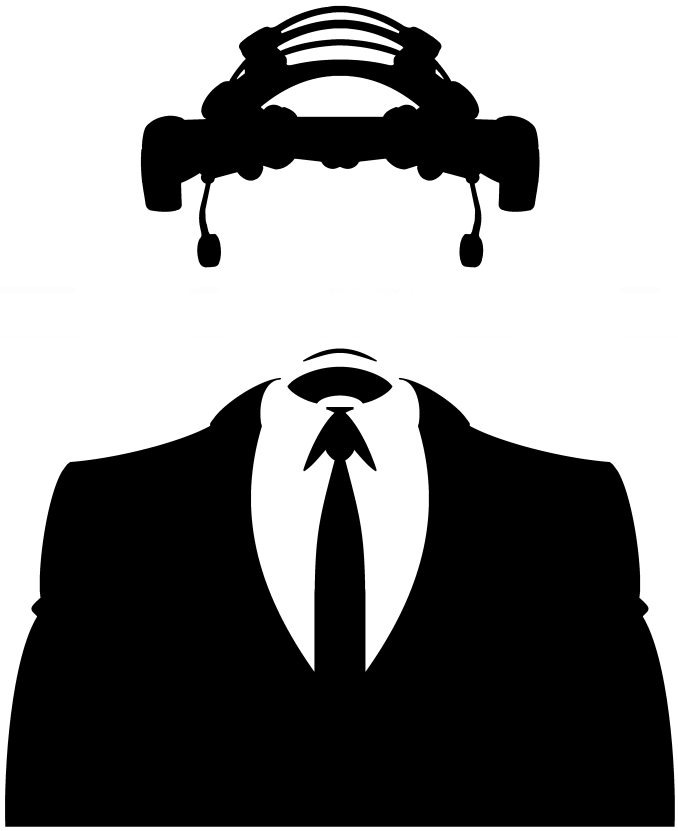
Klassificering af BCI-data fra consumer-grade hardware

ved brug af Riemannsk geometri



DAT5 Projekt

Gruppe d506e13

Institut for Datalogi
Aalborg Universitet

AALBORG UNIVERSITET
STUDENTERRAPPORT

Institut for Datalogi
Aalborg Universitet
Selma Lagerlöfs Vej 300
9220 Aalborg Ø
www.cs.aau.dk

Titel
Klassificering af BCI-data fra consumer-grade hardware ved brug af Riemannsk geometri

Tema
Maskinindlæring

Projektperiode
Efteråret 2013

Projektgruppe
d506e13

Gruppemedlemmer
Lasse Carstensen
Henrik Haxholm
Daniel Hillerström
Mathias Ruggard Pedersen
Michael Stisen

Afsluttet
20. December 2013

Vejleder
Bo Thiesson

Oplagstal: 7

Sideantal: 71

Synopsis

Denne rapport beskriver et system udviklet til at bruge et consumer-grade EEG-headset til at hjælpe med genoptræning. Foruden at undersøge om det er muligt, at klassificere om en bruger tænker på at bevæge højre eller venstre arm med en klassificeringsalgoritme, der går ud fra den Euklidiske distance mellem målingere, laves der også forsøg der går ud fra den Riemannske distance for, at undersøge om det giver forbedringer.
Rapporten beskriver eksperimenter med et datasæt taget med et Emotiv EEG-headset og et BCI-competition datasæt, som begge klassificeres med de to klassificeringsmetrikker for først, at teste algoritmernes effektivitet under kliniske forhold og derefter om deres fejlklassificeringer stiger kraftigt ved brug af consumer-grade EEG-hardware.
Der kan ud fra testdataen konkluderes, at der klassificeres bedre på dataen, når den Riemannske metrik bliver brugt, i forhold til når den Euklidiske metrik bliver brugt. Selvom der kun er optaget en begrænset mængde data med Emotivs headset, kan der ses en tendens til, at testpersonerne har bedre resultater for datasæt, hvor de udfører bevægelsen, end datasæt hvor de tænker på bevægelsen.

Forord

Denne rapport er udarbejdet på baggrund af et 5. semesters studenterprojekt på datalogiuddannelsen ved Aalborg Universitet. Kildekoden til projektet m.m. er tilgængelig på http://www.dhil.net/public/edu/aau/d506e13/.

I rapporten vil vi, medmindre andet fremgår af teksten, benytte os af følgende navngivningskonvention for variable:

- Stokastiske variable skrives med stort X og Y.
- Matricer skrives med andre store bogstaver: $P, \Lambda, I, \ldots$
- Euklidske vektorer i skrives med fed: $\boldsymbol{u}, \boldsymbol{X}, \ldots$
- Abstrakte vektorer skrives med små bogstaver: u, v, w
- Skalarer skrives med små bogstaver: $a, k, i, \ldots$
- Mangfoldigheder skrives med $\mathcal{M}$.
- S betegner en symmetrisk matrix og P betegner en symmetrisk positivt definit matrix.

Lasse Carstensen

Henrik Haxholm

Daniel Hillerström

Mathias Ruggard Pedersen

Michael Stisen

Indhold

1

Problemanalyse

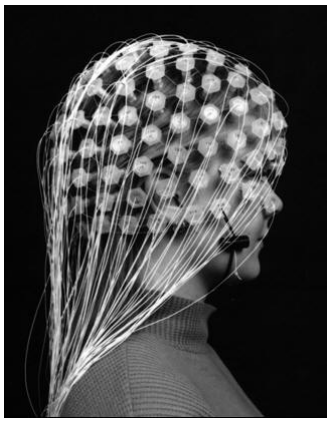
Mange mennesker har på et tidspunkt i deres liv brug for genoptræning af et eller flere af deres legemsdele. Ifølge Danmarks Statistik var der i 2010 ca. 138.368 genoptræningsplaner i Danmark, der tilsammen udgjorde en udgift på ca. 1,3 milliarder [Statistik, 2013, 2010]. I denne udgift er der ikke medregnet andre faktorer som for eksempel tabt arbejdskraft, så den samlede udgift for samfundet kan være endnu større. Der er derfor tale om et emne, som berører mange folk, og som har både følelsesmæssige og økonomiske omkostninger.

Behovet for genoptræning skyldes ofte enten apopleksi, også kaldet slagtilfælde, ulykker, længerevarende sygdomsforløb eller kirurgiske indgreb [Hjernesagen, 2009; Social-, Børne-, og Integrationsministeriet]. Genoptræning kan klassificeres på mange måder, alt efter hvad der er fokus på, og hver klassifikation har igen mange værktøjer til at forsøge at afhjælpe problemet. Der er stor forskel på hvordan problemer med mobilitet, muskelstyrke, balance, gang og fysisk kondition afhjælpes [Rodrigues-de-Paula and Lima, 2010].

Der findes to forskellige former for genoptræning, assistiv, som bruger værktøjer til at hjælpe bruger med at bevæge sig og restorativ, som fokuserer på at genoptræne hjernen. Genoptræning af førlighed fokuserer lige nu hovedsageligt på de fysiske aspekter for at gøre det muligt fx at kunne bevæge sine arme ordentligt igen efter en ulykke, hvor det meste af førligheden er tabt. Dette gøres ofte ved at have en terapeut eller maskine, som gradvist bruges mindre og mindre, til at hjælpe med bevægelsen. I disse situationer kan det ske, at det ikke så meget er et problem med armen som sådan, der gør at patienten ikke har fuld førlighed [Soekadar et al., 2011], men i stedet hjernen, som skal trænes til at sende de rigtige signaler for at flytte armen. Selvom det er hjernen man træner, er værktøjer, som holder øje med, hvad der sker i hjernen under processen, dog ikke udbredte på området. En mulig forbedring til genoptræning vil derfor være at prøve at benytte sig af en teknologi, der kan aflæse hjerneaktivitet, og bruge denne information til at hjælpe med genoptræningen.

1.1 Elektroencefalografi

Elektroencefalografi (EEG) er en teknik til at aflæse elektriske potentialer fra skalpen, som stammer fra hjernen [Nunez and Srinivasan, 2007]. Nuværende EEG-apparater findes både



Figur 1.1: Et EEG-apparat med 128 elektroder [Nunez and Srinivasan, 2007].

i invasive og ikke-invasive varianter. I ikke-invasive EEG-apparater foregår aflæsningen ved at placere et antal elektroder på testpersonens skalp, som hver især aflæser det elektriske potentiale på skalpen netop der, hvor de er påsat. Man vil derfor kunne opnå bedre rumlig opløsning ved at bruge flere elektroder. I invasive EEG-apparater er elektroderne placeret et stykke under skalpen, og elektroderne sidder derfor tættere på kilden til de elektriske potentialer, hvilket betyder at signalet er mindre diffust og kilden derfor kan bestemmes mere præcist end ved ikke-invasive EEG-apparater. Et eksempel på et ikke-invasivt EEG-apparat kan ses på figur 1.1. Generelt har EEG dog dårlig rumlig opløsning i forhold til andre hjerneaflæsningsmetoder som magnetisk-resonans-scanning, også kendt som MR-scanning, eller positron-emissions-tomografi-scanning, også kendt som PET-scanning, mens EEG generelt har bedre temporal opløsning end disse metoder [Nunez and Srinivasan, 2007].

EEG-apparater er også, modsat MR- og PET-scanning, tilgængelige i mere kompakte, brugervenlige og billigere udgaver, der gør det muligt at bruge EEG i langt flere sammenhænge end blot på hospitaler og i laboratorier.

Det elektriske potentiale, der kan måles på skalpen, er meget svagt, og målingerne er derfor meget følsomme overfor støj, både fra andre hjerneaktiviteter men også fra muskelaktiviteter i for eksempel panden og øjnene. I mange tilfælde kan bidraget fra støj være større end bidraget fra det signal, man er interesseret i at måle. For rent faktisk at kunne aflæse dette signal er det derfor meget vigtigt at kunne adskille støj fra signal. [Makeig et al., 2012]

1.2 Nuværende løsninger

Der eksisterer allerede flere løsninger, der forsøger at hjælpe patienter med genoptræning. Vi kan groft inddele de nuværende løsninger i to kategorier i forhold til genoptræning:

1. Genetablering af førligheden udelukkende gennem mental træning,
2. eller gennem mental træning med en form for fysisk assistance.

