Norsk Regnesentral
NORWEGIAN COMPUTING CENTER

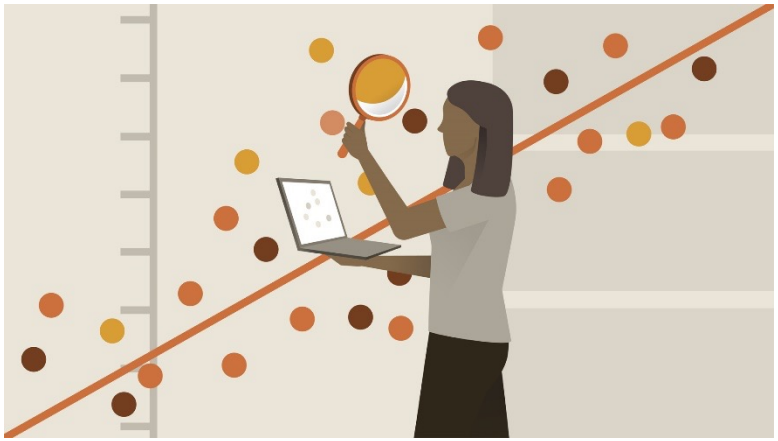
Logistisk regresjon

Pierre Lison
plison@nr.no

IN2110: Språkteknologiske metoder

2. Mars 2022

I dag skal vi snakke om:



Logistisk regresjon

- Trolig den viktigste statistiske modellen i språkteknologi
  - Og i mange andre fagfelt!
- Også en svært vanlig «brikke» i mer avanserte maskinlæringsmodeller (bl.a. nevrale nettverk)

Plan

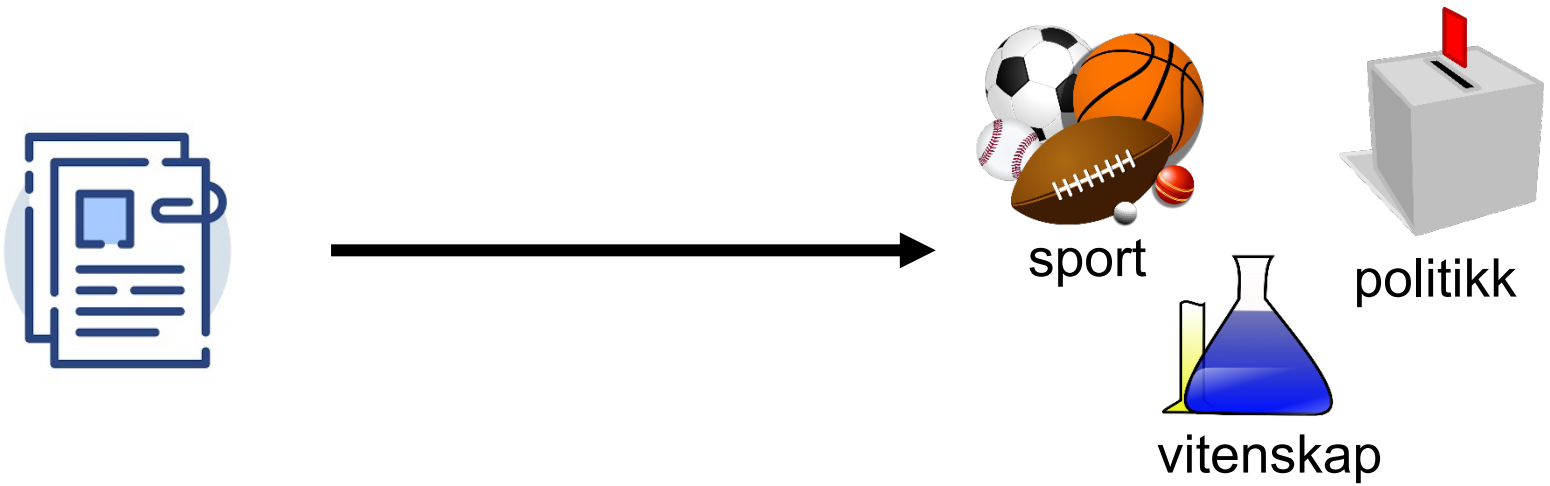
- Del 1: hva er logistisk regresjon?
  - Modell
  - Fordeler og ulemper
  - Mer enn to klasser?
- Del 2: hvordan lærer man en logistisk regresjonsmodell ut fra data?
  - Trening
  - + mer om det neste uke!

Plan

- **Del 1: hva er logistisk regresjon?**
  - **Modell**
  - **Fordeler og ulemper**
  - **Mer enn to klasser?**
- Del 2: hvordan lærer man en logistisk regresjonsmodell ut fra data?
  - Trening
  - + mer om det neste uke!

Logistisk regresjon

= en klassifikasjonsmodell



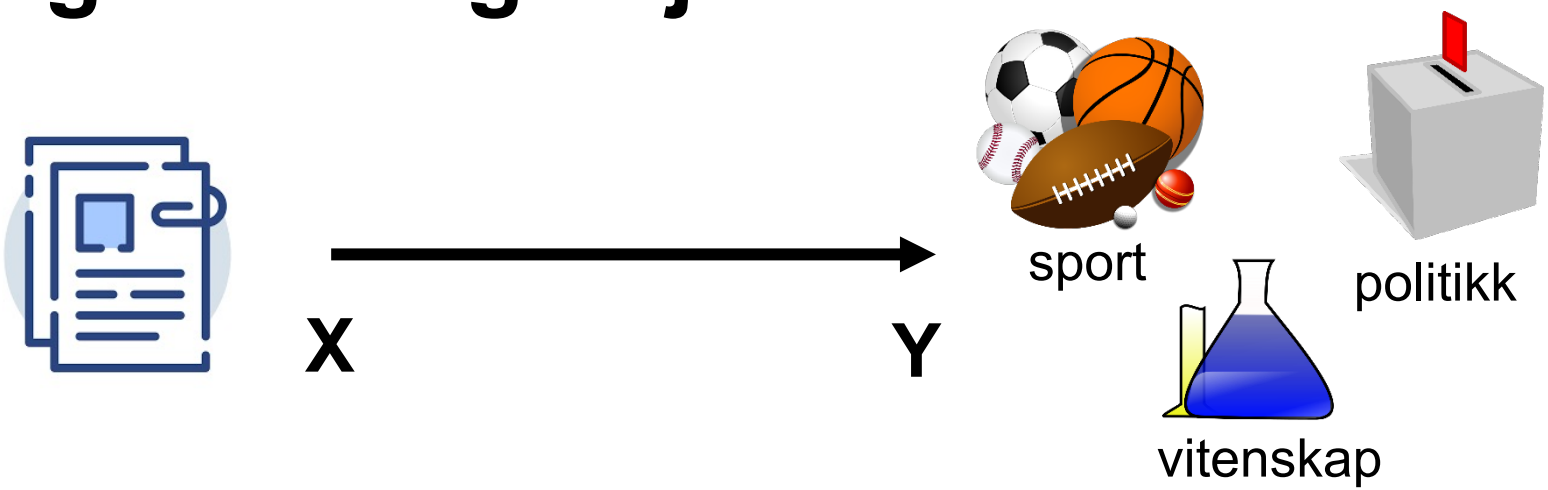
Input **X**

(f.eks. et dokument)

Output **Y**

(f.eks. en kategori)

Logistisk regresjon



En logistisk regresjonsmodell gir oss sansynnlighet P(*y*|*x*) for en outputklasse *y* gitt input *x*

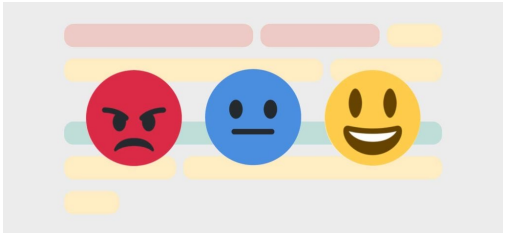
Representert som (numeriske) *features*

Outputklassen (valgt ut fra et sett mulige klasser definert på forhånd)

