TTK4260 Innføring i Multivariat Datamodellering

Kristian Løvland

Innhold

1 Introduksjon

Dette dokumentet er et forsøk på å skaffe meg en strukturert oversikt over emnet TTK4260 Innføring i Multivariat Datamodellering. Ting er først og fremst hentet fra slides, men også noe fra [2] og [1].

2 Grunnleggende statistikk

Før vi tar fatt på multivariat dataanalyse er det greit å gå gjennom noen viktige temaer fra univariat dataanalyse. Generelt for alt jeg skriver i disse notatene er at vi ikke bryr oss noe særlig om hva slags sannsynlighetsfordelinger variabler har (dvs. at vi stort sett antar at ting er normalfordelt). Vi er mest interessert i å, gitt inputdata u og outputdata y, finne en modell

$$y = f(u;\theta) \tag{1}$$

der θ er parametre, som forklarer dataen vår på en god måte. I tillegg kommer det et par halvrelaterte temaer og noen digresjoner.

2.1 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. En minste kvadraters estimator av en parameter θ gitt et datasett $\mathcal{D}$ og en modell $f(u_t;\theta)$, er da gitt av

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

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

$$y_t = \sum_{j=1}^{n} \theta_j \phi_j(u_t) + e_t \tag{3}$$

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

$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} \phi_1(u_1) & \cdots & \phi_n(u_1) \\ \vdots & & \vdots \\ \phi_1(u_N) & \cdots & \phi_n(u_N) \end{bmatrix} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} + \begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix} \tag{4}$$

Som mer kompakt kan skrives som

$$\boldsymbol{y} = \Phi(\boldsymbol{u})\boldsymbol{\theta} + \boldsymbol{e} \tag{5}$$

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

$$\widehat{\theta}_{\mathrm{LS}} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^n} \|\boldsymbol{y} - \Phi(\boldsymbol{u})\boldsymbol{\theta}\|^2 \tag{6}$$

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

$$\Phi(\boldsymbol{u})^T\Phi(\boldsymbol{u})\widehat{\theta}_{\mathrm{LS}} = \Phi(\boldsymbol{u})^T\boldsymbol{y} \tag{7}$$

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 (den gir nærmeste løsning hvis ingen løsning eksisterer, løsning med minst norm hvis mange løsninger eksisterer).

Hvis man vil vektlegge minimeringen av noen tilstander mer enn andre, kan man gjøre dette med en vektmatrise W slik at $\theta = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^n} \|\boldsymbol{y} - \Phi(\boldsymbol{u})\boldsymbol{\theta}\|^2_{W^{-1}}$. 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.

2.2 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

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

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 med moteksempler, men under realistiske antagelser eksisterer stort sett et ML-estimat.

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. Man kan utlede forskjellige estimatorer avhengig av p, men som nevnt kommer det meste av estimatorer vi jobber med til å basere seg på antagelsen om at verden er normalfordelt.

2.3 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".

Vi sier nå farvel til Bayesisk statistikk.

2.4 Statistiske ytelsesindekser

Vi går kort gjennom noen statistiske ytelsesindekser (dvs. for ytelsen til en estimator) for tre ulike typer problemstilling.

2.4.1 Regresjonsproblemer

En mye brukt 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".

2.4.2 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**). Det finnes mange mer eller mindre naturlige måter å vurdere om en estimator sine ja/nei-svar er gode på:

- **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**

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

2.4.3 Sammenligning av sannsynlighetsfordelinger

I utledning av ulike former for estimatorer eller ytelsesindekser ønsker man gjerne å basere seg på statistisk teori enn å bruke "sunn fornuft". Et mye brukt mål på likheten mellom to sannsynlighetsfordelinger er **Kullback-Leibler-divergens**. Denne er utledet med utgangspunkt i informasjonsteori. Gitt sannsynlighetsfordelinger p_0 og p_n og data $\mathcal{D}$, forteller KL-divergensen hvor mye informasjon som tapes på å representere p_0 med p_n (dette er ikke symmetrisk), og kan dermed brukes som mål på hvor "forskjellige" fordelingene er. Den er gitt av

$$KL\left(p_0, p_n\right) := \int \log\left(\frac{p_0\left(\mathcal{D}, \theta_0\right)}{p_n\left(\mathcal{D}, \theta_n\right)}\right) p_0\left(\mathcal{D}, \theta_0\right) d\mathcal{D} \tag{25}$$

2.5 Bias vs. varians

Dette er en avveining man ikke slipper unna når man bedriver estimering. La oss bruke MSE for å illustrere dette. Anta at vi estimerer en parameter θ med $\hat{\theta}$. La

$$\begin{aligned} \mathcal{V} &:= \widehat{\theta} - \mathbb{E}[\widehat{\theta}] \\ \mathcal{B} &:= \mathbb{E}[\widehat{\theta}] - \theta \end{aligned} \tag{26}$$