En af forudsætningerne for denne form for genoptræning er, at denne menneskelige hjerne er plastisk, og at det at forestille sig en bevægelse og det at udføre bevægelsen giver

nogenlunde samme udslag i hjernen [Grafton et al., 1996]. Tilsammen betyder dette at man kan genoptræne sin hjerne til at styre en legemsdel ved at tænke på at bevæge denne legemsdel.

I den første kategori tænker patienten på at bevæge den legemsdel, der har mistet føringen, hvorefter patienten får visuel feedback i form af en digital legemsdel, der bevæger sig. På denne måde aktiveres de relevante områder i hjernen, som på denne måde genoptrænes. [Pfurtscheller et al., 2008]

I den anden kategori understøttes genoptræningen af et eller flere fysiske redskaber. Et eksempel herpå kunne være en person, som har mistet førligheden i sin venstre arm (for eksemplets skyld antages det, at førligheden kan genvindes). Denne person har derfor en motordreven mekanisk arm eller andet fysisk hjælperedskab monteret rundt om sin arm. Den mekaniske arm udfører bevægelsen af venstre arm for personen, hvorimens personen tænker på at bevæge armen, og kan på denne måde forbinde bevægelsen med en tanke og altså genlære evnen til at bevæge armen. [Pfurtscheller et al., 2008] Alternativt kan det være en terapeut der hjælper med at bevæge armen i stedet for et mekanisk apparat.

Ud over disse to kategorier findes løsninger, der ikke hjælper med at genoptræne selve legemsdelen, men i stedet lader patienten styre andre apparater, som fx en mekanisk arm eller en computer, ved at tænke. Der findes løsninger, som aflæser hjerneaktivitet og forsøger at oversætte denne til bevægelser [Soekadar et al., 2011]. Disse bevægelser udføres således af en påmonteret arm eller af et computerprogram. Formålet med denne måde er at associere tanke med bevægelse – og ikke omvendt.

1.2.1 Forsøg med løsninger i Danmark

I Danmark har man også forsøgt sig på nye tiltag for at forbedre genoptræning. Ingen af tiltagene har inkluderet EEG, men er på områder, hvor EEG kan være en mulig del af løsningen.

Esbjerg Kommune har i 2012 afprøvet projektet "Virtuel Genoptræning i Døgnrehabilitering", som består af en all-in-one-PC og et tilhørende Kinect kamera [Rasmussen, 2012; Microsoft, 2013]. Kameraet er oprindeligt udviklet til Microsofts Xbox 360, som gør det muligt at bruge kroppen til at styre med. Denne funktion anvendes i projektet hvor brugeren følger et træningsprogram og bliver guidet igennem det på computeren mens de foran kameraet gentager de forskellige øvelser. Kammeraet registrer brugerens bevægelser og gør det muligt at vurdere udførelsen af øvelsen, mens der samtidig udføres en log, som en fysioterapeut kan holde øje med.

På Fyn afprøves nye rehabiliteringsmetoder som involvere robotter. I Odense Kommune har Syddansk Universitet og Odense Universitetshospital indgået et samarbejde med den japanske industrigigant Honda [WelfareTech, 2013]. Samarbejdsaftalen omhandler afprøvning af Hondas mobilitetsrobot "Walking Assist Device" (WAD), som skal fungere som et redskab til at forbedre en brugers evne til at gå. WAD spændes omkring hoften på brugeren, og foretager en analyse af brugerens gangart således at robotten kan hjælpe med at trække eller skubbe benet når behovet synes at være der.

Et andet projekt Odense Kommune har været med til, er afprøvning af den japanske robotvirksomhed Cyberdynes "HAL", som er en robot-styrkedragt [Pasgaard, 2011].

Cyberdyne aflæser ved hjælp af sensorer de elektroniske signaler som løber igennem musklerne i kroppen, og oversætter signalerne til bevægelse, og styrker på denne måde brugerens nedsatte funktionsevner.

1.3 Problemformulering

Fælles for alle disse løsninger, vi har undersøgt er, at det er dyre industriløsninger. Vi er ikke stødt på anvendelse af consumer-produkter i nogen af løsningerne. Der er ellers et interessant økonomisk perspektiv i at anvende consumer-produkter til genoptræning. Consumer-produkter er billigere end industriprodukter, og dermed også af en anden kvalitet. Eksempelvis kunne en løsning baseret på EEG consumer-produkter muliggøre, at patienter kan genoptrænes i eget hjem, hvilket kan være med til at sænke antallet af nødvendige menneskelige resurser per patient.

Efter at have beskrevet nogle mulige forbedringer til genoptræningsprocessen, vil vi nu formulere vores eget forslag. Mange af de eksisterende løsninger benytter sig af en form for robot-hardware, som hjælper med at styre patientens krop, men vil vi i stedet fokusere på en softwareløsning.

Vi vil lave et stykke software, der kan modtage rå EEG-signaler og ud fra disse afgøre, om en patient tænker på at bevæge venstre eller højre arm. Dette kan så bruges som input til et andet stykke software, som for eksempel et simpelt spil, hvor man enten skal styre til højre eller venstre. På denne måde kan patienten lave genoptræningsøvelser samtidig med at vedkommende bliver underholdt, hvilket forhåbentligt kan virke motiverende i forhold til genoptræningen.

Vi fremsætter derfor følgende problemformulering.

Hvordan kan man implementere et stykke software, der ved brug af consumer-grade EEG-teknologi assisterer en patient med genoptræning?

Problemformuleringen giver anledning til to forskellige fokus. Det ene ligger op til at fokusere på at udvikle et stykke software oven på et consumer-grade EEG-produkt, mens det andet fokus i højere grad drejer sig om, hvordan vi implementere et stykke software, der faktisk hjælper en patient med genoptræning. Begge muligheder er i sig selv store områder. Vi ønsker derfor at afgrænse dette projekt til et enkelt fokus. Vi ønsker at undersøge om, vi ved hjælp af en softwareløsning kan nå en nøjagtighed for et consumer-grade EEG-produkt, der kan konkurrere med industri EEG-produkter. Dette betyder også, at vores kontekstmæssige fokus ligger i det økonomiske aspekt af vores problemformulering. Netop da consumer-grade produkter er billigere end industri-produkter.

Helt specifikt tager vi udgangspunkt i consumer-produktet Emotiv EPOC neuroheadset. Disse valg leder os til en række væsentlige spørgsmål:

1. Hvordan kan vi indsamle, fortolke og analysere EEG-data?
2. Hvordan kan man designe softwaren på en hensigtsmæssig måde?
3. Hvilke kriterier er der for succes af softwaren? Og hvordan kan vi måle dem?

1.4 Indgangsvinkel og retning

Da vi har valgt at arbejde med consumer-grade EEG hardware, så kan vi ikke ændre på hardwaren for at opnå bedre nøjagtighed – det ville underminere hele formålet med projektet. Derfor har vi valgt udelukkende at fokusere på at lave et stykke software, der kan skabe en tilfredsstillende nøjagtighed. Vi forsøger derfor at give en matematisk model for vores løsning. Vi har valgt at tage udgangspunkt i Barachant et al. [2012].

Artiklen beskæftiger sig med klassifikation af BCI-data gennem Riemann geometri. I appendiks A.5 beskriver vi derfor en tilstrækkelig delmængde af teorien om Riemannske mangfoldigheder.

2

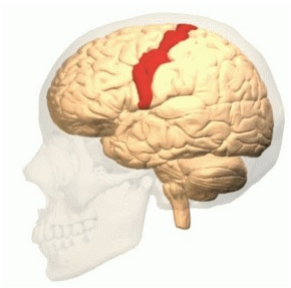
Hjernen

EEG-apparater måler elektrisk potentiale på skalpen, som stammer fra aktiviteter i hjernen. EEG-målinger viser altså indirekte hjerneaktivitet, og vi vil derfor i dette afsnit kort beskrive væsentlige dele af hjernens opbygning og funktion.

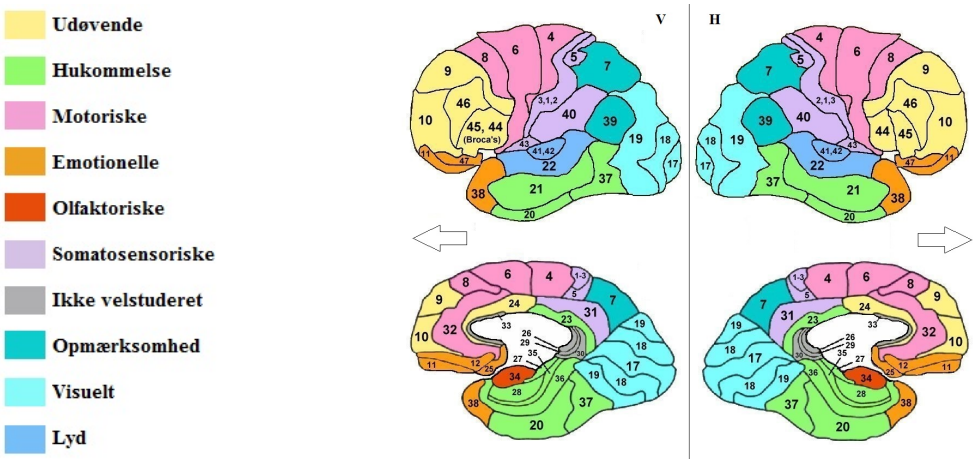
2.1 Opbygning

Menneskets hjerne består af flere milliarder nerveceller, også kaldet neuroner. Det anslås at mennesker har ca. 100 milliarder neuroner i hjernen, hvoraf ca. 20 milliarder findes i de to hjernehalvdele, som kaldes hemisfærer. Hjernen kan deles op på mange måder såsom: Hjernestammen, lillehjernen, midthjernen, storhjernen og hjernebarken. I dette projekt vil fokus være på hjernebarken, som også kaldes *cortex cerebri*. I denne del af hjernen foregår den elektriske aktivitet, som er målbar ved hjælp af elektroencefalografi (EEG). Hjernebarken, som dækker hemisfærernes overflade, består af en grå substans, der kaldes *substantia grisea*. Den grå substans rummer nervecellelegemer samt udløbere fra disse legemer, som giver hjernebarken dens grålige farve. Det er i hjernebarken at de primære motoriske og sensoriske områder findes. Området, som kaldes for den *motoriske cortex*, ligger i en dyb fure kaldet *sulcus centralis*, som begrænses af hjernevindingen *gyrus praecentralis*, der har med bevægelse at gøre. For placeringen i forhold til hjernen se figur 2.1. Fra det motoriske område udgår impulserne spejlvendt til legemets muskler, hvilket vil sige at den venstre halvdel af hjernen styrer den højre halvdel af kroppen og omvendt. Dette skyldes at ledningsbanerne krydses i hjernestammen og sender derfor impulserne til modsatte side. Altså vil det sige, at hvis man bruger højre arm, vil signalet stamme fra venstre *gyrus praecentralis* og omvendt. [Gyldendals åbne encyklopædi]