Nevn noen eksempler av språkteknologiske oppgaver eller anvendelser basert på klassifisering:

(svar på Mentimeter)

Eksempler



Sentimentanalyse

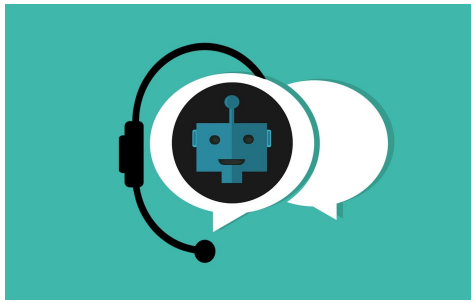
(=identifikasjon av *subjektive meninger*)
Input: tekst (ofte setninger eller korte tekster)
Output: binær (+/-), multi-class eller multi-label

Hver input må være i én klasse

Hver input kan være i et vilkårlig antall klasser (fra 0 til K)

Spamfiltrering

Input: tekst fra epost + andre trekk (senderadresse osv.)
Output: sannsynlighet for at eposten er spam



Intentgjenkjenning

Input: brukerytring i en chatbot, f.eks.
`"når åpner dere på lørdag?"`
Output: Intentklasse, f.eks.
`etterspør_åpningstid(dag=lørdag)`

Lineær regresjon

- Vi først tar en titt på "vanlig" lineær regresjon
- La oss si at vi ønsker å predikere boligpriser i Oslo basert på 2 features:
  - Antall kvm
  - Om de holder på østkanten eller vestkanten
- Vi kan estimere en *regresjonsmodell* basert på dataset av eksempler

kvm	Vestkant?	Pris
84	Nei	2 890 000
54	Nei	3 420 000
73	Ja	6 990 000
74	Nei	5 300 000
137	Ja	12 300 000
76	Nei	5 590 000
31	Ja	2 000 000
61	Nei	3 490 000
116	Nei	8 750 000
42	Ja	3 330 000

Lineær regresjon

$$y = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b$$

Output

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

skjæringspunktet (=verdi av *y* hvis alle features i *x* er lik 0)

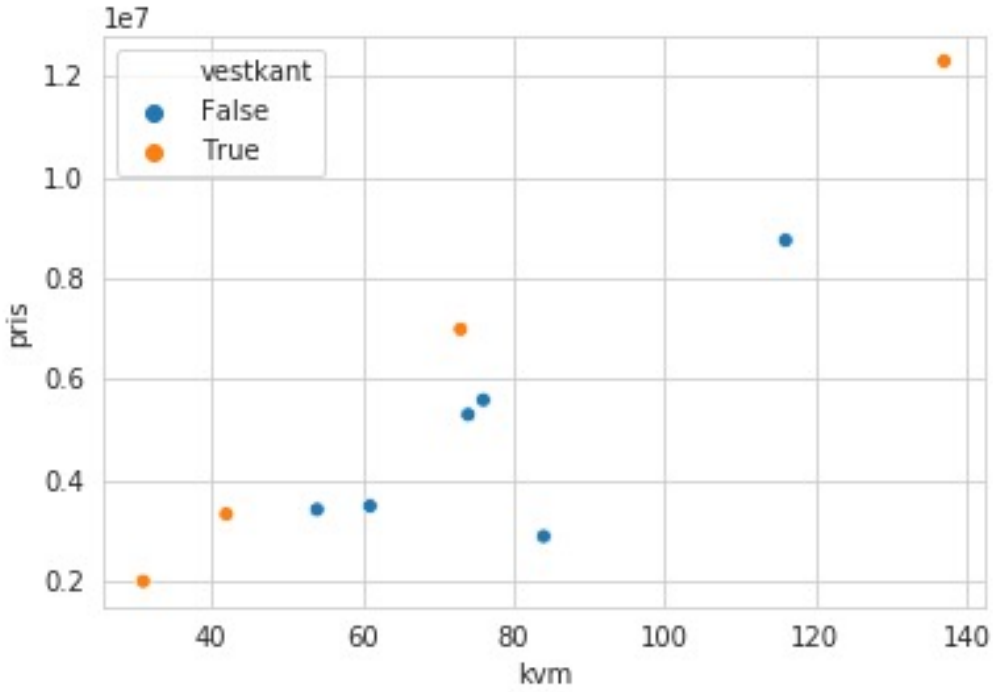
- Verdiene $w_1, w_2, \ldots w_n$ are *vektene* i modellen, og estimeres (sammen med skjæringspunktet) ut fra data
- Vi kan også skrive formelen slik:

$$\boxed{y = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b}$$

hvor $\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \ldots \\ w_n \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix}$

Lineær regresjon

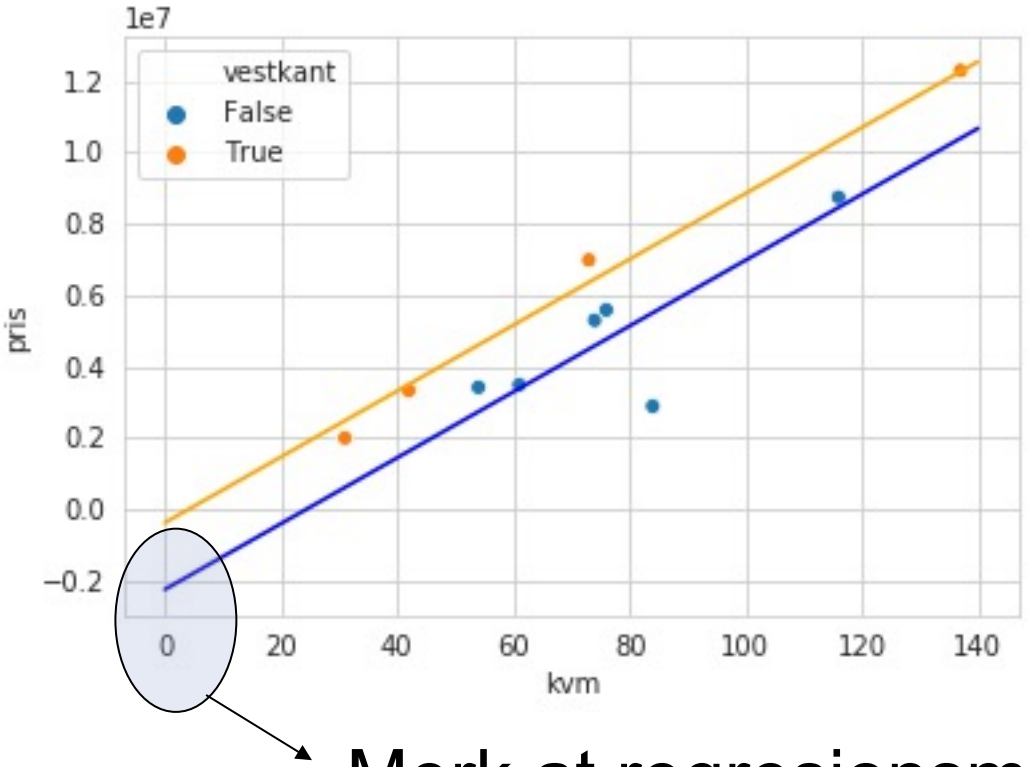
Tilbake på eksempelet vår:



kvm	Vestkant?	Pris
84	Nei	2 890 000
54	Nei	3 420 000
73	Ja	6 990 000
74	Nei	5 300 000
137	Ja	12 300 000
76	Nei	5 590 000
31	Ja	2 000 000
61	Nei	3 490 000
116	Nei	8 750 000
42	Ja	3 330 000

Lineær regresjon

Tilbake på eksempelet vår:



kvm	Vestkant?	Pris
84	Nei	2 890 000
54	Nei	3 420 000
73	Ja	6 990 000
74	Nei	5 300 000
137	Ja	12 300 000
76	Nei	5 590 000
31	Ja	2 000 000
61	Nei	3 490 000
116	Nei	8 750 000
42	Ja	3 330 000

Merk at regresjonsmodellen predikerer en negativ pris hvis bolig har en kvm < 20!

