TTK4260 Innføring i Multivariat Datamodellering

Kristian Løvland

Innhold

1 Introduksjon

Dette dokumentet er et forsøk på å skaffe meg en strukturert oversikt over et middels strukturert fag. Ting er først og fremst hentet fra slides, men også noe fra lærebøker jeg kanskje legger til som kilder senere.

2 Grunnleggende statistikk

Til grunn for alle multivariate metoder som kommer senere, ligger den grunnleggende statistikken som forhåpentligvis er kjent fra før. Den oppsummeres her i korte trekk.

2.1 Noen småting

2.1.1 Notasjon

Gjennom denne oppsummeringen brukes følgende notasjon

- Skalare variabler og funksjoner skrives med små bokstaver – $\alpha, x, g(\cdot)$
- Vektorvariabler og -funksjoner skrives med små, fete bokstaver – $\alpha, \mathbf{x}, \mathbf{g}(\cdot)$
- Matriser skrives med store bokstaver – A, X
- Transponert, invers, pseudoinvers – $\cdot^T, \cdot^{-1}, \cdot^\dagger$
- Definisjonsmengder skrives med kaligrafiske bokstaver, eller stor theta – $\mathcal{X}, \mathcal{Y}, \mathcal{D}$ eller Θ

2.1.2 Definisjoner

En funksjon $\phi(\boldsymbol{y}) : \mathcal{Y}^N \mapsto \mathbb{R}^M$ er en **observator** dersom den er målbar, og den er uavhengig av θ. En observator $\phi(\boldsymbol{y}) : \boldsymbol{y}^N \mapsto \Theta$ er en **estimator** dersom den er målbar og uavhengig av θ (dvs. den er en observator med verdimengde lik Θ.

2.1.3 Antagelser

Når vi snakker om regresjon, antar vi at dataen y_t genereres av funksjonen $y_t = f(u_t; \theta) + v_t$, og at dette resulterer i et datasett $\mathcal{D} = \{(u_t, y_t)\}_{t=1,\ldots,N}$. Ved hjelp av parametre $\theta \in \Theta$, hypoteserommet vårt, vil vi finne en estimator $\widehat{\theta} \in \Theta$ som best mulig forklarer datasettet vårt.

Hva betyr det å "best mulig forklare $\mathcal{D}$? Det finnes det ulike tolkninger av.

2.2 Minste kvadrater

En naturlig tolkning av spørsmålet om å best mulig forklare datasettet vha θ, er å finne parameteren som ved bruk av vår antatte modell $f(u_t;\theta)$, gir den korteste avstanden fra predikert datasett til faktisk datasett. "Avstand" har her den vanlige, euklidiske tolkningen, slik at en minste kvadraters estimator av en parameter θ gitt et datasett $\mathcal{D}$ og en modell $f(u_t;\theta)$, er gitt av

$$\widehat{\theta}_{\mathrm{LS}} = \arg\min_{\theta\in\Theta} \left\| \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} - \begin{bmatrix} f\left(u_1;\theta\right) \\ \vdots \\ f\left(u_N;\theta\right) \end{bmatrix} \right\|^2 = \arg\min_{\theta\in\Theta} \sum_{t=1}^{N} \left(y_t - f\left(u_t;\theta\right)\right)^2 \tag{1}$$

En nyttig verdi som følger av dette estimatet er residualen $r_t(\theta) := y_t - f\left(u_t;\theta\right)$.

En nyttig klasse av problemer er de **separable** problemene. Disse er på formen

$$y_t = \sum_{j=1}^{n} \theta_j \phi_j\left(u_t\right) + e_t \tag{2}$$

Dvs. at parameterne som skal estimeres inngår lineært i modellen vår. Da kan $\phi(u_t)$ være så komplisert den vil, LS-problemet vil uansett reduseres til et lineært likningssett på formen

$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} \phi_1\left(u_1\right) & \cdots & \phi_n\left(u_1\right) \\ \vdots & & \vdots \\ \phi_1\left(u_N\right) & \cdots & \phi_n\left(u_N\right) \end{bmatrix} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} + \begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix} \tag{3}$$

Som mer kompakt kan skrives som

$$\boldsymbol{y} = \Phi(\boldsymbol{u})\boldsymbol{\theta} + \boldsymbol{e} \tag{4}$$

Målet vårt er å minimere $\mathbf{e}$. Dette kan gjøres vha. lineær programmering, men om vi ikke har begrensninger i θ, dvs. $\Theta = \mathbb{R}^n$ for en eller annen n, kan dette løses eksplisitt som

$$\widehat{\theta}_{\mathrm{LS}} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^n} \|\boldsymbol{y} - \Phi(\boldsymbol{u})\boldsymbol{\theta}\|^2 \tag{5}$$

Ved å derivere dette og sette det lik null får vi **normallikningene**

$$\Phi(\boldsymbol{u})^T\Phi(\boldsymbol{u})\widehat{\theta}_{\mathrm{LS}} = \Phi(\boldsymbol{u})^T\boldsymbol{y} \tag{6}$$

Men hva om $\Phi(\boldsymbol{u})^T\Phi(\boldsymbol{u})$ ikke er invertibel? Da vil ikke normallikningene ha noen entydig løsning. Dette ordner vi ved å bruke **pseudoinversen**, som har noen kjekke egenskaper (nærmeste løsning hvis ingen løsning eksisterer, løsning med minst norm hvis mange løsninger eksisterer).

Noen ganger er imidlertid ikke alle tilstander like viktige. Dette løser vi ved å multiplisere hver feil med en vekt, slik at minimeringsproblemet (for ubegrensede problemer) blir

$$\widehat{\theta}_{\text{WLS}} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^n}(\boldsymbol{y} - \Phi(\boldsymbol{u})\boldsymbol{\theta})^T W(\boldsymbol{y} - \Phi(\boldsymbol{u})\boldsymbol{\theta}) \tag{7}$$

Mer kompakt kan dette skrives som $= \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^n} \|\boldsymbol{y} - \Phi(\boldsymbol{u})\boldsymbol{\theta}\|^2_{W^{-1}}$. Et nytt sett med normalligninger faller ut av dette, nå blir

$$\Phi(\boldsymbol{u})^T W\Phi(\boldsymbol{u})\boldsymbol{\theta} = \Phi(\boldsymbol{u})^T W\boldsymbol{y} \tag{8}$$

som kan løses likt som tidligere, f.eks. med pseudoinvers.