Den motoriske hjernebark har en funktionel topografisk opdeling, hvor neuroner i bunden af *gyrus praecentralis* styrer muskler i ansigtet og på halsen, mens midten af *gyrus praecentralis* styrer musklerne på arm og krop, og toppen af *gyrus praecentralis* styrer musklerne på benet. Hjernebarken består af seks forskellige lag, som den tyske anatom Korbinian Brodmann i 1909 underopdelte i 47 funktionelt bestemte områder [Gyldendals åbne encyklopædi]. Hvilke områder der styrer hvilke funktioner kan groft forkortes ned til 10 områder, som står for henholdsvis: Udøvende, hukommelse, motoriske, emotionelle,



Figur 2.1: Placering af gyrus praecentralis på hjernebarken [Database Center for Life Science, 2012].



(a) Overordnede funktionsområder.

(b) Brodmann Atlas.

Figur 2.2: Opdeling af hjernebarkens funktionelt bestemte områder ifølge Korbinian Brodmann [Sterman Kaiser Imaging Laboratory, a]. Nogle af de nummererede felter er beskrevet i tabel 2.2.

olfaktoriske, somatosensoriske, opmærksomhed, visuelle og lyd samt de områder, som endnu ikke er velstuderet. Placeringen af de forskellige funktionaliteter kan ses på figur 2.2, som er et atlas over Brodmanns opdeling af de funktionelle områder. På figur 2.2 ses henholdsvis venstre og højre hjernehalvdel. De to øvre billeder er de to hemisfærers overflade, den *laterale* flade, mens de nedre billeder er et tværsnit af hjernen, den *mediale* flade. De to pile viser hjernernes retning, pegende mod ansigtet.

2.2 Neuroner

De mange neuroner i hjernen stammer fra fødslen. Forskning viser dog, at der i et begrænset omfang dannes nye neuroner gennem livet, samtidig med at vi mister omkring 10 procent af hemisfærernes neuroner gennem livets første 70 år [Teknologi-Rådet; Gyldendals åbne encyklopædi]. Neuroner kan ikke dele sig som de fleste andre celler i kroppen, så hvis der sker en skade på hjernen vil de beskadigede neuroner ikke blive erstatte med nye. Derimod kan de tilbageværende neuroner danne nye forbindelser samt gendanne gamle forbindelser. Derfor er hjernen under konstant udvikling, og tabet af neuroner behøver ikke at resultere i permanent hjerneskade. [Teknologi-Rådet]

Et neuron består af en cellekrop, dendritter, et akson og endeknopper. Inde i cellekroppen findes DNA, der er med til at bestemme dens funktionelle opgaver. På cellekroppen sidder der mange små dendritter. Dendritter fungerer som neuronets øre, da de er neuronens modtagere. På cellekroppen sider der desuden også et akson, som agerer som afsender. På enden af aksonet sidder endeknopperne, der afgiver kemiske signaler som andre neuroners dendritter kan modtage. Neuronerne kommunikerer gennem elektriske impulser, som ændres fra elektrisk signal til et kemisk signal, som dendritterne kan opfange. Det sker ved at aksonet frigiver et kemisk stof, som binder sig til nogle receptorer, der sidder på dendritterne. Receptorne er følsomme over for de kemiske signaler, og kan kun acceptere bestemte signaler. Hvis signalet accepteres kan de virke fremmende ved at sende de elektriske impulser videre mellem neuronerne, eller de kan virke hæmmende ved at bremse den elektriske impuls [Teknologi-Rådet].

2.3 Emotiv headset

Emotiv[1] har udviklet et EEG-headset, som gør brug af 14 elektroder samt to referenceelektroder. Emotivs headset kan ses på figur 2.3.

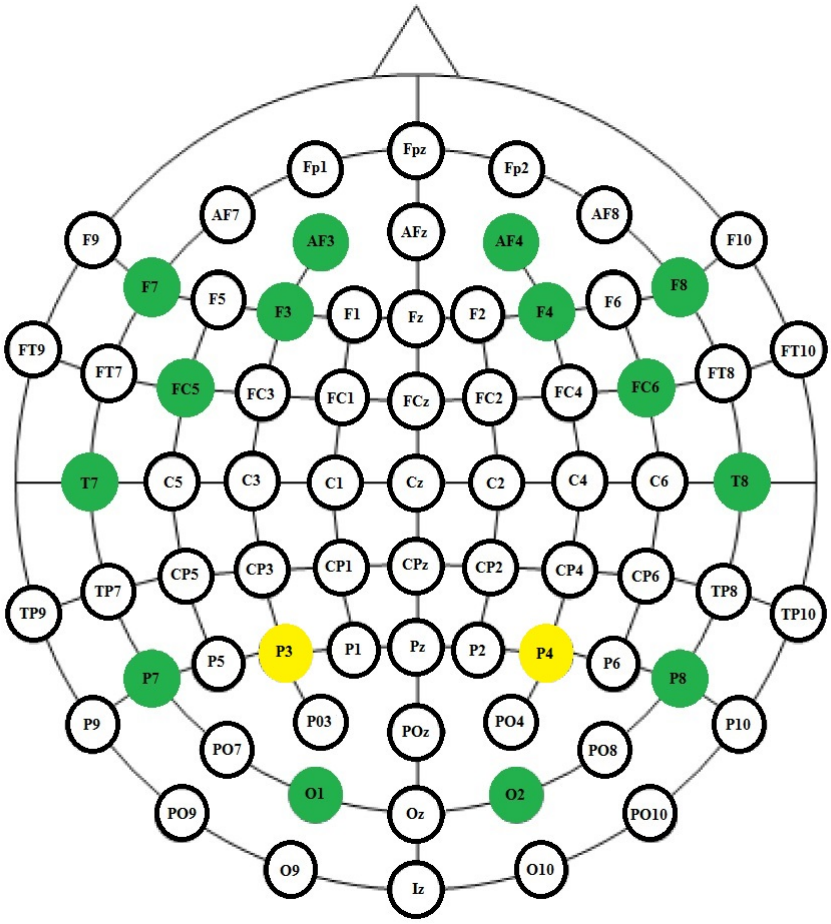
Figur 2.3: Emotivs EEG-apparat med 16 elektroder.

De to referencepunkter anvendes til at kalibrere og give information om signalstyrken fra den enkelte elektrode. I softwaren til Emotiv angives signalstyrken via farverne sort, rød, gul og grøn, der repræsenterer værdier på en skala fra intet signal til godt signal. Elektroderne placeres efter det internationale 10-20 system, der referer til at elektroderne er spredt ud med en procentvis afstand på 10 og 20 procent i forhold til afstanden mellem kraniets front og bag eller venstre og højre side af kraniet [Emotiv]. På figur 2.4 ses placeringen af de 14 elektroder, markeret med grøn, i forhold til resten af det internationale 10-20 system. Det er dog vigtigt at bemærke, at i modsætning til elektroder påsat af en ekspert i et laboratorium, kan den præcise placering af elektroderne på Emotivs headset variere både fra bruger til bruger og fra forsøg til forsøg. Elektroderne navngives efter placeringen på hjernen, og for at adskille elektroderne fra højre og venstre, angives venstre med ulige tal, og højre med lige tal. Navngivningen af elektroderne på Emotiv headsettet

[1] http://www.emotiv.com

kan ses i tabel 2.1 sammen med hvilke områder elektroderne dækker, tabel 2.2 viser hvilke handlinger, de forskellige områder styrer.

Bogstaverne AF, F, FC, T, P og O, står for henholdsvis anterior–frontal, frontal, fronto-central, temporal, parietal, occipital [Jurcak et al., 2007].



Figur 2.4: Placering af Emotivs 14 elektroder i forhold til 10-20 systemet. De grønne elektroder er de elektroder, som Emotiv headsettet aflæser, mens de gule elektroder er referencepunkterne for Emotivs headset.

Elektrode	AF3	F7	F3	FC5	T7	P7	O1	O2	P8	T8	FC6	F4	F8	AF4
Område	09	47	08	BROCA	42	37	18	18	37	21	44	08	45	09

Tabel 2.1: De 14 elektroder, som Emotivs headset anvender, samt deres område i forhold til figur 2.2 [Sterman Kaiser Imaging Laboratory, b].