Logistisk regresjon

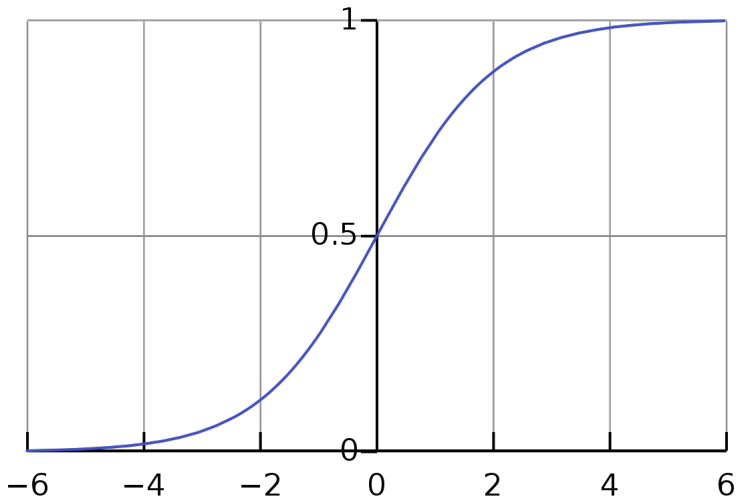
- Kan vi bruke regresjonsformelen til klassifisering?

  → Vi må predikere en *sansynnlighet*, men regresjon gir oss verdier for *y* mellom $-\infty$ og $+\infty$

- Vi kan omforme regresjonsresultaten slik at den blir mellom 0 og 1 med en lite endring:

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

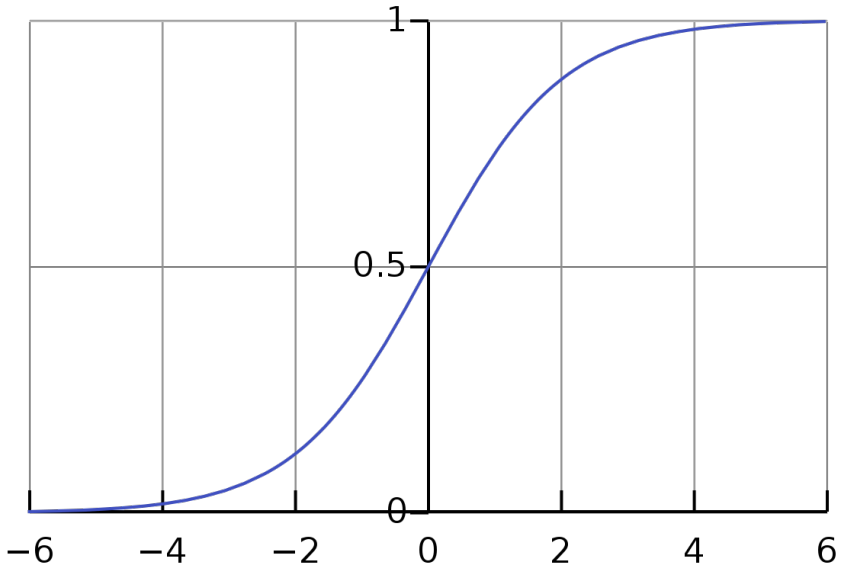
$$\text{hvor } \boldsymbol{\sigma}(z) = \frac{1}{1 + e^{-z}}$$



Logistisk regresjon

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

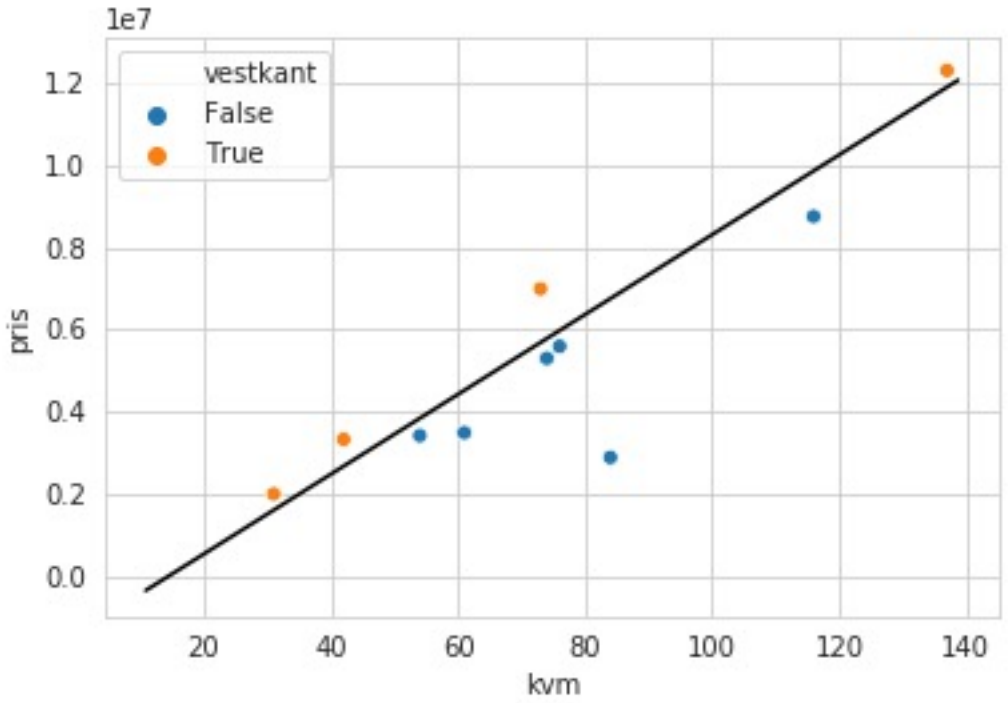
$$\text{hvor } \boldsymbol{\sigma}(z) = \frac{1}{1 + e^{-z}}$$



- Sigmoid-funksjonen $\boldsymbol{\sigma}$ brukes til å tvinge regresjonsresultaten til å være en sansynnlighet $\in [0,1]$
- Hvis $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b$ er
  - Veldig negativ (e.g. -100) → $P(y = 1|x) \approx 0$
  - 0 → $P(y = 1|x) = 0.5$
  - Veldig positiv (e.g. 100) → $P(y = 1|x) \approx 1$

Logistisk regresjon

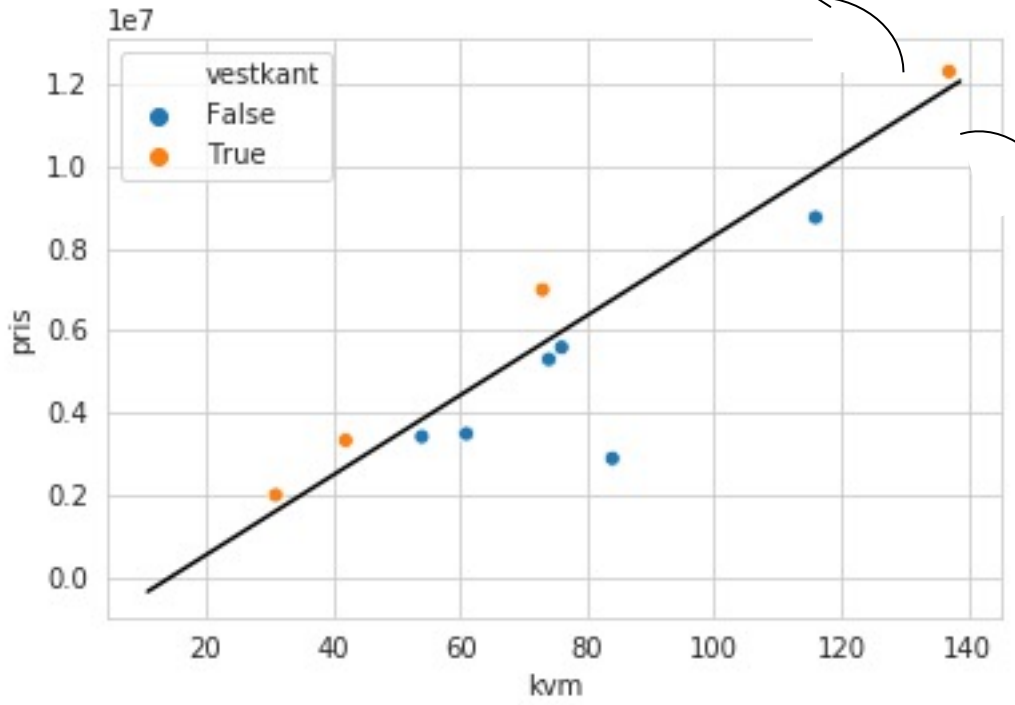
La oss ta igjen boligeksempelet vårt, men denne gangen skal vi predikere *om* boligen er på veskanten (gitt kvm og pris)