Da er

$$\begin{aligned} \mathbb{E}\left[\|\hat{\theta} - \theta\|^2\right] &= \mathbb{E}\left[\|\widehat{\theta} - \mathbb{E}[\widehat{\theta}] + \mathbb{E}[\widehat{\theta}] - \theta\|^2\right] \\ &= \mathbb{E}\left[\|\mathcal{V} + \mathcal{B}\|^2\right] \\ &= \mathbb{E}\left[(\mathcal{V} + \mathcal{B})^T(\mathcal{V} + \mathcal{B})\right] \\ &= \mathbb{E}\left[\|\mathcal{V}\|^2 + \|\mathcal{B}\|^2 + 2\mathcal{V}^T\mathcal{B}\right] \\ &= \mathbb{E}\left[\|\mathcal{V}\|^2\right] + \|\mathcal{B}\|^2 \end{aligned} \tag{27}$$

Dvs. at en estimator sin forventede feil vil bestå både av et bias-ledd (systematisk feil) og et varians-ledd. For å oppnå et godt estimat må begge disse minimeres, men å minimere bias og varians er typisk motstridende interesser, og man må gjøre vurderinger av datasett og estimeringsmetodikk for å finne en god avveining.

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 verktøy som kan brukes som hjelp til å ta beslutninger om slike ting, som vi kommer tilbake til senere.

3 Multivariat Dataanalyse

Dette kapitlet utgjør mesteparten av pensum i emnet. Jeg skal prøve å gå gjennom det nogenlunde systematisk.

3.1 Eksperimentdesign

Selv om det ikke nødvendigvis er en del av arbeidsprosessen for alle dataanalytikere, er innhenting av data en viktig del av prosessen som må gjøres på riktig måte for at man skal kunne gjøre meningsfylt analyse. **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

$$\Delta y = \frac{\sum y_+}{n_+} - \frac{\sum y_-}{n_-} \tag{28}$$

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

$$\text{Statistisk styrke} = (1 - \beta) \cdot 100\% \tag{29}$$

Denne kan tolkes som sannsynligheten for å "avsløre" en effekt av størrelse δ, 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

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_k x_k + f \tag{30}$$

Den beste tilpasningen av en slik modell får man ved å minimere summen av kvadratavviket mellom modellprediksjon og faktisk data, som forklart i avsnitt 2.1. 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

$$SS_{total} = SS_{model} + SS_{error} \tag{31}$$

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.

Man kan også sette opp en ANOVA-tabell for enkeltledd i modellen, hver av disse vil ha samme struktur som øverste rad i tabell 1.

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

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.

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

$$\text{Proj}_{\mathbf{x}}\mathbf{y} = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{|\mathbf{x}|^2}\mathbf{x} \tag{32}$$

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. Den geometriske tolkningen ligner på tolkningen av lineær regresjon, der man finner en linje som passer en emngde punkter. I PCA kan man imidlertid legge til dimensjoner slik at man tilpasser et plan, eller flater av enda høyere dimensjon.

For å forstå PCA kan det være nyttif å forstå **Singulørverdidekomposisjonen (SVD)**, og den gemoetriske tolkningen av denne. 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 1.

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

$$A\mathbf{v}_j = \sigma_j \mathbf{u}_j \tag{33}$$

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

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

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

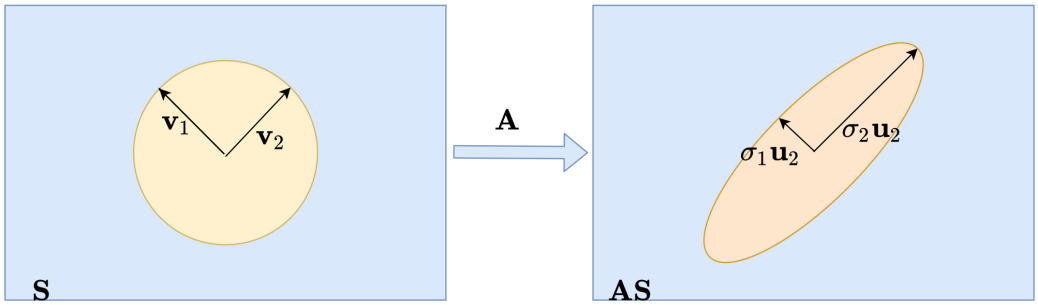

Fig. 1: Geometrisk tolkning av lineærtransformasjon

$$A = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \tag{35}$$

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 2 der A er en sum av vektorytreprodukter (som er matriser med rang 1). 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 😎.