2.4 Hjernerytmer

EEG-målinger består af en række målinger over et givet tidsinterval. En sådan EEG-måling kan kategoriseres i forskellige frekvensbånd, også kaldet rytmer. En oversigt over de forskellige frekvensbånd kan ses i tabel 2.3. Hvert frekvensbånd forbindes med en række kognitive tilstande, hvor aktivitet på netop dette frekvensbånd er fremherskende. Nogle af

Område	Beskrivelse	Styring
08	Lateral og medial supplerende motorisk område	Styrelse af øjenbevægelser
09	Midterfrontale gyrus	Styring af udøvende funktioner
18	Sekundær visuel cortex - midterste occipitale gyrus; extrastriate cortex	Modtage, organisere og udtrykke visuelle stimuli
21	Midtertemporale gyrus	Analyse af visuel information, især form og bevægelse
37	Bagest inferiør temporale gyrus, midtertemporale gyrus og tenformet gyrus	Visuel analysering og association
42	Sekundær auditiv cortex	Genkendelse af lyd og tale
44/BROCA	Inferiør frontal gyrus - Pars opercularis	Producering af sproget
45	Inferiør frontal gyrus - Pars triangularis	Producering af sproget samt mere
47	Inferiør frontal gyrus - Pars orbitalis	Deltager i de emotionelle aktiviteter samt udøvende funktioner

Tabel 2.2: De anvendte områder i forhold til figur 2.2. For den fulde liste henvises der til Sterman Kaiser Imaging Laboratory [b].

Navn	Frekvensbånd (Hz)
δ	$1-4$
θ	$4-8$
α	$8-13$
μ	$10-14$
β	>13
γ	$30-40$

Tabel 2.3: Oversigt over frekvensbånd for EEG-målinger [Alamgir et al., 2010; Nunez and Srinivasan, 2007]. Inddelingen kan variere, se fx Buzsáki [2006].

disse forbindelser, som er særligt interessante i forbindelse med BCI, er som følger [Gittis; Blankertz et al., 2008].

δ og θ forbindes med andre former for søvn end REM-søvn. Derudover forbindes θ med frustration mens δ forbindes med koncentration.

α forbindes med målinger af en vågen person med lukkede øjne, mens amplituden på signalet i dette frekvensbånd falder når personen åbner øjnene.

μ forbindes med motorisk aktivitet, både egentlig og tænkt aktivitet, samt følesansen.

β forbindes med målinger af en person som enten er udsat for eksterne stimuli eller befinder sig i en dyb, REM-søvn. Forbindes også med høj mental aktivitet og at huske.

3

Maskinindlæring med BCI

I de følgende afsnit har vi brug for nogle generelle begreber som bruges til at beskrive forskellige algoritmer. Disse begreber vil vi beskrive her.

3.1 Områder indenfor maskinindlæring

Mange maskinindlæringsproblemer hører til i en af to kategorier af problemer: superviseret og usuperviseret indlæring.

Superviseret indlæring opererer med et input i X og et output i Y, og prøver at finde en afbildning fra X ind i Y [Alpaydin, 2010]. Nogle værdier i X og deres tilsvarende værdi i Y er kendt på forhånd; disse udgør træningsmængden for problemet. Eksempler i træningsmængden kaldes mærkede eksempler. En superviseret indlæringsalgoritme forsøger at lære afbildningen ud fra eksemplerne i træningsmængden. Det antages at der findes en model $g(\cdot)$ så

$$y = g(x|\theta),$$

hvor θ er en mængde af parametre, $x \in X$ og $y \in Y$ [Alpaydin, 2010]. Hvis $Y = \mathbb{R}$, de reelle tal, kaldes problemet et regressionsproblem, og $g(\cdot)$ kaldes da regressionsfunktionen. Hvis Y derimod er diskret, kaldes problemet et klassificeringsproblem, og $g(\cdot)$ kaldes diskriminantfunktionen eller classifieren. X kan være en fuldstændig vilkårlig mængde. For at bruge en superviseret indlæringsalgoritme skal man altså først vælge en model med et antal parametre. Dette kunne for eksempel være en lineær model med parametre w og w_0, hvor

$$y = wx + w_0,$$

og algoritmen vil så optimere parametrene for modellen. Når algoritmen har lært modellen g, kan denne model så bruges til at estimere eller klassificere nye, umærkede eksempler x ved at udregne $y = g(x)$.

I modsætning til superviseret indlæring, har usuperviseret indlæring kun inputtet X, og forsøger at finde strukturer og mønstre i X [Alpaydin, 2010]. En af de mest benyttede usuperviserede maskinindlæringsteknikker er clustering, som forsøger at inddele inputtet i forskellige clusters, hvor elementerne i hvert cluster har nogle fællestræk. En clustering-algoritme fortæller ikke noget om, hvad disse fællestræk er, og en clustering-algoritme kan derfor, i modsætning til superviseret indlæring, ikke bruges til at klassificere ny data i forhold til disse clusters.

I klassificeringsalgoritmer ved man på forhånd, hvilke grupperinger dataen skal inddeles i. Disse grupperinger kaldes klasser.

Definition 3.1
En *klasse* er en af de grupperinger, som en superviseret indlæringsalgoritme skal diskriminere mellem.

Hvis for eksempel vores algoritme skal kunne kende forskel på at en forsøgsperson tænker på at bevæge højre eller venstre arm, så vil vi have to klasser: højre arm og venstre arm.

3.2 BCI-systemer

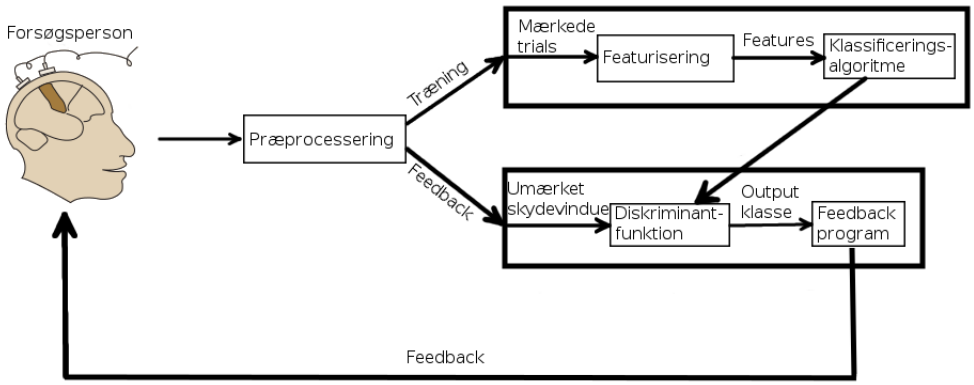
Et system, hvor hjernesignaler måles og bruges til at påvirke eller styre eksterne apparater kaldes en hjerne-maskin-grænseflade [1] [Makeig et al., 2012]. Overordnet set består et BCI-system af et subjekt, hvis hjernesignaler måles, og et apparat, som skal reagere på målingerne. For at gøre dette, er der brug for et mellemled, der fortolker hjernesignalerne, og ud fra denne fortolkning giver et resultat, som kan forstås af apparatet. I dette mellemled kan maskinindlæring med fordel benyttes.

Et eksempel på et maskinindlæringsbaseret BCI-system kan ses på figur 3.1.

EEG-målinger kommer fra EEG-apparatet på forsøgspersonens skalp. Herefter udføres der præprocessering på dataen, som kan bestå af mange forskellige indgreb på dataen. Dette kan for eksempel være for at fjerne støj fra dataen, hvilket kan forbedre resultaterne af EEG-systemet [Makeig et al., 2012]. Støj kan komme fra forskellige kilder såsom el-netværket eller støj fra muskel- og øjenbevægelse i ansigtet. En anden slags præprocessering, der kan forbedre resultater er båndpasfiltrering, som udvælger et bestemt frekvensinterval af dataen og smider resten væk. Hvis man på forhånd ved, i hvilket frekvensinterval, de aktiviteter, man er interesseret i, fremkommer, så kan en båndpasfiltrering gøre en i stand til kun at arbejde med det relevante interval. Nogle klassificeringsalgoritmer antager at dataen har visse egenskaber, såsom at dataen er centreret så middelværdien af dataen er 0. Disse ændringer foretages også i præprocesseringsfasen.

Efter præprocesseringsfasen er der to forskellige faser, som kan bruge den præprocesserede data: træningsfasen og feedback-fasen. I træningsfasen er dataen mærket, så systemet på forhånd ved hvilken klasse hver trial tilhører. På baggrund af disse trials laver systemet features, som bruges af klassificeringsalgoritmen. Hvilken slags features, der er

[1] Forkortelsen BCI kommer fra det engelske "brain-computer interface".



Figur 3.1: Et overblik over et BCI-system [Blankertz et al., 2008]. Processen er delt op i to faser: En træningsfase og en feedback-fase. I træningsfasen lærer systemet at klassificere brugerens målinger, og denne klassificering bruges så i feedback-fasen til at klassificere de input, der kommer fra brugerens målinger under brug, og som derefter påvirker apparatet.

tale om, afhænger i høj grad af hvilken klassificeringsalgoritme, der vælges. Omvendt kan klassificeringsalgoritmen også afhænge af, hvilken slags featurisering der vælges. Klassificeringsalgoritmen lærer ud fra de givne features en diskriminantfunktion, som gemmes og derefter kan bruges i feedback-fasen.

Når diskriminantfunktionen er lært, kan systemet gå i feedback-fasen, hvor det i real-time kan klassificere umærket input fra EEG-målinger, så systemet kan bruges til at styre en applikation eller et apparat. Efter at dataen er blevet præprocesseret, bruges diskriminantfunktionen til at klassificere dataen. I modsætning til træningsfasen, hvor dataen er delt ind i trials, hvor man ved hvilken del af signalet, der rummer den handling, der skal klassificeres, så kommer dataen i feedback-fasen i en kontinuert strøm. Derfor laves der i stedet et skydevindue, hvor ny data løbende bliver tilføjet til vinduet, mens gammel data løbende smides væk. Hver gang skydevinduet opdateres, vil diskriminantfunktionen klassificere skydevinduet, og den givne klassificering sendes derefter til et feedback-program. Feedback-programmet bruger dette input til at give en form for feedback til forsøgspersonen, hvilket for eksempel kan være at afspille en lyd, vise noget på en skærm eller at styre et apparat.

3.2.1 Trials

Et input fra EEG-apparatet beskrives som en *trial*, også kaldet en *data-matrix*. I en trial angiver hver række en kanal fra EEG-apparatet, mens hver kolonne angiver et tidspunkt, hvor hver kanal blev aflæst. Når vi taler om et input T, er T altså en $m \times n$-matrix, hvor m er antallet af kanaler på EEG-apparatet og n er antallet af tidspunkter, som inputtet indeholder målinger fra. En afbildning af dette kan ses på figur 3.2.