kvm	Pris	Vestkant?
84	2 890 000	Nei
54	3 420 000	Nei
73	6 990 000	Ja
74	5 300 000	Nei
137	12 300 000	Ja
76	5 590 000	Nei
31	2 000 000	Ja
61	3 490 000	Nei
116	8 750 000	Nei
42	3 330 000	Ja

Logistisk regresjon

hvis en bolig er på denne siden av linjen, predikerer modellen at den er på vestkanten



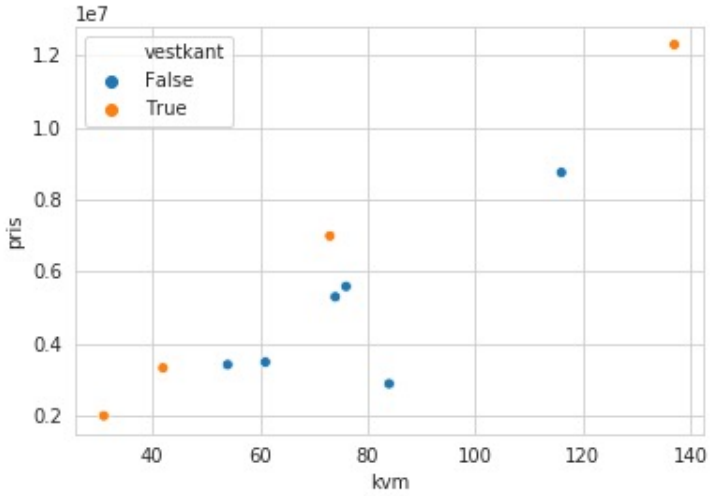
Og på denne siden predikerer modellen at den er på østkanten

NB: jo mer er punktet langt fra linjen, jo høyere er prediksjons sannsynlighet

Logistisk regresjon er en **lineær modell**

➔ Beslutningsregelen er en *vektet sum* av features fra input

Eksempel



- Vi ønsker å bruke en logistisk regresjonsmodell til å predikere om en bolig er på østkant eller vestkant basert på kvm og pris (i tkr)
- Estimert fra data at w_{kvm}=-8.15, $w_{pris(i\ tkr)}$=0.09 og b=104
- **Spørsmål**: Hvor sannsynlig er det er at en bolig på 97 kvm solgt for 7650 tkr. holder på vestkanten, i følge modellen?
  - Svar på Mentimeter!

Husk hovedformelen fra logistisk regresjon:

$$y = \boldsymbol{\sigma}(w_1 x_1 + w_2 x_2 + \ \ldots + b) \qquad \text{hvor } \boldsymbol{\sigma}(z) = \frac{1}{1 + e^{-z}}$$

Eksempel

- Vi har x_1=97 og x_2=7650
- Og w_1=-8.15, w_2=0.09, og b=104
- Så P(y=vestkant | **x**) = $\frac{1}{1+e^{-(-8.15*97+0.09*7650+104)}} = 0.87545$
- Eller i Python:

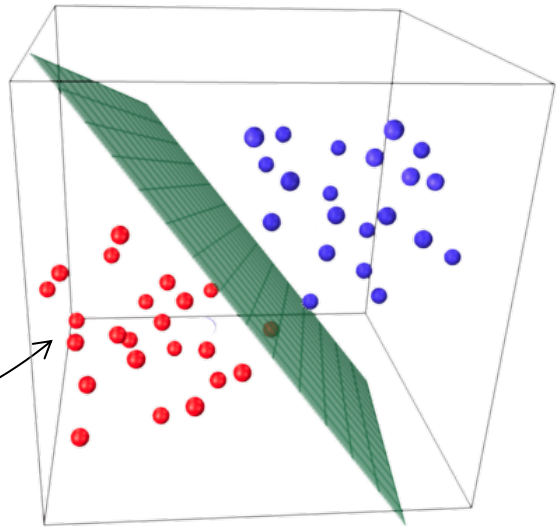
```
In [1]: import math

In [2]: 1/(1+math.exp(8.15*97 - 0.09*7650 -104))
Out[2]: 0.875446641812576
```

Logistisk regresjon

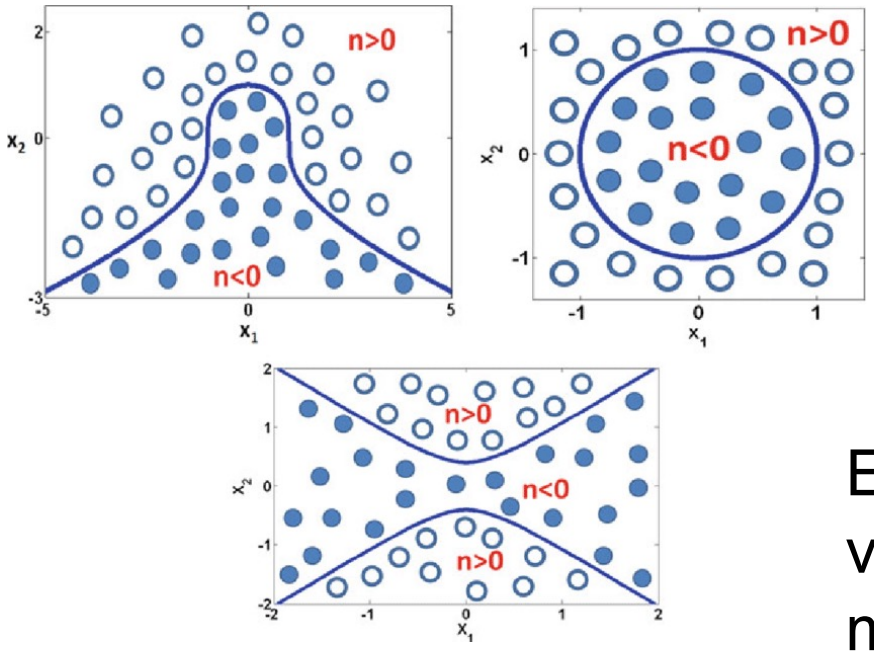
- Antall features ofte langt høyere enn 2
  - Språkteknologiske modeller kan ha tusenvis av features
  - For eksempel “bag-of-words” features som indikerer om et ord forekommer i en tekst eller ikke
- Hvis antall features > 2 er beslutningregelen en (hyper)plan:

Punkter på denne siden av planen vil være klassifisert som **rødt**



Logistisk regresjon

Noen ganger er lineære modeller (som bl.a. logistisk regresjon) et dårlig valg:



En lineær modell vil ikke være i stand til å skille mellom klassene her

Logistisk regresjon

Men lineære modeller har også ganske viktige fordeler:

1. Kan trenes på *beskjedne datamengder*

Hvorfor? Fordi lineærmodeller har (relativt) få parametre som må estimeres: vektene **w** og skjæringspunktet *b*

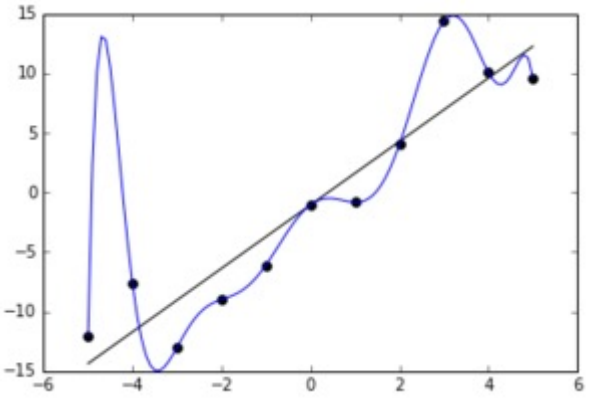
Til sammenligning kan store nevrale nettverk inneholde millioner av parametre → krever tilgang til veldig store treningsdata!