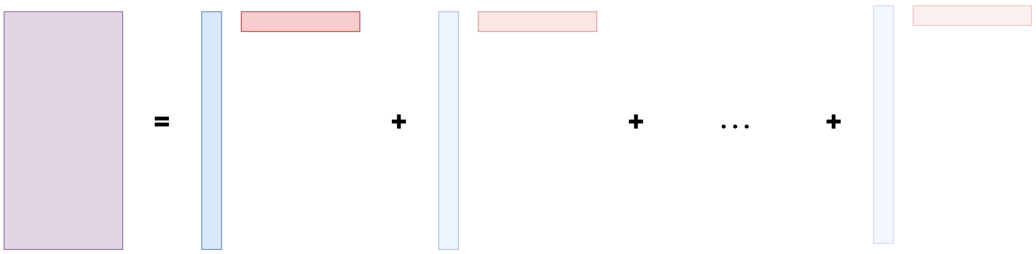

Fig. 2: Tolkning av SVD

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

$$\max(\phi) = \boldsymbol{p}_1^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{p}_1 - \lambda \left(\boldsymbol{p}_1^T \boldsymbol{p}_1 - 1 \right) \tag{36}$$

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

$$\mathbf{X}^T \mathbf{X} \mathbf{p}_1 = \lambda_1 \mathbf{p}_1 \tag{37}$$

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

$$\mathbf{X} = \mathbf{t}_1 \mathbf{p}_1^T + \mathbf{t}_2 \mathbf{p}_2^T + \cdots + \mathbf{E} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \mathbf{E} \tag{38}$$

3.2.4 NIPALS-algoritmen

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. Pseudokoden under viser algoritmen. Merk at projeksjon av en matrise X på en vektor t etterfulgt av en normalisering mer kompakt kan skrives som

$$\frac{\text{proj}_t(X)}{||\text{proj}_t(X)||_2} = \frac{X^T t / t^T t}{||X^T t / t^T t||_2} = \frac{X^T t / t^T t}{\sqrt{(X^T t)^T (X^T t)/(t^T t)^2}} = \frac{X^T t}{||X^T t||_2} \tag{39}$$

Hovedideen bak NIPALS er å utlede ytreproduktene $t_i p_i^T$ én og én, med høyeste varians først. Disse vektorene finnes ved å iterativt tilpasse p til t vha. X, t til p vha. X, frem og tilbake helt til resultatet stabiliserer seg (dette kalles gjerne "criss-cross regressions"). Pseudokode for NIPALS er som følger (merk at man antar at X allerede er sentrert).

Algoritme 1 NIPALS for PCA

prosedyre NIPALS($X, n_{\text{PCA}}, t_{\text{tol}}$) ▷ Numerisk utregning av PCA
    $i \leftarrow 1$
    mens $i < n_{\text{PCA}}$ **gjør**
        Initialisér t_i (tilfeldig, eller som en av kolonnene i X)
        mens Endring i $t_i > t_{\text{tol}}$ **gjør**
            $p_i \leftarrow \frac{X^T t_i}{||X^T t_i||_2}$ ▷ Projeksjon av X på t_i
            $t_i \leftarrow X p_i$ ▷ Projeksjon av X på p_i
        slutt mens
        $X \leftarrow X - t_i p_i^T$ ▷ Deflasjon
        $i \leftarrow i + 1$
    slutt mens
slutt prosedyre

3.2.5 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 ICA

PCA gir en god og forståelig måte å finne struktur i komplisert data på, men som alle andre metoder gjør den noen antagelser som ikke alltid er gyldige. En av disse er at dataen kommer fra én enkelt kilde. Men hva om man vil analysere en enkelt stemme fra en gruppe på flere mennesker som snakker sammen (cocktail party-problemet)? Eller om man vil (annet eksempel)? Dette er motivasjonen for **Independent Component Analysis (ICA)**.

Problemet kan formuleres mer presist. Gitt en lineærekombinasjon (nok en antagelse, men jaja) av N distinkte signaler, dvs. hver x_j i datasettet er gitt av

$$x_j = a_{j1}s_1 + a_{j2}s_2 + \cdots + a_{jN}s_N \tag{40}$$

er målet å finne de originale kildene til dataen vår, dvs. $s_1, s_2, \ldots, s_N$. På matriseform kan dette skrives som

$$x = As \tag{41}$$

Så da bare inverterer vi A og setter $s = A^{-1}x$. Neida, det er nok ikke så enkelt. Jeg bare tulla litt.

Målet vårt er å finne s under så slappe antagelser som mulig. Vi antar at hvert par s_i, s_j er statistisk uavhengige. Ellers ville det ikke vært mulig å skille hvilken årsak som førte til hvilken effekt.

Utgangspunktet for metoden er igjen singulærverdidekomposisjonen, altså transformasjonen

$$A = U\Sigma V^* \tag{42}$$

der U, V^* er unitære matriser (tenkes på som rotasjoner) og Σ er diagonal (altså er den en skalering). Siden $x = As$ ville vi med kjennskap til A kunne finne s ved å anvende U^*, Σ^{-1} og V etter hverandre i den rekkefølge, og da få en approksimasjon av $s = (U\Sigma V^*)^{-1}x = V\Sigma^{-1}U^*x$. Dessverre kjenner vi ikke A. Beväpnet med tolkningene av U, Σ og V^* som vi har fra den tidligere diskusjonen av SVD kan vi imidlertid gjøre et forsøk. ICA *kan* rettferdiggjøres vha. informasjonsteori, som en måte å minimere den felles informasjonen mellom s_1 og s_2. Her holder vi oss imidlertid til den enklere, geometriske tolkningen.