3.3 Præprocessering

Vi vil i dette afsnit beskrive nogle af de teknikker, der bruges i præprocesseringsfasen. Afsnittet behandler datacentrering og signalbehandling. I afsnit 3.3.2 gives en beskrivelse

$$\begin{array}{c} \\ m \text{ rækker} \end{array} \begin{array}{c} \overset{n \text{ søjler}}{\begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \dots & x_{m,n} \end{bmatrix}} \end{array}$$

Figur 3.2: Repræsentation af EEG-input som en data-matrix, hvor m angiver antal måletidspunkter og n angiver antal kanaler.

af Fouriertransformationen, som er essentiel i forhold til at kunne lave signalbehandling. Dernæst giver vi en konkret anvendelse af Fouriertransformationen til signalfiltrering.

3.3.1 Datacentrering

Nogle maskinindlæringsalgoritmer antager, at middelværdien for dataen er 0. For at opnå dette, kan man centrere dataen omkring middelværdien. For en matrix kan dette gøres ved at gange matricen med centreringsmatricen. Centreringsmatricen C_k er en $k \times k$ matrix som er givet ved

$$C_k = I_k - \frac{1}{k}\mathbb{O},$$

hvor I_k er $k \times k$-identitetsmatricen og $\mathbb{O}$ er en $k \times k$ matrix med 1 i alle indgange. For en trialmatrix X er den centrerede matrix X_C givet ved

$$T_C = TC_kT^T. \tag{3.1}$$

3.3.2 Diskret Fouriertransformation

Fouriertransformationen (FT) er en matematisk operator $\mathcal{F}$, der opererer på funktioner over det komplekse tallegeme. Operatoren er defineret både for kontinuerte og diskrete funktioner, og begge definitioner findes i endelige og uendelige tilfælde. Vi beskriver blot den endelige diskrete Fouriertransformation (DFT), da vi arbejder med endelige diskrete blokke af data. Generelt konverterer FT en given funktion fra et tidsdomæne til en funktion i et frekvensdomæne [Brown, 2011]. DFT tager selvfølgelig en diskret funktion, men i det endelige tilfælde reduceres dette blot til en endelig, kompleks vektor.

Definition 3.2 (Diskret Fouriertransformation [Lutus, 2008])
Konvertering fra et tidsdomæne til et frekvensdomæne ved DFT er defineret ved:

$$\hat{x}(\omega) = \sum_{t=0}^{N-1} x(t)e^{\frac{-i\omega t}{N}}, \ \forall \omega \in \{0, 2\pi, \dots, 2\pi(N-1)\}, \tag{3.2}$$

hvor

- $x(t)$ er en kompleks vektor i tidsdomænet, og t er et tidspunkt (et indeks).
- $\hat{x}(\omega)$ er en kompleks vektor i frekvensdomænet, og ω er vinkelfrekvensen $\omega = 2\pi n$ hvor n er et indeks.
- N angiver længden af vektorerne.

Frekvensrepræsentationen af en funktion gør det muligt at ændre på signalet, såsom at fjerne støj. Efter at man har foretaget sine ændringer er det muligt få tidsdomænerepræsentationen tilbage vha. den diskrete inverse Fouriertransformation (DIFT).

Definition 3.3 (Diskret invers Fouriertransformation [Lutus, 2008])
Konvertering fra et frekvensdomæne til et tidsdomæne ved DIFT er defineret ved:

$$x(t) = \frac{1}{N} \sum_{\omega=0}^{N-1} X(\omega) e^{\frac{i\omega t}{N}}, \ \forall t \in \{0, \ldots, N-1\}, \tag{3.3}$$

hvor variablerne betyder det samme som i definition 3.2.

Selv om signalet $x(t)$ er en reel vektor, er frekvensrepræsentationen $\hat{x}(\omega)$ generelt kompleks.

Generelt er de to operationer FT og IFT reciprokke operationer [Lutus, 2008], dvs. de er hinandens inverse. Det betyder, at de ændringer der foretages i frekvensdomænet bevares under konvertering til tidsdomænet og omvendt.

3.3.3 Butterworth-filtrering

Butterworth-filtret er et signalbehandlingsfilter, som benyttes til at filtrere støj fra et signal og udvælge et frekvensvindue af signalet ved båndpasfiltering. Filtret skyldes Stephen Butterworth, og får derfor sit navn efter ham [Butterworth, 1930]. Ofte beskrives Butterworth-filtret som et lavpasfilter, og Butterworth beskrev det også selv sådan, men han viste at hans lavpasfilter kunne modificeres til at give et højpas-, båndpas- eller spærrefilter [Butterworth, 1930]. Filtret opererer på frekvensdomænerepræsentationen af en funktion, hvilket kan opnås gennem Fouriertransformationen. Vi benytter Butterworth-filtret til at filtrere støj fra signalerne ved at lave et båndpasfilter, som kun bibeholder frekvenser i et givet frekvensinterval, og fjerner alle andre.

Hvordan amplituden af de enkelte frekvenskomponenter af signalet ændres under en filtrering kan beskrives ved gainfunktionen $G(\omega)$, der giver hvor meget signalet forstærkes ($G(\omega) > 1$) eller dæmpes ($G(\omega) < 1$) for en given frekvens ω.

Vi kan udtrykke gainfunktionen af et Butterworth lavpasfilter vha. tre parametre som i definition 3.4.

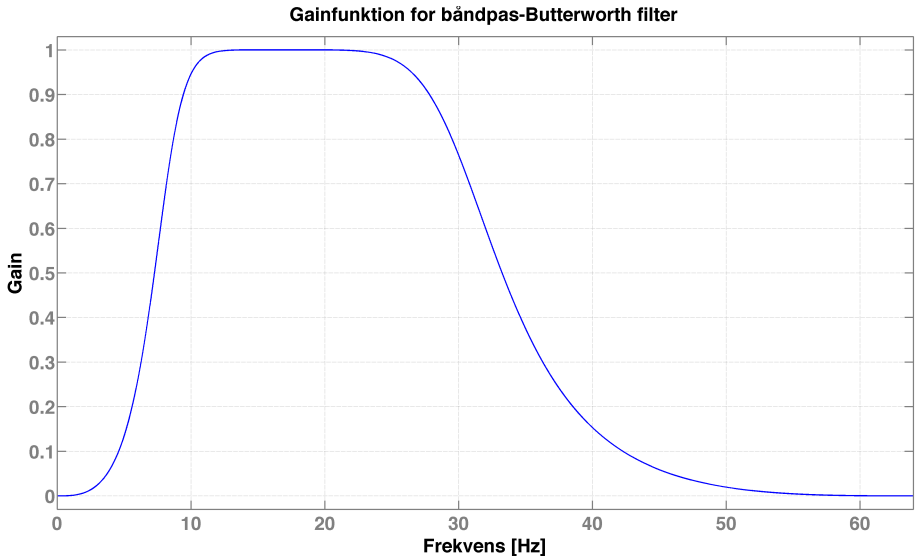
Definition 3.4 (Butterworth-filter [Shirkey, 2013])
Butterworths gainfunktion er givet ved:

$$G(\omega) = \frac{G_0}{\sqrt{1 + \left(\frac{\omega}{\omega_c}\right)^{2n}}}, \tag{3.4}$$

hvor $G(\cdot)$ angiver gainfunktionen og parametrene

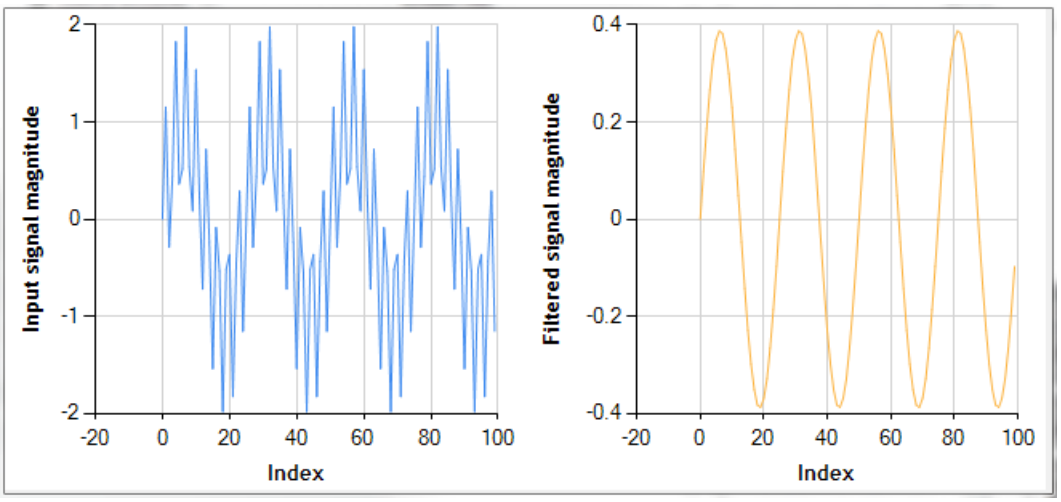
1. ω_c angiver cut-off frekvensen.
2. G_0 angiver DC gain, dvs. gain ved nulfrekvensen.
3. n angiver orden af filtret.

Ordnen n på filtret bestemmer, hvor skarpt funktionen aftager eller vokser i området omkring cut-off frekvensen. Cut-off frekvensen ω_c er den frekvens der ligger cirka halvvejs på stykket hvor funktionen aftager eller vokser. Præcis ved cutoff-frekvensen er $G(\omega_c) = \frac{G_0}{\sqrt{2}} \approx 0,707 \times G_0$.



Figur 3.3: Gainfunktion for 6. ordens båndpas-Butterworth filter med frekvens-cutoffs 8 og 30 Hz.

Figur 3.3 viser gainfunktionen for et båndpasfilter i båndet 8-30Hz, der er det frekvensvindue vi anvender til eksperimenter. Figur 3.4 viser et eksempel på støjfjernelse fra et signal ved filtrering.