Logistisk regresjon

Men lineære modeller har også ganske viktige fordeler:

1. Kan trenes på *beskjedne datamengder*
2. Gode *generaliseringsevner*
   (= mindre risiko for overtrening)

Eksempel:



pga. deres enkelhet
generaliserer lineærmodeller
ganske bra fra treningsdata
til nye, ukjente datapunkter

Logistisk regresjon

Men lineære modeller har også ganske viktige fordeler:

1. Kan trenes på *beskjedne datamengder*
2. Gode *generaliseringsevner*
   (= mindre risiko for overtrening)
3. Rask (trening og prediksjon) og skalerbar

Kan skalere uten videre til store datasett (med millioner av datapunkter og/eller millioner av features)
… og uten å kreve tilgang til GPU-er!

Logistisk regresjon

Men lineære modeller har også ganske viktige fordeler:

1. Kan trenes på *beskjedne datamengder*
2. Gode *generaliseringsevner*
   (= mindre risiko for overtrening)
3. Rask (trening og prediksjon) og skalerbar
4. Forklarbar

   ↘

   Kan vi forstå hvilke del av input påvirker modellens prediksjoner?

   [mer om det neste uke!]

Mer enn to klasser?

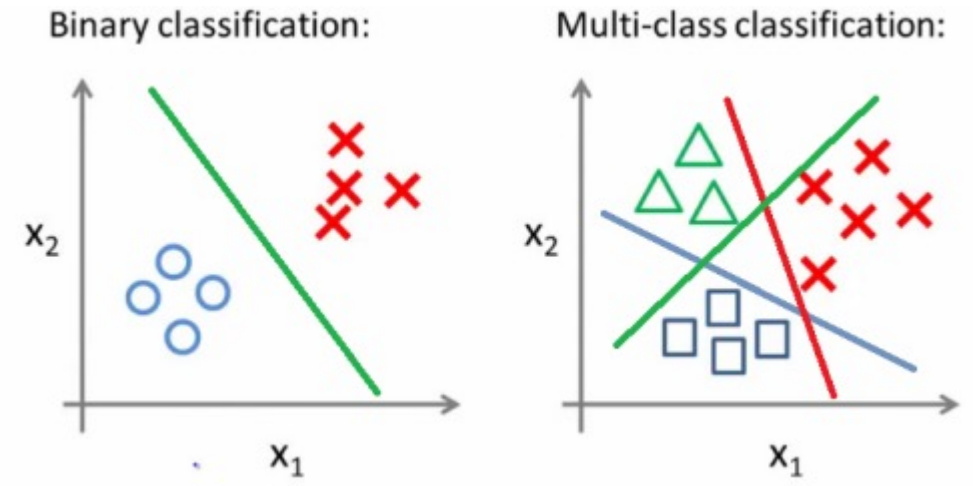
- Logistisk regresjon med sigmoid-funksjon $\boldsymbol{\sigma}$ er begrenset til 2 klasser (som f.eks. spam/ikke spam)
- For K klasser (med K > 2) bruker man *multinomial logistisk regresjon* og erstatter sigmoid med softmax:

$$\mathrm{P}(\mathrm{y} = \mathrm{i}|\mathbf{x}) = \mathrm{softmax}(\mathbf{w_i}^T\mathbf{x} + \mathrm{b_i})$$

$$\text{hvor } \mathrm{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

➔ Sansynnlighet for å tilhøre en klasse *i* er igjen basert på en vektet sum av inputpunktet **x**
… og derettert normalisert med "softmax" slik at vi får en riktig sansynnlighetsfordeling

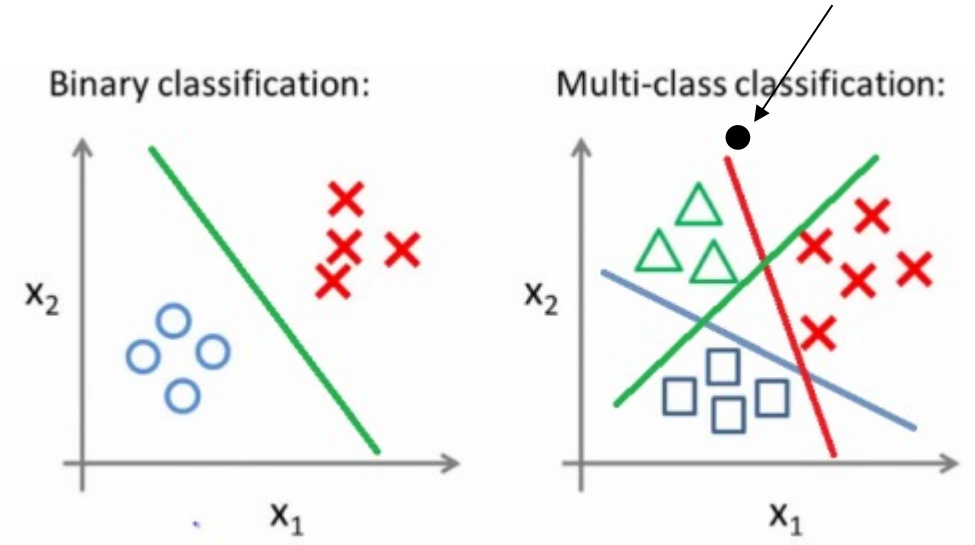
Mer enn to klasser?



I multinomial (=“multi-klasser”) logistisk regresjon har vi én beslutningsregel per klasse

Hva er den mest sansynnlige klassen for dette punktet?

(svar på Mentimeter)



I multinomial (=“multi-klasser”) logistisk regresjon har vi én beslutningsregel per klasse

Eksempel

- La oss si at vi lager en modell for tekstklassifisering basert på (multinomial) logistisk regresjon, med 10 klasser
- Modellen tar i bruk 120 features, og blir trent på et datasett med 300 000 tekster
- **Spørsmål**: Hvor mange parametre inneholder modellen?
  - Svar på Mentimeter

Logistisk regresjon med K klasser:

$$\mathrm{P}(\mathrm{y} = \mathrm{i}|\mathbf{x}) = \mathrm{softmax}(\mathbf{w_i}^T\mathbf{x} + \mathrm{b_i})$$

$$\text{hvor } \mathrm{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

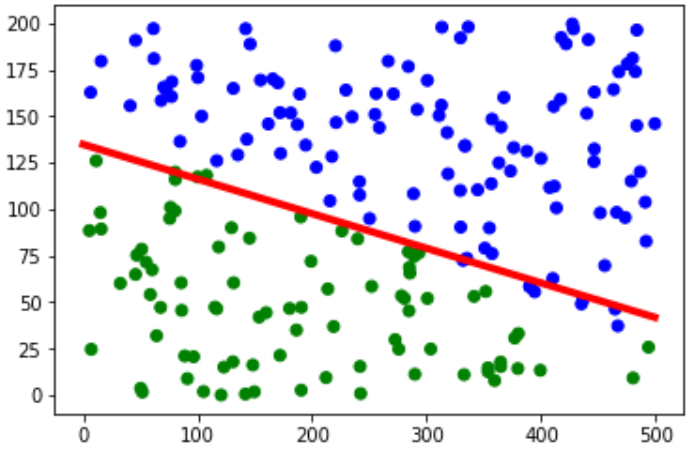
Plan

- Del 1: hva er logistisk regresjon?
  - Modell
  - Fordeler og ulemper
  - Mer enn to klasser?
- **Del 2: hvordan lærer man en logistisk regresjonsmodell ut fra data?**
  - **Trening**
  - **+ mer om det neste uke!**