For å tilnærme U^* fra data, kan man simpelthen regne ut retningen i dataen som minimerer varians. Ved å la U^* være en matrise parametrisert av vinkelen θ kan man minimere variansen til dataen x mhp. θ, og la U^* være rotasjonsmatrisen som roterer med denne vinkelen, gjerne kalt θ_0. Deretter finner man Σ^{-1} ved å regne ut variansen langs hver retning i (det nye) koordinatsystemet. Den mest utfordrende delen av ICA kommer nå: Vi har allerede brukt middelverdi og varians, så vi trenger ny informasjon som ikke er avhengig av noen av disse, dvs. at vi trenger høyere ordens momenter. Derfor må vi også anta at s ikke er normalfordelt (for da vil alle momenter av orden over to være null). Det finnes ulike måter å rettferdiggjøre de ulike metodene man kan bruke for å finne V, men én metode som er forholdsvis enkel er å maksimere fjerde ordens moment, **kurtose**. Denne henger sammen med ikke-gaussiskhet, som ønskes maksimert for å best mulig skille datakildene fra hverandre (dette kan også rettferdiggjøres bedre enn jeg gjør det her). Siden U^* representerer en rotasjon, parametriserer vi denne ved variabelen ϕ, og løser $\frac{d\text{Kurt}}{d\phi} = 0$. Løsningen, som vi kaller ϕ_0, er rotasjonen U^* beskriver. Vi har nå alt vi trenger for å finne

$$s = V\Sigma^{-1}U^*x \tag{43}$$

For spesialtilfellet der $N = 2$ og $x \in \mathbb{R}^2$ vil

$$s = \begin{bmatrix} \cos\phi_o & \sin\phi_o \\ -\sin\phi_o & \cos\phi_o \end{bmatrix} \begin{bmatrix} 1/\sqrt{\sigma_1} & 0 \\ 0 & 1/\sqrt{\sigma_2} \end{bmatrix} \begin{bmatrix} \cos\theta_o & \sin\theta_o \\ -\sin\theta_o & \cos\theta_o \end{bmatrix} x \tag{44}$$

3.4 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

$$\hat{\theta}_{\text{OLS}} = (X^T X)^{-1} X^T y \tag{45}$$

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 et 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

$$y = \hat{\theta}_{\text{PCR}} p \tag{46}$$

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 enn å gjøre som man vanligvis gjør i PCA. Om man er så heldig å ha tilgang til et testsett vil man også kunne forvente at forklart y-varians vil ha et maksimum, som gjør det enklere å velge antall PC-er.

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

$$\sum_{i=0}^{n}(y_i - f(x_i))^2 + \lambda R(f) \tag{47}$$

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** (ev. L_2-regularisering), hvor

$$R(f(x)) = R(\theta_1 x_1 + \cdots + \theta_n x_n) = \sum_{i=1}^{n} \theta_i^2 \tag{48}$$

og en metode som enda hardere driver små θ_i til å bli eksakt 0, **LASSO** (ev. L_1-regularisering), der

$$R(f(x)) = R(\theta_1 x_1 + \cdots + \theta_n x_n) = \sum_{i=1}^{n} |\theta_i| \tag{49}$$

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

Som en fun fact kan det nevnes at både L_2- og L_1-regularisering kan utledes fra Bayesisk statistikk. Om man antar at feilen i θ er normalfordelt får man en posterior fordeling med mode lik L_2-estimatet, mens om man antar en Laplacefordeling vil moden være lik L_1-estimatet.

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, og jeg kommer aldri til å nevne dette igjen.

3.5 PLS

PCR har en åpenbar svakhet: Metoden baserer seg på antagelsen om at de prinsipale komponentene, kun utledet fra x, også er egnet til å forklare 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

$$X = TP^T + E \tag{50}$$

$$Y = UQ^T + F \tag{51}$$

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 feilmatriser, som antas uavhengige med identisk fordeling. Akkurat hva retningene kolonnene i T og U beskriver er litt vanskeligere å beskrive enn for PCA, men overordnet kan man tenke på det som at de ligner score-matrisene utregnet i PCA, men at man her også sørger for at T og U har stor kovarians (T forklarer U.

For en dypere forståelse av betydningen kan det være nyttig å undersøke NIPALS-algoritmen for PLS, beskrevet under. Denne har mange likhetstrekk med den enklere NIPALS-algoritmen som finner PCA, men regner nødvendigvis ut flere matriser (alt gjøres parallelt). Som tidligere antar man at X og Y er sentrert.