Hva om problemet vårt er ulineært (ikke separabelt)? Optimeringsproblemet kan fortsatt være veldefinert, og da kan det løses numerisk. MATLAB gjør dette med `fmincon`, og de fleste programmeringsspråk med respekt for seg selv har rammeverk som gjør det samme.

2.3 Maksimal sannsynlighet

Forrige avsnitt ga en rent geometrisk tolkning av minste kvadrater. Ofte har min imidlertid kunnskap om de statistiske egenskapene til støyen og feilen som forsøpler dataen din, og denne informasjonen er gjerne nyttig å bruke. Utgangspunktet for dette er sannsynlighetsfordelingen, skrevet som $p(y;\theta)$. Ofte operer man med θ fiksert og y varierende. I vårt tilfelle er y gitt (den utgjør datasettet vårt $\mathcal{D}$, og vi er ute etter å finne en θ som best forklarer dette. Da kalles $p(y;\theta)$ for **sannsynlighet**.

Med dette definert er vi klare for å definere vår maksimale sannsynlighetsestimator

$$\widehat{\theta}_{\text{ML}} = \arg\max_{\theta\in\Theta} p(\mathcal{D};\theta) \tag{9}$$

Vi ser at denne ligner i formen på LS-estimatoren, men at funksjonen p gir oss mer valgfrihet, og muligheter til å inkludere kjent informasjon om modell og data.

Det er verdt å merke seg at et ML-estimat ikke nødvendigvis trenger å eksistere. Man kan komme opp med massevis av eksempler på at denne ikke

eksiterer, f.eks. kan man la Θ er en åpen mengde. Hvis p er kontinuerlig og Θ er kompakt tror jeg imidlertid vi kan føle oss ganske trygge, i hvert fall hvis man er av typen som stoler på det Weierstrass hadde å si oss.

Et viktig eksempel på en ML-estimator er når p er normalfordelt. Da vil sannsynlighetsfunksjonen til et datasett være gitt av

$$p\left(y_1, \ldots, y_N; m, \sigma^2\right) = \prod_{t=1}^{N} p\left(y_t; m, \sigma^2\right) = \prod_{t=1}^{N} \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \frac{(y_t - m)^2}{\sigma^2} \right) \right) \tag{10}$$

Dette grumsete uttrykket motiverer definisjonen av log-sannsynlighetsfunksjonen. Siden $\log(\cdot)$-funksjonen er monotont stigende i input-argumentet sitt, vil maksimerende input til funksjonen være lik maksimerende input til logaritmen av funksjonen. Vi definerer

$$\ell(\theta) := -\log p(\mathcal{D}; \theta) \tag{11}$$

I eksempelet med normalfordelt p vil vi nå kunne formulere

$$\widehat{\theta}_{\mathrm{ML}} = \arg\max_{\theta \in \Theta} p(\mathcal{D}; \theta) = \arg\min_{\theta \in \Theta} \ell(\theta) \tag{12}$$

Eksponential- og logaritmefunksjonen spiller hverandre gode, og ved litt regning kan man se at

$$\arg\min_{m \in \mathbb{R}, \sigma^2 \in \mathbb{R}_+} \ell\left(m, \sigma^2\right) = \arg\min_{m \in \mathbb{R}, \sigma^2 \in \mathbb{R}_+} N \log\left(\sigma^2\right) + \frac{\sum_{t=1}^{N} (y_t - m)^2}{\sigma^2} \tag{13}$$

Dennes gradient settes lik null, men her vil det inngå informasjon vi ikke har tilgang på. Vi ender opp med å benytte estimatene, og får

$$\bar{m} = \frac{1}{N} \sum_{t=1}^{N} y_t \tag{14}$$

$$\bar{\sigma}^2 = \frac{1}{N} \sum_{t=1}^{N} (y_t - \bar{m})^2 \tag{15}$$

som ser fornuftig ut.

2.4 Maksimal a posteriori

Vi begynner med Bayes' lov

$$P(A|B) = \frac{P(AB)}{P(B)} = \frac{P(B|A)P(A)}{P(B)} \tag{16}$$

som kan bevises ved Venn-diagram eller lignende.

Ved å bytte ut A og B med θ og y, får vi en måte å oppdatere vår tro om en variabel sin fordeling på, basert på data. Vi er imidlertid avhengige av å ha en initiell formening om fordelingen til θ, dette kalles en **prior**. I praksis vil denne gjerne gjøre få antagelser, men utelukke fullstendig urealistiske muligheter (f.eks. utelukke negativ høyde på personer, om man vil estimere dette). Med en modell $P(y|\theta)$, en prior $P(\theta)$, og data som gir oss $P(y)$, får vi da

$$P(\theta|y) = \frac{P(y|\theta)P(\theta)}{P(y)} \tag{17}$$

Basert på denne nye, evidensbaserte fordelingen, kan vi definere estimatoren

$$\widehat{\theta}_{\text{MAP}} := \arg\max_{\theta\in\Theta} P(\theta|y) = \arg\max_{\theta\in\Theta} \frac{P(y|\theta)P(\theta)}{P(y)} = \arg\max_{\theta\in\Theta} P(y|\theta)P(\theta) \tag{18}$$

som vil være moden til den posteriore fordeling. Merk at denne ikke trenger å være representativ for fordelingen, f.eks. hvis fordelingen har en smal, høy topp langt unna "tyngdepunktet".

2.5 Statistiske ytelsesindekser

Det nevnes tre bruksområder for indekser som måler statistisk ytelse

- Regresjonsproblemer
- Klassifiseringsproblemer
- Sammenligning av sannsynlighetsfordelinger

Vi går gjennom disse i den rekkefølgen.