Figur 3.4: Eksempel på anvendelse af Butterworth-filtret. Den venstre graf viser signalet med støj før anvendelse af Butterworth-filtret, mens højre graf viser resultatet efter anvendelse af Butterworth-filret. [Shirkey, 2013]

4

Klassificering ved Riemannsk geometri

I dette kapitel vil vi beskrive hvordan man kan klassificere EEG-data ved brug af koncepter fra Riemannsk geometri. Vi vil først beskrive nogle af disse koncepter abstrakt og derefter give en klassificeringsalgoritme, der gør brug af koncepterne. Essensen af denne klassificeringsmetode er at definere afstand og middelværdi for punkter i en Riemannsk mangfoldighed og så bruge disse til at finde den mindste afstand mellem det data, der ønskes at klassificere, og middelværdien for de enkelte klasser af træningsdata.

Den generelle matematiske baggrund om Riemann-geometri er beskrivet i appendiks A.5, og benyttes her på rum af symmetriske positiv definitte (SPD) matricer som beskrevet i appendiks A.2.

4.1 SPD-matricer som Riemann-flader

I dette afsnit vil vi beskrive sammenhængen mellem EEG-målinger og Riemannsk geometri og introducere de begreber og koncepter vi har brug for for at kunne definere afstand og middelværdi for punkter i en sådan Riemannsk mangfoldighed.

Data-matricerne givet som input fra et EEG-apparat kan bruges til at få information om den spatiale struktur i dataen, og kovariansmatricen for et givet input indeholder netop denne information.

Definition 4.1 (Kovariansmatrix)
Givet en vektor af n stokastiske variable

$$\boldsymbol{X} = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix},$$

er kovariansmatricen Σ for $\boldsymbol{X}$ en matrix hvis (i,j)'te indgang er $\mathrm{Cov}[X_i, X_j]$ (se

appendiks A.1). Dette giver en matrix

$$\Sigma = \begin{bmatrix} \mathrm{Cov}[X_1, X_1] & \mathrm{Cov}[X_1, X_2] & \dots & \mathrm{Cov}[X_1, X_n] \\ \mathrm{Cov}[X_2, X_1] & \mathrm{Cov}[X_2, X_2] & \dots & \mathrm{Cov}[X_2, X_n] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}[X_n, X_1] & \mathrm{Cov}[X_n, X_2] & \dots & \mathrm{Cov}[X_n, X_n] \end{bmatrix},$$

hvor indgangene på diagonalen er variansen for $X_1, X_2, \dots, X_n$.

Da sandsynlighedsfunktionerne for udfaldsrummet sjældent er kendte, benyttes i stedet et estimat for kovariansmatricen, som kaldes en *sample covariance matrix* (SCM).

Definition 4.2 (Sample covariance matrix (SCM))
Givet en trialmatrix T er den tilsvarende sample covariance matrix Σ givet ved

$$\Sigma = \frac{1}{t_s - 1} T T^T,$$

hvor t_s er antallet af søjler i T.

I ovenstående udtryk for en SCM antages det at middelværdien $\mathrm{E}[X] = 0$.

Mængden af symmetrisk positivt definitte (SPD) $n \times n$ matricer $P(n)$ udgør en differentiabel Riemannsk mangfoldighed $\mathcal{M}$ [Fletcher and Joshi, 2004]. SCM'er er garanteret at være SPD, og er derfor indeholdt i $\mathcal{M}$ [Barachant et al., 2012]. Klassificeringsalgoritmens teoretiske grundlag er geometrien af denne Riemannske mangfoldighed.

4.1.1 Tangentrummet og eksponential/logaritmeafbildningerne

Tangentvektorer i et punkt på mangfoldigheden ligger ikke i rummet $P(n)$, men ligger i stedet i tangentrummet $T_P\mathcal{M}$, som er ækvivalent med mængden af $n \times n$ symmetriske matricer $S(n)$ [Barachant et al., 2012]. Bemærk at tangentrummet afhænger af punktet P.

For ethvert punkt $P \in P(n)$ kan vi altså definere et tangentrum $T_P P(n) = S(n)$ bestående af alle tangentvektorerne i punktet P. Ved at tage gradienten af en skalar funktion over $P(n)$ i et punkt P får vi altså en tangentvektor i tangentrummet $S(n)$. For at afbilde tangentvektoren ned i den Riemannske mangfoldighed igen, bruger vi Riemann-eksponentialfunktionen $\mathrm{Exp}_\mathrm{P} : S(n) \to P(n)$ defineret ved

$$\mathrm{Exp}_\mathrm{P}(S_i) = P_i = P^{\frac{1}{2}} \exp(P^{-\frac{1}{2}} S_i P^{-\frac{1}{2}}) P^{\frac{1}{2}},$$

hvor $\exp(\cdot)$ er eksponentialfunktionen for matricer som defineret i appendiks A.3. En algoritmisk tilgang til at beregne $\mathrm{Exp}_\mathrm{P}(S_i)$ er givet i algoritme 4.1.

Input: Startpunkt $P \in P(n)$
Input: Tangentvektor $S_i \in S(n)$.
Output: $\text{Exp}_\text{P}(S_i) \in P(n)$.
1 Diagonalisér $P = U\Lambda_1 U^T$, hvor U har egenvektorerne som søjler og Λ_1 har egenværdierne på diagonalen;
2 Lad $G = U\sqrt{\Lambda_1}$;
3 Lad $Y = G^{-1}S_i(G^{-1})^T$;
4 Diagonalisér $Y = V\Lambda_2 V^T$;
5 Lad $\text{Exp}_\text{P}(S_i) = (GV)\exp(\Lambda_2)(GV)^T$;

Algoritme 4.1: Riemann-eksponentialfunktionen [Fletcher and Joshi, 2004].

Riemann-eksponentialfunktionen har også en invers afbildning, som kaldes Riemann-logaritmefunktionen $\text{Log}_\text{P} : P(n) \to S(n)$, og er defineret ved

$$\text{Log}_\text{P}(P_i) = S_i = P^{\frac{1}{2}} \log(P^{-\frac{1}{2}} P_i P^{-\frac{1}{2}}) P^{\frac{1}{2}},$$

hvor $\log(\cdot)$ er logaritmefunktionen for matricer som defineret i appendiks A.3. For at beregne $\text{Log}_\text{P}(P_i)$, kan algoritme 4.2 benyttes.

Input: Startpunkt $P \in P(N)$
Input: Endepunkt $P_i \in P(n)$.
Output: $\text{Log}_\text{P}(P_i) \in S(n)$.
1 Diagonalisér $P = U\Lambda_1 U^T$, hvor U har egenvektorerne som søjler og Λ_1 har egenværdierne på diagonalen;
2 Lad $G = U\sqrt{\Lambda_1}$;
3 Lad $Y = G^{-1}P_i(G^{-1})^T$;
4 Diagonalisér $Y = V\Lambda_2 V^T$;
5 Lad $\text{Log}_\text{P}(P_i) = (GV)\log(\Lambda_2)(GV)^T$;

Algoritme 4.2: Riemann-logaritmefunktionen [Fletcher and Joshi, 2004].

Forholdet mellem Riemann-eksponential- og Riemann-logaritmefunktionen kan ses på figur 4.1. Man kan se, at ved at tage Log_P i et punkt $P_i \in \mathcal{M}$ får vi et punkt $S_i \in T_P\mathcal{M}$ som er en tangentvektor for punktet P. Dette punkt kan vi afbilde ned i $\mathcal{M}$ igen ved at tage $\text{Exp}_\text{P}(S_i)$, hvilket giver et punkt $P_i \in \mathcal{M}$. Vi ser desuden at $\text{Log}_\text{P}(\text{Exp}_\text{P}(S_i)) = S_i$ og $\text{Exp}_\text{P}(\text{Log}_\text{P}(P_i)) = P_i$, altså at Log_P og Exp_P er hinandens inverse.

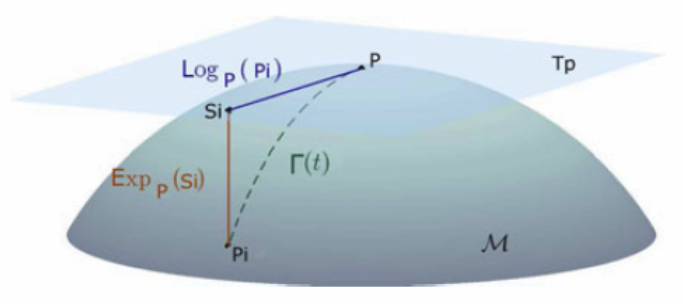
4.1.2 Afstande i $P(n)$

Det indre produkt på tangentrummet $S(n)$ i punktet P er givet ved den såkaldte naturlige metrik [Barachant et al., 2012]

$$\langle S_1, S_2 \rangle_P = \text{Tr}(S_1 P^{-1} S_2 P^{-1}). \tag{4.1}$$

Den tilhørende norm er $||S||_P = \sqrt{\langle S, S \rangle}$.