Algoritme 2 NIPALS for PLS

prosedyre NIPALS($X, Y, n_{\text{PLS}}, t_{\text{tol}}$) ▷ Numerisk utregning av PLS
  $i \leftarrow 1$
  mens $i < n_{\text{PLS}}$ **gjør**
    Initialisér u_i (tilfeldig, eller som en av kolonnene i Y)
    mens Endring i $t_i > t_{\text{tol}}$ **gjør**
      $w_i \leftarrow \frac{X^T u_i}{||X^T u_i||_2}$ ▷ Projeksjon av X på u_i
      $t_i \leftarrow X w_i$ ▷ Projeksjon av X på w_i
      $q_i \leftarrow \frac{Y^T t_i}{||Y^T t_i||_2}$ ▷ Projeksjon av Y på t_i
      $u_i \leftarrow Y q_i$ ▷ Projeksjon av Y på q_i
    slutt mens
    $p_f \leftarrow \frac{X^T t_f}{t_f^T t_f}$
    $b_f \leftarrow \frac{u_f^T t_f}{t_f^T t_f}$
    $X \leftarrow X - t_i p_i^T$ ▷ Deflasjon av X
    $Y \leftarrow Y - b_i^T q_i^T$ ▷ Deflasjon av Y
    $i \leftarrow i + 1$
  slutt mens
slutt prosedyre

Det er mye som kan sies om matrisene som returneres, men en av de viktigste plottene å undersøke er plottet av t vs u. Dette representerer den lineære regresjonen mellom de to nye koordinatsystemene PLS definerer for X og Y, og forteller noe om hvordan komponentene i X forklarer komponentene i Y. Det er også verdt å merke seg at vi har definert en ny matrise, W, som kalles X-**loading weights**. Det er denne som plottes (i stedet for P) når

man tolker PLS-regresjon, siden det er denne som maksimerer kovariansen mellom t og u (motivasjonen for PLS).

3.6 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.

3.6.1 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.

3.6.2 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):

$$\mathbb{E}[(\mu - \hat{\mu})^2] = \cdots = -n\sigma^2 + \mathbb{E}[(y-\mu)^2] + \underbrace{2\text{cov}(y, \mu)}_{\text{trøbbel}} \tag{52}$$

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?

3.6.3 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

$$\mathrm{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \mathrm{MSE}_i \tag{53}$$

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.

3.6.4 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 treningsdata).

3.6.5 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 tidligere nevnt. Denne er gitt av

$$KL\left(p_0, p_n\right) := \int \log\left(\frac{p_0\left(\mathcal{D}, \theta_0\right)}{p_n\left(\mathcal{D}, \theta_n\right)}\right) p_0\left(\mathcal{D}, \theta_0\right) d\mathcal{D} \tag{54}$$

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.)

$$\mathrm{AIC} = \frac{1}{N}\sum_{t=1}^{N} \ell(\varepsilon(t, \theta_n)) + \frac{1}{N}\dim(\theta_n) \tag{55}$$

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**)

$$\mathrm{BIC} = \frac{1}{N}\sum_{t=1}^{N} \ell(\varepsilon(t, \theta_n)) + \frac{\log N}{2N}\dim(\theta_n) \tag{56}$$

3.6.6 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

$$\widehat{\text{bias}}(\widehat{\theta})_{\text{jk}} := \frac{n-1}{n} \sum_{i=1}^{n} \left(\widehat{\theta}_{[i]} - \widehat{\theta} \right) \quad \text{var}(\widehat{\theta})_{\text{jk}} := \frac{n-1}{n} \sum_{i=1}^{n} \left(\widehat{\theta}_{[i]} - \widehat{\theta}_{\text{ave}} \right)^2 \quad (57)$$

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 57.

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

3.7 Utliggerdeteksjon

Om et måleinstrument er upresist kan vi anta at målesignalene er utsatt for Gaussisk støy, og likevel få nyttig informasjon fra hver observasjon $y_i = f(x_i) + \varepsilon_i$. Det er imidlertid ikke alle feil som er fullt så hyggelige. En feil inntasting av data er vanskelig å modellere som noe annet enn feil som ideelt sett burde fjernes, og korrekte målinger av ekstreme tilfeller kan gjøre mer skade enn nytte når man vil ha en modell som stort sett fungerer. Disse to tilfellene ville antagelig hatt hhv. høy *residual* og *leverage*, vist i figur 3.

La T være en matrise med scores slik at $X \approx TP^T$. Et mål på "utliggerhet" er bør da si noe om avstand til "resten av dataen". Noen slike verdier følger.