2.5.1 Regresjonsproblemer

En naturlig indeks er **Mean Squared Error (MSE)**

$$\text{MSE} = \mathbb{E}\left[\|\theta - \widehat{\theta}\|^2\right] \tag{19}$$

Av denne følger **Root Mean Square Error**

$$\text{RMSE} = \sqrt{\mathbb{E}\left[\|\theta - \widehat{\theta}\|^2\right]} \tag{20}$$

Det er viktig å merke seg at siden MSE er en funksjon av θ, så kan den ikke regnes ut. Hvorfor bryr vi oss om den da? Tja, den kan i hvert fall inspirere lignende indekser. **Residual Sum of Squares (RSS)** baserer seg på residualene til estimatene

$$\text{RSS}(\widehat{\theta}) := \sum_i \left(y_i - \widehat{y}_i(\widehat{\theta})\right)^2 \tag{21}$$

Det finnes imidlertid problemer med alle disse. Først og fremst er det et problem at de er avhengig av mengden av og størrelsen på dataen man vurderer estimatene av. Man vektlegger å unngå avvik i estimatene fra store målinger mer enn små. En metode som fungerer noe bedre, uten å bruke normalisering, er å bruke 1-normen i stedet for kvadratet. Dette gjøres i **Mean Absolute Deviaton (MAD)**

$$\text{MAD} := \mathbb{E}[|y - \widehat{y}|] \tag{22}$$

En metode som bruker en form for normalisering er **Fraction of Variance Unexplained (FVU)**

$$\text{FVU}(\widehat{\theta}) := \frac{\text{RSS}(\widehat{\theta})}{\text{var}(y)} = \frac{\sum_i \left(y_i - \widehat{y}_i(\widehat{\theta})\right)^2}{\sum_i \left(y_i - \frac{1}{N}\sum_i y_i\right)^2} \tag{23}$$

Denne må tolkes med måte, siden hva som er en god forklaringgrad er veldig avhengig av hva slags felt man jobber i, og det konkrete bruksområdet. Dette er uansett en mye brukt indeks, men da i form av R^2 Denne tolkes som "andel av variansen i avhengig variable som er predikerbar fra de uavhengige variablene".

2.5.2 Klassifiseringsproblemer

Vi diskuterer her klassifisering i form av "ja/nei". Da kan man gjøre to typer feil: Falsk positiv (**Type 1**) og falsk negativ (**Type 2**). Figur 1 viser definisjonen på en del uttrykk som beskriver egenskapene til en klassifikator.

$$\text{accuracy} := \frac{\#\text{ of true positives} + \#\text{ of true negatives}}{\text{total }\#\text{ of instances}}$$

$$\text{precision} := \frac{\#\text{ of true positives}}{\#\text{ of true positives} + \#\text{ of false positives}} = \frac{\#\text{ of true positives}}{\text{total }\#\text{ of instances predicted as true}}$$

$$\text{sensitivity} := \frac{\#\text{ of true positives}}{\#\text{ of true positives} + \#\text{ of false negatives}} = \frac{\#\text{ of true positives}}{\text{total }\#\text{ of true instances}}$$

$$\text{specificity} := \frac{\#\text{ of true negatives}}{\#\text{ of true negatives} + \#\text{ of false positives}} = \frac{\#\text{ of true negatives}}{\text{total }\#\text{ of negative instances}}$$

Fig. 1: Egenskaper til klassifikator

Muntlig kan disse forklares som

- **Prevalens** – Hvor ofte opptrer ja-tilfellet i datasettet vårt?
- **Nøyaktighet** – Hvor ofte har klassifikatoren rett?
- **Feilklassifiseringsrate** – Hvor ofte tar klassifikatoren feil?
- **Presisjon** – Når klassifikatoren gjetter ja, hvor ofte er dette rett?
- **Falsk positiv-rate** – Når svaret er nei, hvor ofte gjetter klassifikatoren ja?
- **Sensitivitet** – Når svaret er ja, hvor ofte gjetter klassifikatoren ja?
- **Spesifitet** – Når svaret er nei, hvor ofter gjetter klassifikatoren nei?

Man kan også kombinere to av disse for å få **F1-score**

$$\text{F1-score} = 2\frac{\text{presisjon} \cdot \ \text{sensitivitet}}{\text{presisjon} + \ \text{sensitivitet}} \tag{24}$$

Om man vil finne ut hvor ulike to klassifikatorer er, kan man bruke **Kappa-koeffisient**. Dette forklares ikke mer, men det eksisterer.

2.5.3 Sammenligning av sannsynlighetsfordelinger

Her kan man bruke **Kullback-Leibler-divergens**.

2.6 Bias vs. varians

Dette er en avveining man ikke slipper unna når men bedriver estimering. La oss bruke MSE for å illustrere dette. La

$$\begin{aligned} \mathcal{V} &:= \widehat{\theta} - \mathbb{E}[\widehat{\theta}] \\ \mathcal{B} &:= \mathbb{E}[\widehat{\theta}] - \theta \end{aligned} \tag{25}$$

$$\begin{aligned} \mathbb{E}\left[\|\hat{\theta} - \theta\|^2\right] &= \mathbb{E}\left[\|\widehat{\theta} - \mathbb{E}[\widehat{\theta}] + \mathbb{E}[\widehat{\theta}] - \theta\|^2\right] \\ &= \mathbb{E}\left[\|\mathcal{V} + \mathcal{B}\|^2\right] \\ &= \mathbb{E}\left[(\mathcal{V} + \mathcal{B})^T(\mathcal{V} + \mathcal{B})\right] \\ &= \mathbb{E}\left[\|\mathcal{V}\|^2 + \|\mathcal{B}\|^2 + 2\mathcal{V}^T\mathcal{B}\right] \\ &= \mathbb{E}\left[\|\mathcal{V}\|^2\right] + \|\mathcal{B}\|^2 \end{aligned} \tag{26}$$

Denne avveiningen henger sammen med hvor komplisert man gjør forklaringsmodellen $f(u_t; \theta)$. Om man gjør den veldig komplisert vil man kunne følge dataen nøyaktig, men man vil være utsatt for at dette ikke lar seg generalisere til andre datasett **overfitting**. Dette svarer til lav bias, men stor varians. Om modellen er for enkel vil man få en enkel modell som generaliserer, men man vil også kunne unngå å beskrive viktig struktur i dataen. Dette er **underfitting**, og svarer til liten varians, men stor bias.