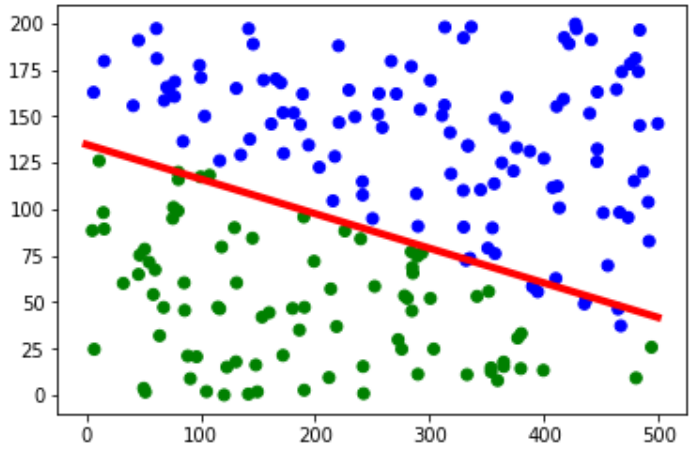
Trening

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

- Hvordan velger man vektene **w** og skjæringspunktet *b*?
- Vi bruker et treningsett for å estimere disse!
- Intuisjon: vi ønsker å finne verdiene for **w** og *b* som minimere "klassifiseringfeiler" på treningsett



Trening



Vi trenger å definere:

1. En **tapsfunksjon** $\mathrm{L}(\hat{y}, y)$ som utrykker hvor mye "feil" vi gjør hvis modellen gir sansynnlighet $\hat{y}$ som output mens den "ekte" verdi er y

2. En **optimeringsalgoritme** som skal søke på verdiene for **w** og *b* som minimerer tapsfunksjonen på hele treningsett

Tapsfunksjon

Den meste vanlige tapsfunksjon for logistisk regresjon er *cross-entropy*:

$$\mathrm{L}(\hat{y}, y) = -[y \log \hat{y} + (1 - y)\log(1 - \hat{y})]$$

↓

Eksempler:

Hvis $\hat{y}$ = 0.99 and y =1 er $\mathrm{L}(\hat{y}, y) \approx 0.01$

Hvis $\hat{y}$ = 0.01 and y =1 er $\mathrm{L}(\hat{y}, y) \approx 4.6$

Det finnes mange typer tapsfunksjoner, avhengig av hva man ønsker at modellen skal optimere

Optimering

Vi ønsker å finne modelparametrene $\boldsymbol{\theta} = (\mathbf{w}, b)$ som minimisere "tapet" på treningsett:

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

Vi samler tapsverdiene på hele treningsett

$D = \{\left(x^{(i)}, y^{(i)}\right), 1 \leq i \leq m\}$

$\hat{y}^{(i)}$ er prediksjonen for $x^{(i)}$ med de nåværende modellparametrene:

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

➔ Å finne de "beste" verdiene for $\boldsymbol{\theta}$ er et **optimeringsproblem**

Gradient descent

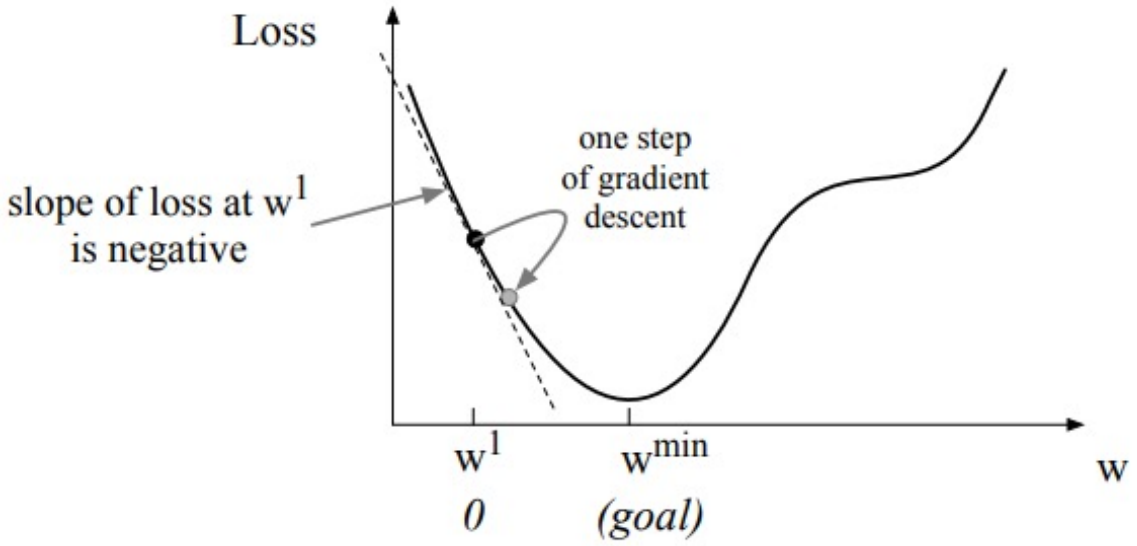
- **Gradient descent** er en populær optimeringsalgoritm

- Gradient descent fungerer slik:
  1. Vi starter med tilferdige verdier for $\boldsymbol{\theta}$
  2. Vi tar et treningseksempel $\left(x^{(i)}, y^{(i)}\right)$
  3. Vi beregner hvordan vi best kan endre $\boldsymbol{\theta}$ slik at vi reduserer tapet på eksempelet
  4. Vi endrer deretter $\boldsymbol{\theta}$ verdiene med en liten inkrement i denne retningen
  5. Vi gjentar for alle treningseksemplene

Det gjøres ved å beregne *gradienten* til tapsfunksjonen i forhold til $\boldsymbol{\theta}$ (mer om det neste uke!).

Gradient descent

Vi finner ut hvordan **θ**-verdiene skal endres med å beregne *gradienten* av tapsfunksjonen L avhengig av θ



θ har vanligvis mange dimensjoner, slik at gradienten er en *vector* $\nabla_{\boldsymbol{\theta}} \mathrm{L}(\hat{y}, y; \boldsymbol{\theta})$