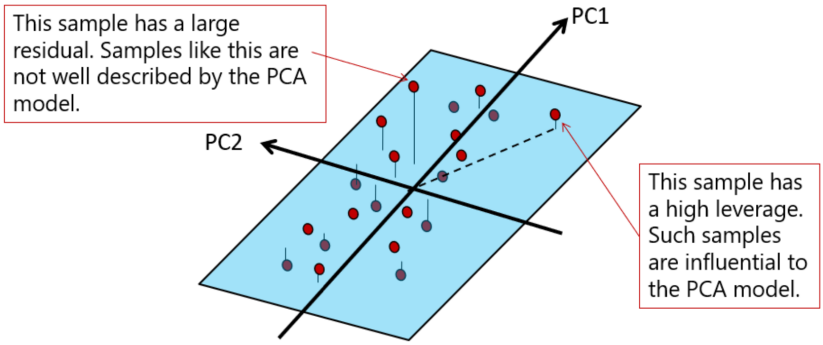

Fig. 3: Ulike typer utliggere

3.7.1 Hotellings T^2

Hotellings T^2 representerer avstanden til modellsenter i modellrommet (definert av T). For en observasjon y_n er

$$T_n^2 = \sum_{a=1}^{A} (t_{na}(T_a^T T_a)^{-1} t_{na}^T)(n-1) \tag{58}$$

T^2 er F-fordelt, og kan dermed brukes til å finne ut om y_n kommer fra samme populasjon som resten av observasjonene (med et eller annet signifikansnivå). En typisk måte å visualisere dette på er som en ellipse i et score-plott.

3.7.2 Leverage

For observasjonen y_n er **leverage** gitt av

$$h_n = \frac{1}{N} + t_{n,1:A}(T_{1:A}^T T)^{-1} t_{n,1:A}^T \tag{59}$$

For leverage brukes vanligvis ikke signifikans for å bestemme kritisk grense (dvs. hva som defineres som en utligger), men heller en definisjon som er ganske så ad-hoc: $h_{\text{kritisk}} = \frac{3(A+1)}{N}$.

Summen av h_n over alle n observasjoner er 1, så leverage kan sees på som andelen varians en observasjon bidrar med. Denne verdien er lineært avhengig av Hotellings T^2 gjennom formelen

$$T^2 = (N-1)(h - \frac{1}{N}) \tag{60}$$

3.7.3 F- og Q-Residualer

Som tidligere beskrevet er residualene for en modell TP^T tilpasser X gitt av

$$E_A = X - \sum_{a=1}^{A} t_a p_a^T \tag{61}$$

der A er antall PC-er brukt i modellen. For en enkelt observasjon x_{nk} er residualen (gitt samme modell) gitt av

$$e_{nk,A} = x_{nk} - \sum_{a=1}^{A} t_{na}(p^T)_{ak} \tag{62}$$

Disse gir ulike måter å representere residualene på. Om man er interessert i å undersøke enkeltobservasjoner (som gir mening siden utliggere jo er observasjoner) kan kritiske grenser for $e_{nk,A}$ estimeres vha. F- eller Q-fordelingen.

Q-residualen er gitt av

$$Q_{i,A} = e_{n,A}^T e_{n,A} \tag{63}$$

og om man ønsker å finne kritisk Q-verdi for signifikans α, kan man finne c_α i en tabell for standardnormalfordelingen, og bruke denne saftige formelen

$$Q_a = \theta_1 \left(\frac{c_a \sqrt{2\theta_2 h_0^2}}{\theta_1} + \frac{\theta_2 h_0 (h_0 - 1)}{\theta_1^2} + 1 \right)^{\frac{1}{h_0}} \tag{64}$$

der $\theta_1 = \text{trace}(E)$, $\theta_2 = \text{trace}(E^2)$, $\theta_3 = \text{trace}(E^3)$ og $h_0 = 1 - \frac{2\theta_1\theta_3}{3\theta_2^2}$.

F-residualen kan finnes vha. Q-residualen, og regnes som mer konservativ. Den er gitt av

$$F_{i,A} = \frac{Q_i}{K} \tag{65}$$

Når man kjenner denne kan man gjennomføre en vanlig F-test.

Om man ikke vil bruke formler og finne disse verdiene manuelt, kan man bruke de innebygde metodene for dette som antageligvis allerede eksisterer i `verkøytet` man bruker for dataanalyse.

3.7.4 Plott

Det finnes mange nyttige plott man kan bruke når man vil oppdage utliggere, typisk i form av scatterplot med grensene diskutert i dette avsnittet plottet som linjer eller kurver. Kan hende kommer det noen figurer her etter hvert, eller kanskje et lite eksempel.

4 Andre temaer

Her kommer noen temaer som ikke er en del av det mer standard pensumet om multivariat statistikk, men likevel er en del av emnet.

4.1 Tidsserier

Dette kapitlet er høyst relevant for alle som interesserer seg for dynamiske systemer.