Det finnes flere metoder som forsøker å gjøre denne avveiningen. Noen av dem er

- Akaikes informasjonskriterium
- Det Bayesiske informasjonskriteriumet
- Minimum lengde-beskrivelse

3 Multivariat Dataanalyse

Her begynner de nye metodene. Jeg ser frem til å lære ny teori, og på å anvende denne 😎.

3.1 Eksperimentdesign

Eksperimentdesign er forhåndsplanlagt, systematisk variasjon av kontrollerte faktorer i et eksperiment, med det formål å få mest mulig informasjon med minst mulig innsats. Dette er et alternativ til den naive måten å gjøre det på, dvs. å endre én variabel om gangen. Denne måten å gjøre det på har flere fordeler. I tillegg til den åpenbare fordelen med et lavt, forutsigbart antall eksperimenter som må gjennomføres, får man mer informasjon om interaksjon mellom variabler. Det finnes flere ulike typer eksperimentdesign. Her går vi gjennom noen av dem.

3.1.1 Factorial designs

Dette er den enkleste metoden som gås gjennom. Den er optimal for å detektere hovedeffekter, og deres interaksjoner. Den er også basisen for mange andre ttyper eksperimentdesign. Her velger man seg to verdier for hver input-variabel, og gjennomfører et eksperiment for hver av de mulige kombinasjonene av disse. Altså trenger man 2^n eksperimenter når man har n variable. La y_+ og y_- være observasjonene av en variabel y i de to ulike tilstandene, og n være antall observasjoner for hver av disse. Effekten av endringene i y er da gitt av

$$\Delta y = \frac{\sum y_+}{n_+} - \frac{\sum y_-}{n_-} \tag{27}$$

Før man begynner å gjennomføre et eksperiment, er det viktig å regne ut den statistiske styrken til eksperimentet. Denne er avhengig av

1. δ: Hva som regnes som en signifikant endring
2. σ: Støy i målt responsvariabel
3. Antall eksperimenter
4. α: Signifikansnivå

Basert på disse parameterne kan man regne ut

$$\text{Statistisk styrke} = (1 - \beta) \cdot 100\% \tag{28}$$

Denne kan tolkes som sannsynligheten for å "avsløre" en effekt av størrelsedelta, gitt støyet som nevnt over. Dette er en signal til støy-ratio, gitt av $\frac{\Delta}{\sigma}$. Denne burde være høy, minst 80%. I tillegg til å gjennomføre eksperimentene som nevnt over, kan det være nyttig å gjennomføre noen ekstra eksperimenter der tilstandene befinner seg rundt middelverdi, for å kunne estimere feilvarians. Repliserte eksperimenter er heller aldri noen ulempe å gjennomføre.

3.1.2 Multippel lineær regresjon (MLR)

En mer fornuftig (synes jeg) måte å analysere multivariabel data på er å sette opp en enkel lineær modell

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_k x_k + f \tag{29}$$

Den beste tilpasningen av en slik modell får man ved å minimere summen av kvadratavviket mellom modellprediksjon og faktisk data, som forklart i avsnitt 2.2. Når dette er gjennomført analyserers resultatene ved hjelp av en ANOVA-tabell (analysis of variance). I denne deles data inn i bidrag fra hhv. struktur og støy, slik at kvadratsummen

$$SS_{total} = SS_{model} + SS_{error} \tag{30}$$

Gitt k forklaringsvariable og l forsøk, vil ANOVA-tabellen for hele modellen se ut som i tabell 1 (her betyr MS mean square). F-verdien er et mål på statistisk signifikans, og deler navn med fordelingen den er basert på. Denne observatoren oppfyller $E[F] = 1 + n\frac{\sigma^2_{model}}{\sigma^2_{error}}$, og en F-verdi langt over 1 gir dermed grunnlag for å forkaste nullhypotesen om at modellen vår ikke forklarer resultatene.

Kilde	**SS**	**df**	**MS**	**F-ratio**	**p-verdi**
Modell	SS_{modell}	k	$MSR = SS_{modell}/(k)$	MSR/MSE	p
Feil	SS_{feil}	$l - k - 1$	$MSE = SS_{feil}/(l - k - 1)$		
Total	SS_{total}	$l - 1$	$MST = SS_{total}/(l - 1)$		

Table 1: ANOVA-tabellstruktur

Man kan også sette opp en ANOVA-tabell for enkeltledd i modellen, hver av disse vil ha samme struktur som øverste rad i tabelll **??**.

3.1.3 Fractional Factorial Designs

Jeg vet ikke hva dette heter på norsk.

Full Factorial designs er kostbare hvis det er mange variabler. I Fractional Factorial designs gjennomfører man kun en delmengde av disse, og forsøker å dekke så "mye" som mulig av designrommet. Prisen man betaler er at estimatene av alle variablene ikke lenger kan være uavhengige.

For eksempel, gitt tre designvariabler A, B, C. Ved å velge å kun gjennomføre eksperimenter i planet $C = AB$, kan man halvere antall eksperimenter som gjennomføres.

3.1.4 Optimeringsbaserte design

Om man vet mer om hva slags respons man forventer i modellen sin, kan man utnytte denne informasjonen til eksperimentdesign. Det finnes mange metoder som baserer seg på ulike optimalitetskriterier. I pensum trekkes **Central Composite Design** og **Bob-Behnken Designs**, frem som viktige metoder. Begge er eksempler på bruk av **Response Surface Methodology**. Om man kjenner til begrensninger i tilstandsrommet kan man bruke dette til å utelukke umulige tilstander fra eksperiemntdesignet.

