklarbar
  - Men begrenset til lineære kombinasjoner – kan ikke lære komplekse interaksjoner mellom features

Midtveiseevaluering

- Fint om dere kan fylle ut dette skjemaet: https://nettskjema.no/a/254616

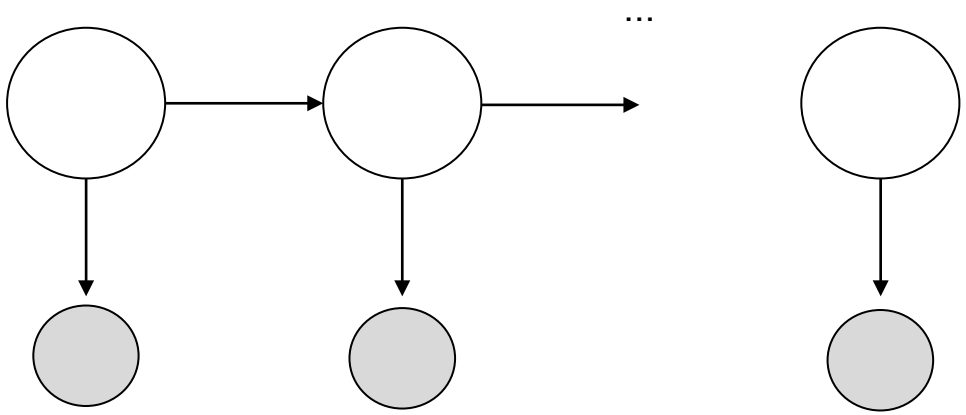
- **Frist**: onsdag neste uke

Vi søker deltakere til et eksperiment!

- **Mål**: dere blir tildelt ulike korte oppgaver (ofte spørsmål som skal besvares) og må løse disse ved å snakke med en resepsjonist-robot
- **Belønning**: gavekort på 350 kr for å delta (circa én time)
- **Interessert?** Ta kontakt med Nick Walker, walker@nr.no

Eksperimentet er en del av et forskningsprosjekt i human-robot interaction (GraphDial)

Neste uke



Modellering av sekvenser:
Markovkjeder og Hidden Markov Models