Gradient descent

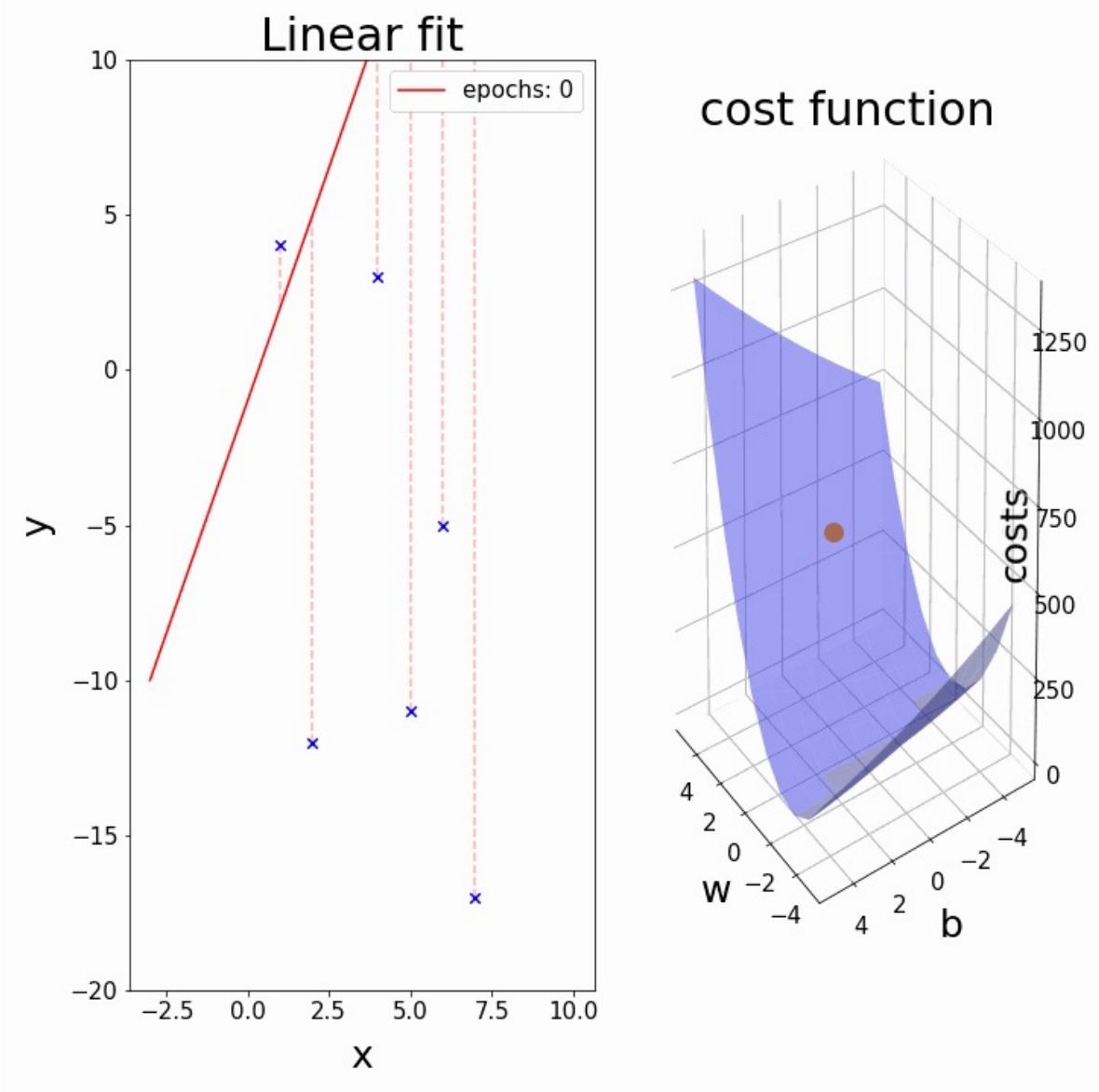
1. Vi starter med tilferdige verdier for $\boldsymbol{\theta}$
2. Vi tar en treningseksempel $\left(x^{(i)}, y^{(i)}\right)$
3. Vi beregner gradienten $\nabla_{\boldsymbol{\theta}} \mathrm{L}\left(\hat{y}^{(i)}, y^{(i)}\right)$
4. Vi endrer deretter $\boldsymbol{\theta}$ verdiene med en liten inkrement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

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

5. Vi gjentar for alle treningseksemplene

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

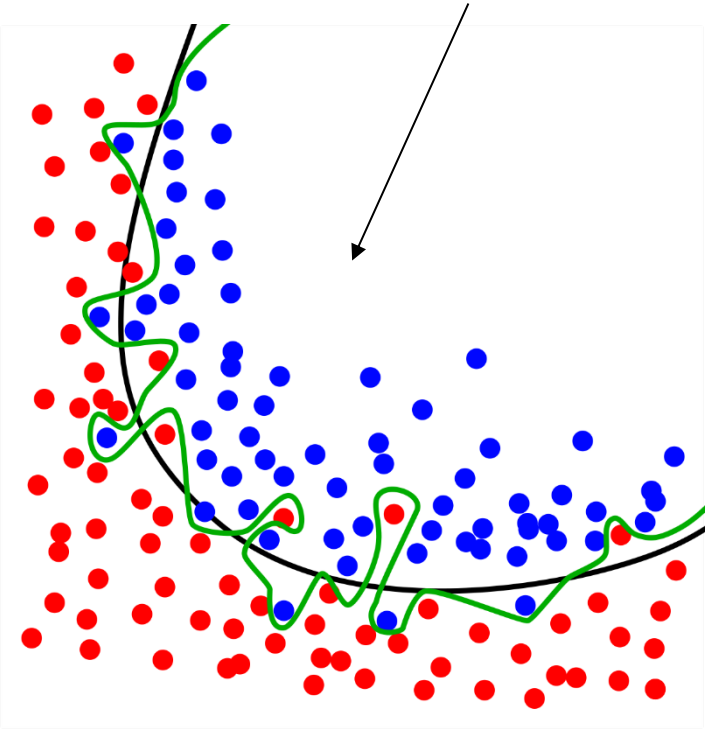
2D eksempel (animert)



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

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!

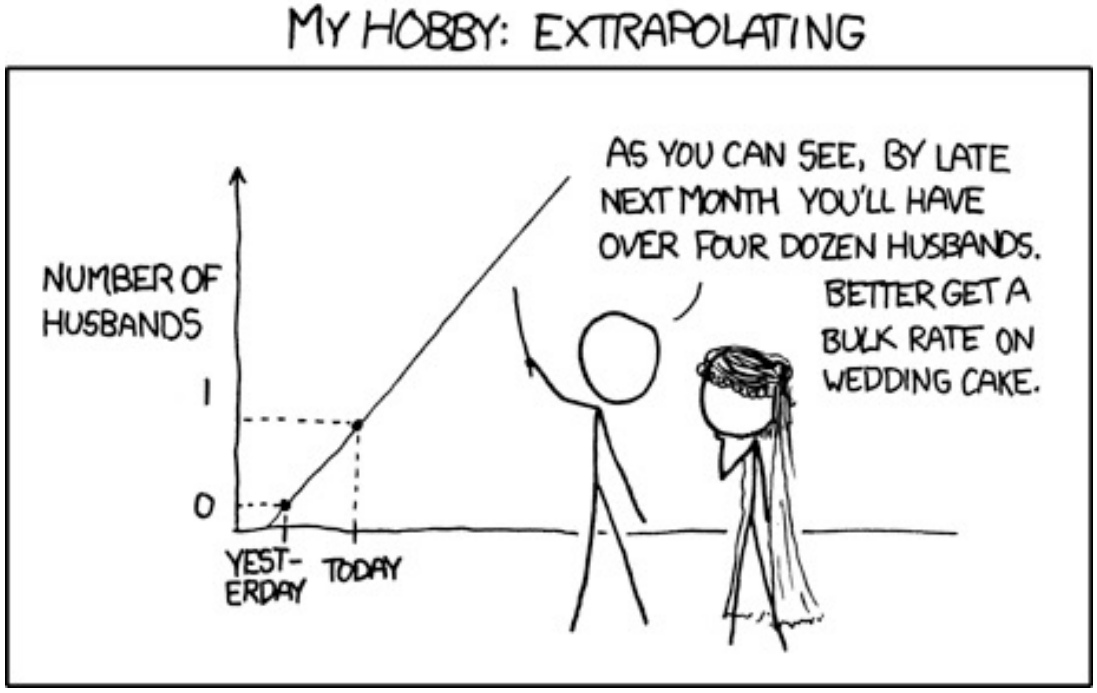


- Logistisk regresjon er relativ robust, men kan allikevel overtrenes når antall features er veldig høy
- Man kan redusere faren for overtrening ved å bruke *regularisering*