Uansett hvordan man velger å gjennomføre eksperimentene, bør man plotte fornuftige plott når man er ferdig. Dette gjelder stort sett alltid, har jeg inntrykk av.

3.2 PCA

Prinsipalkomponentanalyse (PCA) er en veldig nyttig metode som benyttes til bl.a. mønstergjenkjenning, dimensjonalitetsreduksjon, klyngeanalyse, klassifisering og utliggerdeteksjon. Den baserer seg på noen viktige konsepter fra lineæralgebra.

3.2.1 Matematisk bakgrunn

La vektorene $\mathbf{x}, \mathbf{y} \in V$, der V er et indreprouktrom. Disse vektorene er **ortogonale** dersom indreproduktet $\langle \mathbf{x}, \mathbf{y} \rangle = 0$. To underrom A og B er ortogonale dersom alle vektorer i A er ortogonal med alle vektorer i B, og omvendt. En mengde vektorer S er ortonormal dersom alle vektorene den inneholder er ortogonale med hverandre, og har norm 1.

Projeksjonen av vektoren $\mathbf{y}$ på $\mathbf{x}$ er gitt av

$$\text{Proj}_{\mathbf{x}}\mathbf{y} = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{|\mathbf{x}|^2}\mathbf{x} \tag{31}$$

En matrise A kan sees på som en **lineærtransformasjon**, f.eks. en rotasjon eller en skalering i en viss retning.

3.2.2 Tolkning av PCA

PCA går kort oppsummert ut på å finne vektorer som best mulig forklarer varians i dataen, og som er ortogonale med hverandre. Håpet er at noen få av disse **prinsipale komponentene** forklarer en stor del av variansen. Metoden har en fin geometrisk tolkning. La S være en sirkel, og A være en lineærtransformasjon (f.eks. representert av en matrise). Da vil A (med mindre den er singulær) ha egenvektorer $\mathbf{v}_1$ og $\mathbf{v}_2$. Disse vil da strekkes med faktorene σ_i, egenverdiene deres. Dette er vist i figur 2.

For egenvektorer $\mathbf{v}_i$ med egenverdier σ_i vil dermed

$$A\mathbf{v}_j = \sigma_j \mathbf{u}_j \tag{32}$$

For høyere dimensjon enn 2 (som i figuren) vil likningene være helt like. En hypersfære $\mathbf{V}$ av egenverdier transformeres av A til en hyperellipse $\mathbf{U}$ under likningen

$$A\mathbf{V} = \mathbf{U}\mathbf{\Sigma} \tag{33}$$

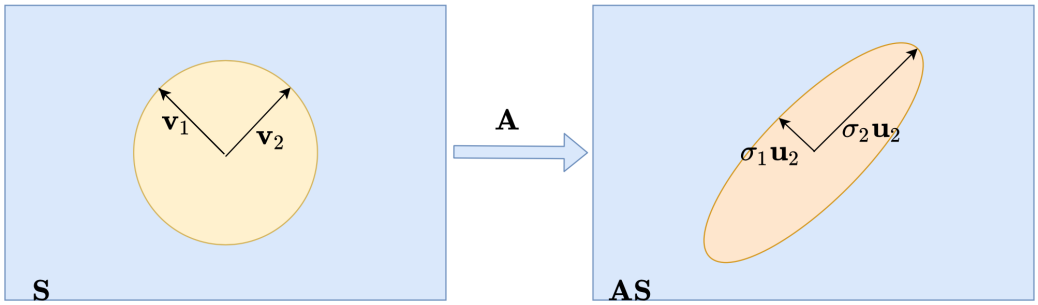

Fig. 2: Geometrisk tolkning av lineærtransformasjon

Hvis egenvektorene til A er lineært uavhengige, vil $\mathbf{V}$ være invertibel, og vi kan definere **singulærverdidekomposisjonen**

$$A = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \tag{34}$$

Ved å slå sammen matrisene $\mathbf{U}\mathbf{\Sigma}$ står vi igjen med en ny tolkning av A, forklart av kolonnene i $\mathbf{V}^*$ og kolonnene i $\mathbf{U}\mathbf{\Sigma}$. Dette er illustrert i figur 3. Mye av styrken til PCA ligger i at hvis A har en lav nok underliggende dimensjon, kan vi approksimere A ganske bra ved å kaste bort vektorene som svarer til lave singulærverdier σ_i. Modellordenen reduseres drastisk uten at vi mister viktig informasjon 😎.

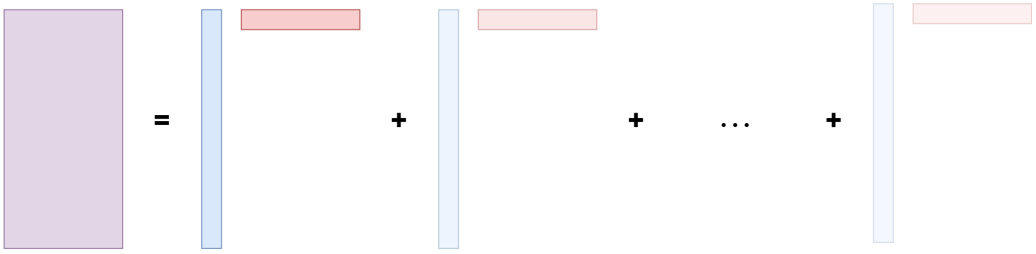

Fig. 3: Tolkning av SVD

3.2.3 Utledning av prinsipale komponenter

Det finnes flere måter å finne de prinsipale komponentene. En av dem, kalt **egenverdidekomposisjonen**, baserer seg på at vi ønsker å maksimere variansen i ladningene $\mathbf{t} = \mathbf{X}\mathbf{p}$, der X er dataen vår er p er en prinsipal kompo-

nent (slik at $\mathbf{t}_j = \text{proj}_{\mathbf{p}}\mathbf{X}$, stol på dette). For å finne p_1, kan vi dermed løse optimeringsproblemet

$$\max(\phi) = \boldsymbol{p}_1^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{p}_1 - \lambda \left(\boldsymbol{p}_1^T \boldsymbol{p}_1 - 1 \right) \tag{35}$$

som er funnet ved å inkludere begrensningen $\mathbf{p}_1^T \mathbf{p} = 1$ som Lagrange-multiplikator. Ved å derivere mhp. $\mathbf{p}_1$ og sette lik null, finner man at løsningen er gitt av likningen

$$\mathbf{X}^T \mathbf{X} \mathbf{p}_1 = \lambda_1 \mathbf{p}_1 \tag{36}$$

Ved å innføre begrensningen $\mathbf{p}_1 \perp \mathbf{p}_2$ kan man finne at $\mathbf{p}_2$ kan finnes på akkurat tilsvarende vis osv. for de neste prinsipale komponentene.