Givet to punkter $P_1, P_2 \in P(n)$, ønsker vi at kunne bestemme afstanden mellem disse to punkter. Til dette vil vi bruge den Riemannske geodætiske metrik (afstandsfunktion) d_R



Figur 4.1: En Riemannsk mangfoldighed $\mathcal{M}$ med tangentrum $T_P\mathcal{M}$ i punktet P. Hvis vi betragter vejen $\Gamma(t)$ mellem P og P_i, så giver $\mathrm{Log_P}(P_i)$ en tangentvektor S_i som opfylder at $\mathrm{Exp_P}(S_i) = P_i$. [Barachant et al., 2012]

givet i definition A.17 , som er længden af den korteste vej (geodæten) mellem to punkter. For rummet $S(n)$ med indre produkt 4.1 er den givet ved [Moakher, 2005]

$$d_R(P_1, P_2) = \| \log(P_1^{-1} P_2) \|_F = \left(\sum_{i=1}^{n} \log(\lambda_i)^2 \right)^{1/2}, \tag{4.2}$$

hvor $\lambda_i, i = 1, 2, \ldots, n$ er de reelle egenværdier for $P_1^{-1} P_2$ og $\| \cdot \|_F$ er Frobeniusnormen af en matrix givet ved

$$\|P\|_F^2 = \mathrm{Tr}(PP^T).$$

Ved at definere en diagonalmatrix af egenværdierne $\Lambda_{P_1,P_2} = \mathrm{diag}(\lambda_1, \cdots, \lambda_n)$ kan dette skrives

$$d_R(P_1, P_2) = \sqrt{\mathrm{Tr}\left(\log(\Lambda_{P_1,P_2})^2\right)}. \tag{4.3}$$

4.1.3 Middelværdier

For SPD-matrixrummet $P(n)$ definerer vi middelværdien af en mængde matricer fra Fréchet-middelværdien i ligning A.25. Givet en mængde af punkter $A = \{P_1, P_2, \ldots, P_k\}$, hvor $P_1, P_2, \ldots, P_k \in P(n)$, så findes middelværdien for disse punkter ved at minimere funktionen

$$\rho_A(P) = \frac{1}{2k} \sum_{i=1}^{k} d_R(P, P_i)^2 = \frac{1}{2k} \sum_{i=1}^{k} \mathrm{Tr}\left(\log(\Lambda_{P,P_i})^2\right), \tag{4.4}$$

hvor Λ_{P,P_i} er den matrix af egenværdier der indgår i ligning 4.3 for $d_R(P, P_i)$. Λ_{P,P_i} svarer til Λ_2 som den er defineret i algoritme 4.2.

Middelværdien μ for punkterne i S er givet ved

$$\mu = \arg\min_{P \in \mathcal{M}} \rho_A(P). \tag{4.5}$$

Dette minimeringsproblem har en løsning som er entydig, dvs. der findes et globalt minimum. [Fletcher and Joshi, 2004]

Vi vil benytte os af en gradientnedstigningsalgoritme til at løse minimeringsproblemet. Givet en funktion f, som man ønsker at minimere, er en gradientnedstigningsalgoritme generelt givet ved at man ud fra et punkt x_i finder det næste punkt x_{i+1} ved

$$x_{i+1} = x_i - \gamma \nabla f(x_i),$$

hvor γ er skridtlængden og $\nabla f(x)$ er gradienten for f i punktet x. Da funktionen altid aftager mest i den negative gradients retning, vil dette, ud fra et givet startpunkt x_0, give en aftagende følge $f(x_0) \geq f(x_1) \geq f(x_2) \geq \ldots$.

Vores gradientnedstigningsalgoritme går ud fra samme princip, og vi skal derfor tage gradienten af funktionen. Problemet er dog, at gradienten i et punkt $P \in P(n)$ ligger i tangentplanet $T_P P(n)$, og vi skal derfor, efter at have fundet gradienten, afbilde algoritmens skridt fra tangentplanet ned i den Riemannske mangfoldighed. Dette gøres ved brug af Riemann-eksponentialfunktionen. Gradienten af funktionen er givet ved [Fletcher and Joshi, 2004]

$$\nabla \rho_A(P) = -\frac{1}{k} \sum_{i=1}^{k} \mathrm{Log}_{\mathrm{P}}(P_i). \tag{4.6}$$

Vores gradientnedstigningsalgoritme er derfor givet ved algoritme 4.3, hvor ε angiver, hvor præcis vi ønsker at algoritmen skal være. Skridtlængden γ kan bestemmes vha. en

Input: Mængde af kovariansmatricer $A = \{P_1, P_2, \ldots, P_k\}$, hvor $A \subset P(n)$.
Output: Middelværdien $\mu \in P(n)$ for kovariansmatricerne.
1 Lad $\mu_0 = I$, identitetsmatricen;
2 **while** $\|x_i\| > \varepsilon$ **do**
3 $\quad x_i = -\gamma \nabla \rho_A(\mu_i) = \frac{1}{k} \sum_{j=1}^{k} \mathrm{Log}_{\mu_i}(P_j)$;
4 $\quad \mu_{i+1} = \mathrm{Exp}_{\mu_i}(x_i)$;
5 $\quad i = i + 1$;
6 **end**

Algoritme 4.3: Algoritme til at udregne middelværdien af kovariansmatricer $P_1, P_2, \ldots, P_k$ [Fletcher and Joshi, 2004].

linjeafsøgning. I appendiks A.9 giver vi en beskrivelse af linjeafsøgningsmetoden "Gyldne Snit Søgning", som vi har benyttet i vores implementering.

4.2 Klassificeringsalgoritme

I dette afsnit vil vi beskrive en algoritme til klassificering på baggrund af Riemannsk geometri beskrevet i Barachant et al. [2012]. Kovariansmatricer indeholder information om den spatiale struktur af EEG-dataen, og andre BCI-klassificeringsalgoritmer, såsom Common Spatial Pattern (CSP) [Blankertz et al., 2008], benytter sig af kovariansmatricer til at lave features ud fra, som beskrevet i appendiks A.8. I modsætning til dette, bruger denne algoritme i stedet kovariansmatricerne direkte som features. Den centrale idé i algoritmen er at den Riemannske afstand for en måling til middelværdien af træningsdataen for en klasse kan bruges til at måle dets ligheder med træningsdataen, og klassifikationen er derfor gjort ved at vælge den klasse, som minimerer denne distance.

Vi kan derfor se afstanden mellem SCM'erne for forskellige data i mangfoldigheden af SPD-matricer som et udtryk for, hvor ens den spatiale struktur af de forskellige data er. For at klassificere er idéen derfor at finde afstanden på mangfoldigheden fra input-dataens kovariansmatrix til de forskellige klassers middel-kovariansmatricer, og klassificere ud fra hvilken klasse der er tættest på kovariansmatricen for det givne input-data, dvs. har minimal afstand.

For at klassificere et nyt input, som er givet ved inputtets kovariansmatrix P, ønsker vi at finde middelværdien for hver af vores klasser. Hvis en klasse har data der svarer til kovariansmatricerne $P_1, P_2, \ldots, P_N$, kan vi bruge gradientnedstigningsalgoritmen givet ved algoritme 4.3 til at bestemme middelværdien for disse.

Efter at have fundet middelværdien for de enkelte klasser, kan vi nu give en klassificeringsalgoritme baseret på Riemannsk geometri. Algoritmen tager en trial som skal klassificeres som input og udregner derefter afstanden fra inputtet til alle klassernes middelværdier og klassificerer efter hvilken klasses middelværdi, der er tættest på inputtet. Algoritmen er givet i algoritme 4.4.

Input: Trial T der skal klassificeres.
Input: En mængde af N trials T_i af K forskellige klasser.
Input: K mængder af indekser $\mathcal{I}_k$ som hver indeholder indekserne til de trials, der hører til den k'te klasse.
Output: Den estimerede klasse $\hat{k}$ af input T.

for $i = 1$ *to* N **do**
  Find SCM P_i for T_i ved brug af definition 4.2.
end
Find SCM P for T ved brug af definition 4.2;
for $k = 1$ *to* K **do**
  Brug algoritme 4.3 til at finde middelværdien μ_k for alle P_i hvor $i \in \mathcal{I}_k$.
end
$\hat{k} = \arg\min_k d_R(P, \mu_k)$;

Algoritme 4.4: Klassificeringsalgoritme baseret på Riemannsk geometri [Barachant et al., 2012].

5

Design af programmel

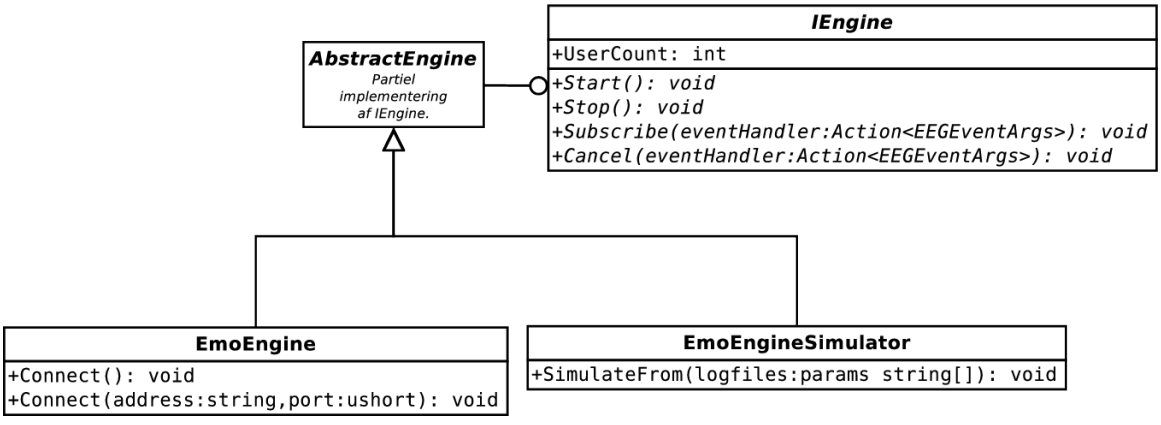
5.1 Arkitektur

I dette afsnit beskriver og diskuterer vi den valgte arkitektur for vores software. Desuden vil vi også redegøre for, hvordan vi har integreret de væsentligste komponenter. Vi forholder os udelukkende analytisk til arkitekturen og integrationen i dette afsnit, og henviser derfor til afsnit 5.2 for implementeringsdetajler. Ydermere tillader vi os at undlade detajler, der ikke er nødvendige for at forstå sammenspillet mellem komponenterne.

Da vi har skrevet vores software i programmeringssproget C# benytter vi C#-konventioner. Dvs. alle interfaces benytter et stort I som præfiks, og alle metodenavne og attributer er skrevet i Pascal-case.