4.1.1 Modellstruktur

Å jobbe med kontinuerlige systemer av ulineære differensiallikninger

$$\dot{x} = f(x, u) \tag{66}$$

har manger fordeler, men mulighet for å simulering og tilpassing av måledata er ikke en av dem. I den virkelige verden er vi typisk gitt tidsseriedata $X = [x_1 \quad \cdots \quad x_n]$, der hver x_i er en kolonnevektor som inneholder en tidsserie med målinger av tilstanden x_i. Om vi mistenker at et system har en bestemt struktur $f(x, u)$ kan vi linearisere

$$f(x) \approx f(x_0) + \nabla f_{x_0}(x - x_0) \tag{67}$$

og deretter diskretisere

$$\frac{df}{dt} \approx \frac{f(\Delta(k+1)) - f(\Delta k)}{\Delta} \tag{68}$$

Vi innfører notasjonen $y(\Delta k) = y(k)$ for systemer i diskret tid, og z-operatoren

$$z^r y(k) = y(k + r) \tag{69}$$

Med denne operatoren definert er vi klar til å skrive alle differenslikninger

$$y(t) + a_1 y(t-1) + \cdots + a_{n_a} y(t - n_a) = b_0 u(t) + \cdots + b_{n_b} u(t - n_b) + e(t) \tag{70}$$

som en transferfunksjon

$$y(t) = \frac{B(q)}{A(q)}u(t) + \frac{1}{A(q)}e(t) \tag{71}$$

der

$$A(q) = 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a} \tag{72}$$

$$B(q) = b_1 q^{-1} + \ldots + b_{n_b} q^{-n_b} \tag{73}$$

og vi av en eller annen grunn har byttet ut bokstaven z med q. Denne modellen er en såkalt **ARX-modell**. AR fordi den er autoregressiv (y har en form for dynamikk), X fordi den er eksogen (har en input).

Om man kjenner e kan man inkludere målinger av denne i modellen gjennom transferfunksjonen

$$C(q) = 1 + c_1 q^{-1} + \ldots + c_{n_c} q^{-n_c} \tag{74}$$

og få en **ARMAX**-modell (MA = Moving Average) på formen

$$y(t) = \frac{B(q)}{A(q)}u(t) + \frac{C(q)}{A(q)}e(t) \tag{75}$$

Disse modellene har noen restriksjoner, f.eks. at både u og e har samme nevner (dvs. utsettes for samme dynamikk). En mer generell modell er **Box-Jenkins**-modellen, der

$$y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t) \tag{76}$$

om man setter $C(q) = D(q)$ for man en **Output Error**-modell. Det kan imidlertid være problematisk med så generelle modeller, så det er verdt å gjøre en vurdering av hvor komplisert modellstruktur man egentlig har behov for.

4.1.2 Ettstegsprediktorer

Heretter vil vi anta at vi har gjort et valg av modellstruktur, og jobber med et system på formen

$$y(t) = G(q)u(t) + H(q)e(t) \tag{77}$$

hvor vi antar at $H(q)$ er monisk og minimum fase (dette blir nyttig senere).

Et naturlig ønske vil være å finne ut hvor systemet vil ende opp ved neste tidssteg gitt nåværende tilstand, dvs. finne en ettstegsprediktor $\hat{y}(t|t-1)$. Siden $u(t)$ velges av oss er den kjent, men $e(t)$ er ikke det. Derfor må effekten av denne estimeres. La $v(t) = H(q)e(t)$. Om vi klarer å finne en ettstegsprediktor $\hat{v}(t|t-1)$ kan vi enkelt finne $\hat{y}(t|t-1) = G(q)u(t)+\hat{v}(t|t-1)$. Men hvordan kan vi finne denne prediktoren? Jo, bare se her

$$v(t) = H(q)e(t) = \sum_{k=0}^{+\infty} h(k)e(t-k) = e(t) + \sum_{k=1}^{+\infty} h(k)e(t-k) \tag{78}$$

$$\implies \quad \widehat{v}(t|t-1) = \sum_{k=1}^{+\infty} h(k)e(t-k) \tag{79}$$

fordi vi får et best mulig estimat ved å anta $e(t) = \mu_e = 0$, tror jeg. Skrevet om på transferfunksjonform får vi

$$\hat{v}(t|t-1) = [H(q) - 1]e(t) = [1 - H^{-1}(q)]v(t) \tag{80}$$

altså er det av interesse å finne $H^{-1}(q)$. Ved hjelp av litt utregning kan man da finne ut praktisk uttrykk for tilstandsprediktoren $\hat{y}$, gitt av

$$\hat{y}(t|t-1) = H^{-1}(q)G(q)u(t) + [1 - H^{-1}(q)]y(t) \tag{81}$$

Merk at siden vi antar $H(q)$ monisk, så inneholder leddet $[1 - H^{-1}(q)]$ ingen konstantledd, så prediktoren er ikke avhenig av kjennskap til $y(t)$. Vi har antatt $H(q)$ minimum fase, så impulsresponsen dens eksisterer og gir oss en veldefinert prediksjon. Nærmere bestemt er

$$\left[H^{-1}(q)G(q)\right] u(t) = \left[\sum_{k=1}^{+\infty} \ell(k)q^{-k}\right] u(t) \tag{82}$$

og

$$\left[1 - H^{-1}(q)\right] y(t) = \left[-\sum_{k=1}^{+\infty} \widetilde{h}(k)q^{-k}\right] y(t) \tag{83}$$

I praksis vil vi lette på kravet om at summene skal gå til ∞, og heller la dem gå til t. Forhåpentligvis vil impulsresponsen avta raskt og eksponensielt nok til at prediksjonen blir nokså nærme sannheten.