PCA kan også gjøres ved å finne SVD-en til X, dette finnes det funksjoner for i de fleste `programmeringsspråk` som brukes til dataanalyse. Når man kjenner denne er dekomposisjonen av $\mathbf{X}$ i ladninger $\mathbf{p}_i$ med tilhørende score $\mathbf{t}_i$ gitt av

$$\mathbf{X} = \mathbf{t}_1 \mathbf{p}_1^T + \mathbf{t}_2 \mathbf{p}_2^T + \cdots + \mathbf{E} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \mathbf{E} \tag{37}$$

En effektiv metode for utregning av PC-er kalles Nonlinear Iterative Partial Least Squares (**NIPALS**). Denne ble blant annet brukt i tidlige versjoner av Google sin PageRank-algoritme. Algoritmen går ut på å iterativt finne hver prinsipale komponent, og å fjerne variabiliteten denne bidrar med før neste PC regnes ut. Noen fordeler med NIPALS er at man kan håndtere store datamengder i $\mathbf{X}$, at man kan håndtere manglende data, og at man er garantert konvergens.

3.2.4 Visualisering av PCA

Det finnes mange nyttige måter å bruke PCA på for å visualisere data. Kanskje blir dette kapitlet lengre etter hvert.

3.3 Valg av modellorden, og validering

I forrige delkapittel lovpriste vi PCA-ens evne til å reduserte dimensjonen til en datamatrise uten at vi mister informasjon. Men hvor mye kan man egentlig redusere denne dimensjonen, og hvordan velger man hvilken informasjon man har råd til å kaste vekk? Det er denne typen problemstilling **modellordenseleksjon** handler om.

3.3.1 Problemformulering

Modellordenseleksjon er et problem i mengden av **modellseleksjonsproblemer**. Disse går ut på å velge en statistisk modell av flere gitt et datasett som skal forklares. Dette kan virke enkelt (bare velg modellen som gir lavest kvadratavvik for datasettet da vel!), men få datasett representerer virkeligheten 100%. Bias vs. varians-tradeoffen gjør seg dessverre gjeldende og tvinger oss til å balansere på en knivsegg mellom over- og undertilpassing av modellen vår. Uff.

3.3.2 Motivasjon

Anta at vi har et datasett med én observasjon, $y \sim \mathcal{N}(\mu, \sigma^2)$, som estimeres vha. funksjonen $\hat{\mu}(y)$. Hvordan vet vi hvor god denne stimatoren er? Vi kan jo forsøke å finne MSE av estimatoren direkte, det er bare ett problem (her har jeg vært litt lat med å vise utregninger):

$$\mathbb{E}[(\mu - \hat{\mu})^2] = \cdots = -n\sigma^2 + \mathbb{E}[(y-\mu)^2] + \underbrace{2\text{cov}(y, \mu)}_{\text{trøbbel}} \tag{38}$$

Det siste leddet har man vanligvis ikke kjennskap til. Hvordan kan vi da anslå hvor god en estimator er, når man ikke er så heldig å ha fått utdelt et testsett å teste på?

3.3.3 Kryssvalidering

Om man ikke har et dedikert testsett å validere en modell på, kan man lage sitt eget ved å dele opp treningssettet. Dette kalles **kryssvalidering**. Ulike måter å dele opp på gir ulike former for kryssvalidering.

Det er ønskelig å ha et så stort treningssett som mulig, så et naturlig valg vil være å kun utelate én observasjon som treningssett. Ved å utelate en og

en observasjon, kan man regne ut n ulike MSE-er. Gjennomsnittet av disse gir **Leave One Out Cross-Validation (LOOCV)**.

Å tilpasse en modell n ganger for et datasett på n observasjoner kan være beregningskrevende (i tillegg til å gi høy varians i MSE, uten at jeg helt kan forklare hvorfor). I stedet for å utelate én observasjon som treningssett n ganger, kan man utelate $\frac{n}{k}$ observasjoner k ganger, og regne ut gjennomsnittet av de k MSE-ene dette resulterer i. Dette kalles **k-Fold Cross-Validation**, og regnes altså ut som

$$\mathrm{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \mathrm{MSE}_i \tag{39}$$

Om man vil ha en likning for LOOCV får man det ved å sette $k = n$ her.

Uansett hvordan man velger k, er det viktig å sørge for at naturlige strukturer ikke brytes. For eksempel burde et datasett av vær over flere år deles opp i år, slik at sesongvariasjoner plukkes opp. Fullstendig tilfeldig inndeling vil her kunne gi merkelige resultater.