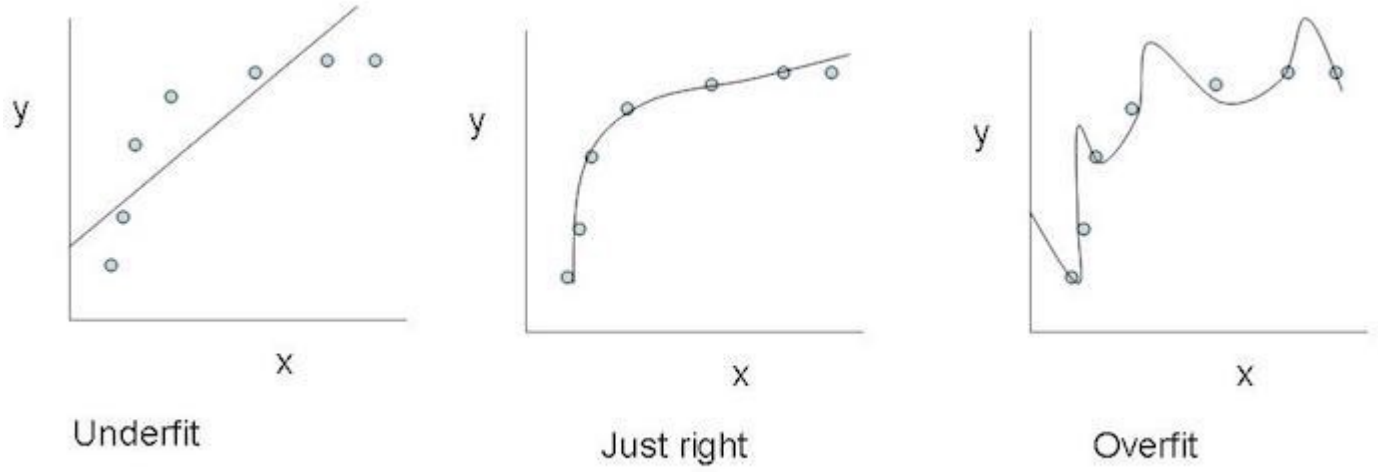
Overtrening?

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

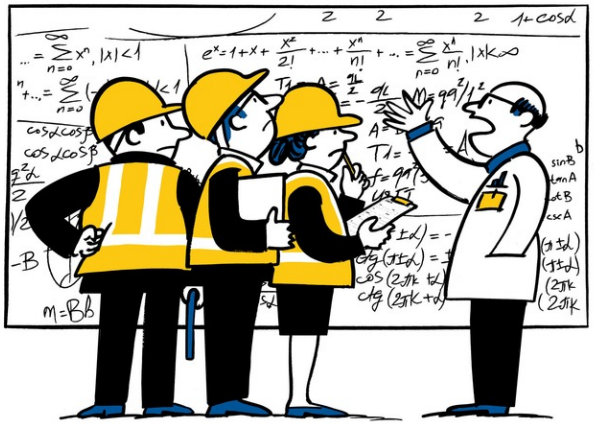
Regularisering

- **Spørsmål**: etter din mening, hvorfor bidrar regulariseringsmetoder (slik som L2-regularisering) til å redusere risiko for overtrening?
  - Svar på mentimeter

L2-regularisering:

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

Forklarbarhet



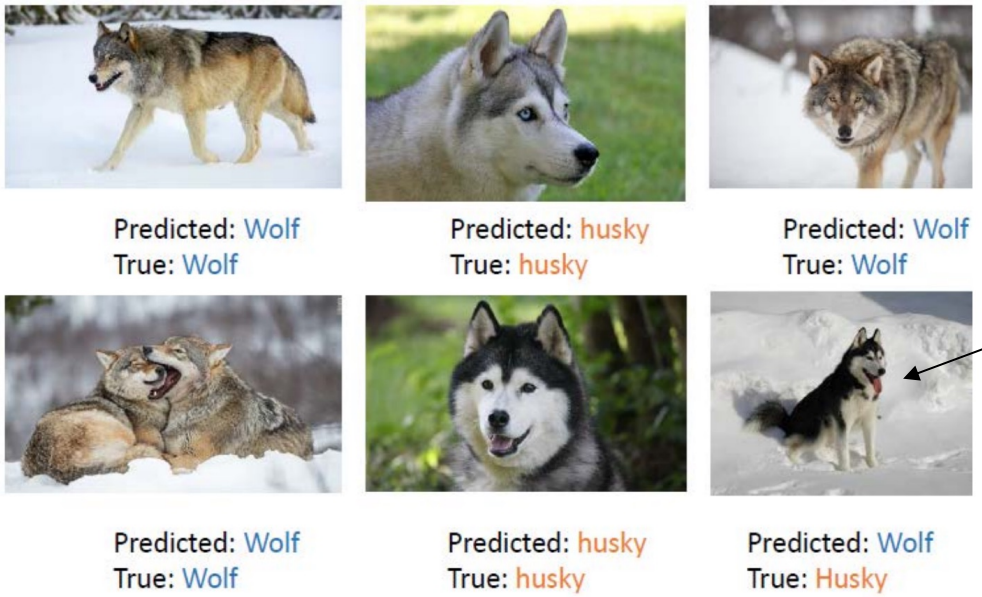
- Tilbake til hovedformelen:

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

- 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

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

(a) Husky classified as wolf (b) Explanation

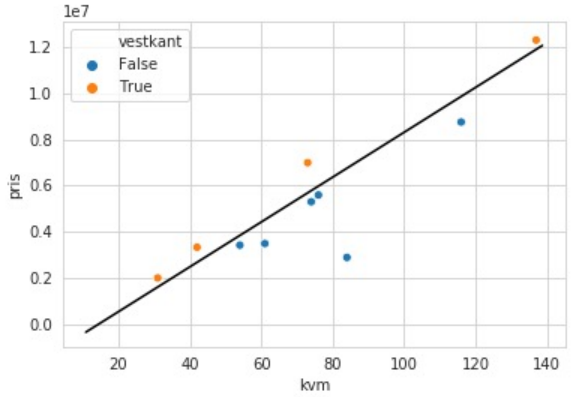
[Ribeiro et al. (2016). " Why should i trust you?" Explaining the predictions of any classifier. *In Proceedings of SIGKDD*.]

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

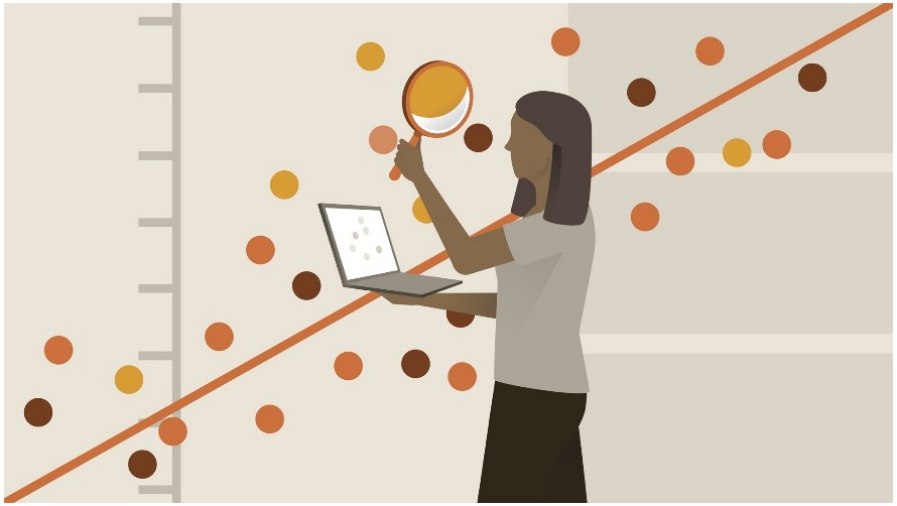
Forklarbarhet



- Ta igjen vår logistisk regresjonsmodell 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 sannsynligheten til å holde på østkant eller vestkant?
  - Svar på Mentimeter

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