Merk at hittil har vi ikke utnyttet noen informasjon om de statistiske egenskapene til $e(t)$. Det skal vi heller ikke gjøre, men en liten fun fact er at om $e(t)$ er Gaussisk, er den optimale $\hat{y}(t|\ t-1)$ gitt av Kalmanfilteret.

4.1.3 Systemidentifikasjon

Nå står vi bare igjen med et lite spørsmål: Hvordan finner vi $H^{-1}(q)$ og alle dens venner, som kreves for å gjennomføre disse prediksjonene? Dette gjøres ganske rett frem ved å minimere en eller annen loss-funksjon $\ell(y-\hat{y})$. Man velger da gjerne en mengde modellstrukturer M_j med tilhørende parametre θ_j gitt av

$$\theta_j^* = \min_{\theta_j \in \Theta_j} \sum_t \ell\left(-H^{-1}(q)G(q)u(t) + H^{-1}(q)y(t)\right) \tag{84}$$

og velger modellstruktur basert på et eller annet kriterie (AIC, BIC, eller noe annet). Ved å sette inn verdiene for $H(q)$ og $G(q)$ gitt i avsnittet om modellstrukturer i likning 81.

Merk at det finnes massevis av praktiske hensyn som burde tas, som ikke er nevnt her. Et av de viktigste er persistent eksitasjon, som handler om hvor mye informasjon den gitte inputen gir om systemets dynamikk. For eksempel vil ikke et pådrag $u(t) = 0$ gi særlig mye informasjon om $G(q)$. Om $G(q)$ er komplisert vil $u(t)$ typisk også måtte være nokså komplisert for å få vite alt man trenger om $G(q)$. Et typisk krav for mange parameterestimeringsmetoder er at $u(t)$ må oppfylle at det eksisterer konstanter $\alpha_0, \alpha_1, T_0 > 0$ slik at

$$\alpha_0 I \leq \frac{1}{T_0}\int_t^{t+T_0} u(\tau)u(\tau)^T d\tau \leq \alpha_1 I \quad \text{for alle } t \leq 0 \tag{85}$$

$u(t)$ kalles da persistent eksiterende.

4.2 Nevrale nettverk

Vi ønsker å approksimere en funksjon

$$y = f(x) \tag{86}$$

der $y \in \mathbb{R}^{n_y}$ og $x \in \mathbb{R}^{n_x}$. Vi kan gjøre vanlig lineær regresjon, men hva om f er ulineær, eller y består av kategorisk data?

Definér en **dybde** n. For $i = 1, 2, \ldots, n$, definer **bredden** n_i. La $n_0 = n_x$. La $W^{(i-1)} \in \mathbb{R}^{n_i \times n_{i-1}}$ for $i = 1, 2, \ldots, n$. For hver i, definér **aktiveringsfunksjonen** σ_i, og for en vektor $v = [v_1 \quad \cdots \quad v_n]^T$, skriv $\sigma(v) = [\sigma(v_1) \quad \cdots \quad \sigma(v_n)]^T$. La $a^{(0)} = x$, og $a^{(i)} = \sigma_i(W^{(i-1)}a^{(i-1)})$ for $i = 1, 2, \ldots, n$. La $\hat{y} = a^{(n)}$. Finn parametre $W^{(i)}$ for alle i som minimerer avviket mellom $\hat{y}$ og y for datasettet ditt. Dette er et nevralt nettverk. Hver vektor $a^{(i)}$ illustreres gjerne som en kolonne av noder, med input og output mellom hver kolonne (som er en vektorvaluert funksjon) tegnet opp som streker. Denne grafen brukes gjerne heller enn den tråkige notasjonen over for å illsutrere hva slags nettverk man bruker.

Dette var ingen god forklaring, men det var en veldig kompakt oppsummering av hvordan det fungerer. Valget av n, σ og alle n_i er en helt egen vitenskap, det samme er hvordan man basert på disse finner W på en effektiv måte. Uansett hva slags problem man vil bruke et nevralt nettverk til å løse, er det viktig å velge et som har en struktur som godt representerer den faktiske prosessen som generer y, har jeg hørt. Mer kommer jeg ikke til å si om den saken.

References

[1] Kim H. Esbensen, Brad Swarbrick, and Frank Westad. *Multivariate Data Analysis - An Introduction to Multivariate Analysis*. CAMO Software AS, 2018. URL: https://www.amazon.co.uk/Multivariate-Data-Analysis-introduction-Analytical/dp/826911040X.

[2] Gareth James et al. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. URL: https://faculty.marshall.usc.edu/gareth-james/ISL/.