3.3.4 Valg av antall PCA-komponenter

Et spesialtilfelle av modellordenseleksjon er valg av antall PCA-komponenter å beholde. Noen muligheter som nevnes er

- Bartletts test
- "Broken stick"-testesn
- Behold alle egenverdier ≤ 1 (for variabler skalert til å ha enhetsvarians)
- La summen av PC-ene forklare 95% av variansen (ikke anbefalt)
- Bruk et SCREE-plot til å tolke viktigheten av komponentene

I kombinasjon med sistnevnte kan kjennskap til systemet og dens dimensjonalitet være nyttig å bruke. Dette gjelder også når man skal gjøre kryssvalidring av PCA-modeller. I tillegg er det viktig å sjekke stabiliteten (dvs. sensitivitet til variasjoner i treninsdata). Om dette er uklart burde følgende figur forklare alt

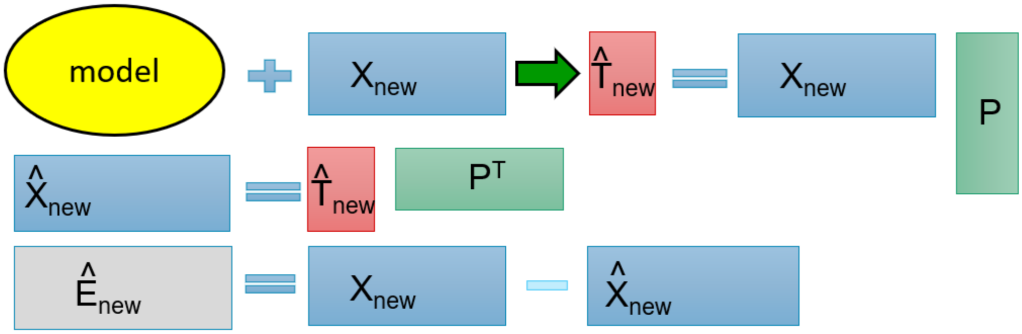

Fig. 4: Selvforklarende figur

3.3.5 Informasjonskriterier

Om man har begrenset med regnekapasitet eller små datasett med dårlige muligheter for oppdeling, kan kryssvalidering være upraktisk å gjennomføre. Da kan at alternativ være å bruke et **informasjonskriterie**. Disse gir oss estimatorer for MSE som kompenserer for det faktum at MSE på treningssett typisk er underestimater.

Anta at vi for en modell av orden n, basert på et datasett (D), har gitt et estimat θ_n. Observasjonen estimatet vårt er basert på er fordelt med sannsynlighetstettheten $p_n(\mathcal{D}, \theta_n)$. Vi ønsker å minimere avstanden (hva nå enn det måtte bety) til den faktiske fordelingen $p_0(\mathcal{D}, \theta_0)$. En naturlig måte å måle denne avstanden på er ved å bruke Kullback-Leibler-divergens, som måler informasjon tapt når man representerer p_0 ved p_n. Denne er gitt av

$$KL\left(p_0, p_n\right) := \int \log\left(\frac{p_0\left(\mathcal{D}, \theta_0\right)}{p_n\left(\mathcal{D}, \theta_n\right)}\right) p_0\left(\mathcal{D}, \theta_0\right) d\mathcal{D} \tag{40}$$

Det kan vises at denne er lik $\mathbb{E}_{p_0}[\ell(\mathcal{D}, \theta_n)]+$ et eller annet uavhengig av θ_n. Det siste leddet betyr at vi mister informasjon om absolutt informasjonstap, men om vi kun er interessert i å sammenligne ulike θ_n er ikke dette noe problem. Om vi finner et estimat av denne forventingsverdien kan vi minimere mhp. n, og dermed bestemme hvilken modell som er best etter dette kriteriet.

Akaike gjorde noen antagelser, fant et estimat, og fikk kriteriet dette resulterte i oppkalt etter seg (**AIC**). Dette er gitt av (der ε er feil, typisk $y - \hat{y}$ e.l.)

$$\mathrm{AIC} = \frac{1}{N}\sum_{t=1}^{N} \ell(\varepsilon(t,\theta_n)) + \frac{1}{N}\dim(\theta_n) \tag{41}$$

Under andre antakelser kan man finne andre kriterier. Et annet mye brukt kriterie (gjerne på mindre datasett, siden den ikke er en konsistent estimator) er det Bayesiske informasjonskriteriet (**BIC**)

$$\mathrm{BIC} = \frac{1}{N}\sum_{t=1}^{N} \ell(\varepsilon(t,\theta_n)) + \frac{\log N}{2N}\dim(\theta_n) \tag{42}$$

3.3.6 Jackknifing og bootstrapping

Når man har en estimator, kan det være nyttig å estimere hvor god den er. Fest setebeltet, for dette kommer til å bli META.

Jackknifing er en enkel metode for å estimere bias og varians til en estimator, som typisk bruke spå små datasett. Man begynner med å dele opp datasettene i delmengdene $X_{[i]} = \{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n\}$. For hver av disse defineres estimatoren $\hat{\theta}_{[i]} = \hat{\theta}(X_{[i]})$. La gjennomsnittet av alle disse være $\hat{\theta}_{\text{ave}}$. Da er

$$\widehat{\mathrm{bias}}(\widehat{\theta})_{\text{jk}} := \frac{n-1}{n}\sum_{i=1}^{n}\left(\widehat{\theta}_{[i]} - \widehat{\theta}\right) \quad \mathrm{var}(\widehat{\theta})_{\text{jk}} := \frac{n-1}{n}\sum_{i=1}^{n}\left(\widehat{\theta}_{[i]} - \widehat{\theta}_{\text{ave}}\right)^2 \tag{43}$$

Metoden kan også generaliseres til å fjerne k elementer i hvert reduserte datasett. Man kan se at jackknifing i praksis er en form for kryssvalidering.

En mer sofisitkert metode er **bootstrapping**. Denne håndterer bl.a. skjeve fordelinger bedre. Metoden går ut på å generere B nye datasett av samme dimensjon som den originale ved å plukke observasjoner *med tilbakelegging*. F.eks. kan man fra datasettet $\{1,2,3,4\}$ generere mengden $\{1,4,2,2\}$. Deretter kan man regne ut bias og varians som i likning 43.

Hvis man kjenner fordelingen til en variabel θ kan man f.eks. bruke estimatene til å finne et tosidig α-konfidensintervall $P[|\hat{\theta} - \theta| \leq \alpha]$.

3.4 Mer om multippel lineær regresjon

Med den nye multivariate statistikken som er innført skulle man kanskje tro at den multiple lineære regresjonen som er beskrevet tidligere ville være lite nyttig. Vi skal her se at dette er det motsatte av riktig (det er feil). I dette avsnittet antar vi at målingene våre er generert av prosessen $y = X\theta + e$.

3.4.1 OLS

Vanlig minste kvadrater (OLS) for dataen vår minimerer e i likningen $y = X\theta + e$, og er følgelig gitt av

$$\hat{\theta}_{\text{OLS}} = (X^T X)^{-1} X^T y \tag{44}$$

Den numerisk anlagte leser vil reagere på leddet $(X^T X)^{-1}$, og det med rette. Hvis X inneholder kolonner som avhenger mer eller mindre lineært av hverandre (høy kolinearitet) vil det kunne føre til numerisk ustabilitet. Dette er en av flere problemer som motiverer metodene som snart beskrives.

3.4.2 TLS

I Total Least Squares (TLS) minimerer man ikke størrelsen til residualene, men heller størrelsen til projeksjonen av disse på modellen sin. Likningene er litt mer kompliserte, men tolkningen av hva man gjør ligner på tolkningen av PCA-komponenter.

3.4.3 PCR

Om man er fornøyd med resultatene fra en PCA, kan man være fristet til å benytte seg av de nye PC-ene som forklaringsvariabler. Det er det ingen som stopper deg i å gjøre. Gitt prinsipale komponenter $p_1, p_2, \ldots, p_n$, kan man gjøre lineær regresjon på disse, ved å minimere e i likningen

$$y = \hat{\theta}_{\text{PCR}} p \tag{45}$$

Forhåpentligvis gir dette en modell som plukker opp mer av den underliggende strukturen til prosessen som genererer dataen, og mindre av støyet. Kriteriene for at PCR skal fungere bra er stort sett de samme som for at PCA skal fungere bra. En forskjell det er verdt å merke seg er imidlertid at når man skal velge antall komponenter, vil det i en PCR gi mer mening å

velge basert på hvor godt den resulterende modellen forklarer y (gjerne ved hjelp av et testsett) enn å gjøre som man vanligvis gjør i PCA.

Gitt PCA sin evne til å redusere dimensjonaliteten til et datasett, kan man være fristet til å tro at PCR på en eller annen måte reduserer dimensjonaliteten til forklaringsmodellen vår. Dette er ikke tilfellet, siden hver PC er avhengig av alle variablene i vår opprinnelige modell. Heldigvis finnes det en metode for dette også.

3.4.4 Regularisering

I vanlig, lineær regresjon vil man i de fleste tilfeller ende opp med modeller som tar i bruk alle mulige forklaringsvariabler. Sunn fornuft kan ofte fortelle oss at vi ikke trenger alle disse variablene. I mange tilfeller vil variabelmisbruk kunne føre til overtilpasning til det gitte datasettet. Regularisering er en måte å inkorporere sunn fornuft i modellen vår på, og er rett frem å gjøre når modelltilpasningen er formulert som et optimeringsproblem. Da kan vi simpelthen legge til en kost for variabelbruk. Dvs. at vi legger føringer for strukturen til modellen vår $y = f(x)$ med en egen kostfunksjon $R(f)$, slik at kostfunksjonen vår for vanlig LS blir

$$\sum_{i=0}^{n}(y_i - f(x_i))^2 + \lambda R(f) \tag{46}$$

der λ er en tuningparameter. Valget av denne er viktig, og man tester gjerne ut mange ulike, og velger den endelige verdien vha. f.eks. kryssvalidering eller en annen form for modellseleksjon.

Det finnes mange typer regularisering. For $f(x)$ lineær er to vanlige valg **Tikhonov-regularisering/Ridge-regresjon**, hvor

$$R(f(x)) = R(\theta_1 x_1 + \cdots + \theta_n x_n) = \sum_{i=1}^{n} \theta_i^2 \tag{47}$$

og en metode som enda hardere driver små θ_i til å bli eksakt 0, **LASSO**, der

$$R(f(x)) = R(\theta_1 x_1 + \cdots + \theta_n x_n) = \sum_{i=1}^{n} |\theta_i| \tag{48}$$

For oss som virkelig er svake for modellskralhet finnes også **L_0-regularisering**, der $R(f) =$ antall ikke-null koeffisienter i f.

3.4.5 Gauss-Markovs teorem

Nå har vi gått gjennom mange former for multippel lineær regresjon, men hvilken er egentlig best? Det kommer jo an på mange ting, både datasett, faktisk modellstruktur, og mange andre ting. Men én ting kan vi si sikkert. Det er at om vi antar at $\mathbb{E}[e] = 0$, at e er homoskedastisk ($\text{var}(e) = \sigma^2$ konstant og lik for alle e_i), og at e for alle observasjoner er ukorrelerte ($\text{cov}(e_i, e_j) = 0$), så er $\hat{\theta}_{\text{OLS}}$ den beste lineære forventingsrette estimatoren (BLUE) av θ. Dette er Gauss-Markovs teorem.

3.5 PLS

PCR har en åpenbar svakhet: Metoden baserer seg på antagelsen om at de prinsipale komponentene, kun utledet fra x, også forklarer variasjon i y. PLS ligner på PCR, men kvitter seg med denne antagelsen, og sikter heller på å finne komponenter som forklarer variasjon i y. Metoden beskrives ikke av noen enkel likning, men baserer seg på å regne ut latente strukturer (PLS forklares gjerne som Projection into Latent Structures) som gjør at dataen vår kan skrives på formen

$$X = TP^T + E \tag{49}$$

$$Y = UQ^T + F \tag{50}$$

Her har P og T samme tolkning som i PCA anvendt på X (ladning og score), og U og Q forstås på tilsvarende vis i rommet Y lever i. E og F er feil, som antas uavhengige med identisk fordeling. Ved hjelp av en lur måte å gjøre iterasjoner på når man regner ut disse dekomposisjonene, finner man T og U med stor kovarians. Pseudokode for dette, og flere refleksjoner rundt PLS kommer her om jeg får tid.