5.1.1 EEG-motor

Vi har defineret en abstrakt beskrivelse af en EEG-motor. Vi har anvendt denne beskrivelse til at implementere en konkret motor, der processerer data fra Emotiv headsettet. Figur 5.1 giver en UML-repræsentation af motor-interfacet. Den konkrete



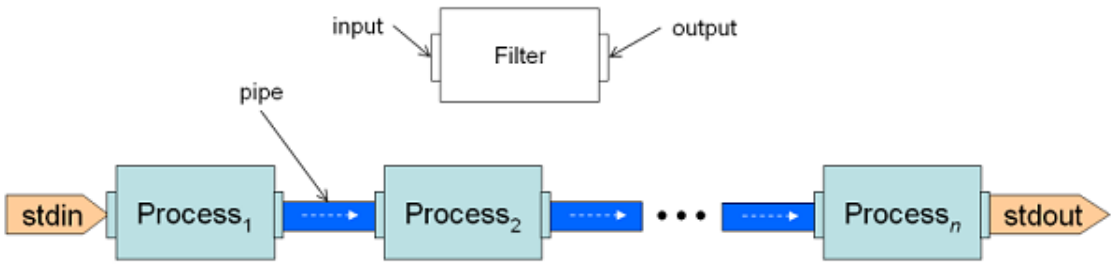
Figur 5.1: UML-repræsentation af *IEngine*-interfacet.

implementering af Emotiv-motoren gør brug af en række designmønstre, hvoraf det

vigtigste er *Observermønstret*. Emotiv headsettet sender diskrete blokke af data med få millisekunders mellemrum. Tiden mellem hver blok af data er så kort, at det giver illusionen af en kontinuert strøm af data. Motoren benytter Observermønstret til at sende en besked til alle abonnenter om at ny data fra headsettet er tilgængeligt. Faktisk pakker motoren dataen ind i beskeden, og selve dataen skubbes derfor også ud til alle abonnenter. Dermed har vi en event-dreven arkitektur, der tillader os at have en løs kobling mellem motoren og eventuelle abonnenter. Desuden giver den event-drevne arkitektur god mulighed for at skalere systemet. Det er muligt at tilslutte flere abonnenter uafhængigt af hinanden til motoren. Det kunne eksempelvis være anvendeligt, hvis man ønsker at afvikle to forskellige ting på samme data samtidigt.

5.1.2 Præprocessering af EEG-data

Headsettet laver en støjfiltrering af signalet før det modtages og behandles af motoren. Vi laver også selv filtrering på den modtagne data før vi giver den som input til en klassifikationsalgoritme. Da vi ønsker at foretage eksperimenter på effekten af forskellige filtre, har det været vigtigt at have en nem måde at integrere og fjerne dem. Derfor benytter vi designmønstret *Pipes & Filters* [Yacoub, 2001], som er et integrationsmønster. Figurativt kan mønstret ses som et stort rør, hvor data bliver skudt ind i den ene ende. Gennem røret bliver dataen behandlet, hvorefter det kommer ud i den anden ende. Figur 5.2 illustrerer netop dette. Mønstret benyttes ofte når en hændelse aktiverer en følge

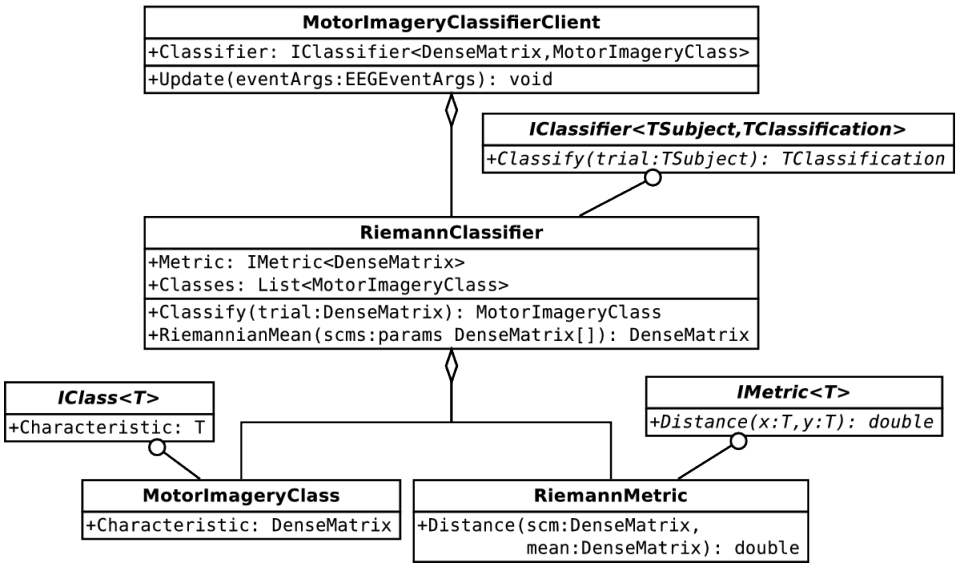


Figur 5.2: Visuel beskrivelse af Pipes & Filters designmønstret [Ortiz, 2013].

af behandlingsskridt, der hver især udfører en veldefineret og specifik funktion [Yacoub, 2001]. Det fungerer dermed godt sammen med vores event-drevne arkitektur, da enhver hændelse (EEG-data) skal gennemgå en række forskellige filtreringer.

5.1.3 Klassificering af EEG-data

En klassificeringsklient indeholder en metode `Update`, således at den kan abonnere på hændelser fra EEG-motoren. Figur 5.3 giver en UML-repræsentation af en klassificeringsklient. Desuden aggregerer klienten også et klassificeringsobjekt, som indeholder klassificeringslogikken. Selve klassificeringsalgoritmen kaldes via metoden `Classify`. Klassificeringsobjektet aggregerer en metrik og en mængde af klasser. Eksempelvis indeholder `RiemannClassifier` klasser af typen `MotorImageryClass`, som hver især har en attribut tilknyttet, der karakteriserer klassen. En konkret attribut kan for eksempel være middelværdien for en motorisk klasse. Med dette design kan vi generalisere vores implementering af Riemann-klassificeringsalgoritmen, således den kan arbejde med arbitrært mange klasser. På intet tidspunkt binder vi os op på antallet af klasser. Desuden binder vi os heller ikke til *hvad*

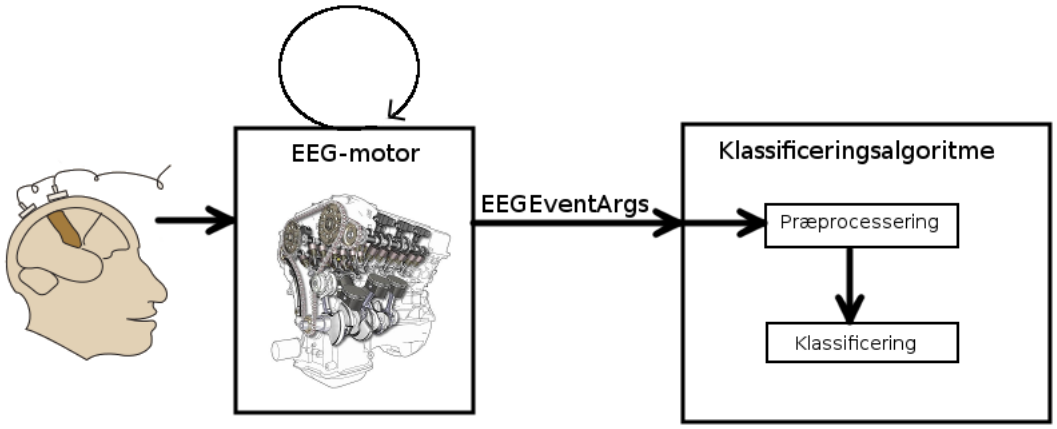


Figur 5.3: UML-repræsentation af klassificeringsklientklassen.

der karaktiserer en klasse. I afsnit 4.1 siger vi, at en motorisk klasse er karaktiseret ved sin middelværdi, men egentlig kan vi med dette design vælge en anden måde at karakterisere en klasse på uden at skulle ændre hele designet.

5.1.4 Fra tanke til klassificering

Figur 5.4 giver et skematisk overblik over flowet mellem enhederne. EEG-motoren undersøger regelmæssigt sine interne buffere efter ny data fra headsettet. Hvis der findes ny data, så genereres en hændelse, som indkapsler de seneste data fra headsettet. Alle abonnenter bliver herefter gjort opmærksom på, at der er ny data tilgængelig. Dataen skubbes ud til abonnenterne via hændelsesobjektet `EEGEventArgs`. Når en abonnent



Figur 5.4: Et skematisk overblik over det samlede program.

modtager en hændelse fra EEG-motoren starter præprocesseringen af hændelsesdataen, hvorefter dataen er klar til at blive klassificeret. Det er op til implementøren at bestemme, hvad er skal ske med klassificeringen af EEG-dataen. Hvis man anskuer vores program

som en samlet komponent, så er det en generisk komponent der kan indgå i funktions- eller servicelaget i en applikation, der benytter EEG-data.

5.2 Implementeringsdetajler

I dette afsnit viser og diskuterer vi en delmængde af vores kildekode[1].

5.2.1 EmoEngine wrapper

Vores implementering af EEG-motoren er faktisk en wrapper rundt om det C++/C# API der følger med Emotiv Research Edition. Formålet med wrapperen er at skjule detajler vedrørende data polling og trådprogrammering. Kildekode 5.1 viser funktionen `PollingAbstraction`, hvis formål er at forespørge den underliggende Emotiv-motor om ny data fra headsettet.

Kildekode 5.1: Data polling

```
public partial class EmoEngine : IEngine {
  protected override void PollingAbstraction() {
    try {
      lock (ConnectedUsers) {
#1      foreach (var user in ConnectedUsers) {
          // Forespoerg om evt. ny data
#2        Dictionary<EdkDll.EE_DataChannel_t, double[]> data = engine.GetData((uint)user);
          if (data != null) {
            uint samplingRate = engine.DataGetSamplingRate((uint)user);
            /* Transformation af datasamling fra
             * Dictionary<EdkDll.EE\_DataChannel\_t,double[]>
             *   til
             * Dictionary<ChannelName,double[]> */
            var transformedData = DataTransformer.Transform(data);
            // Skub haendelsen ud til abnonnenterne
#3          notify(new EEGEventArgs(transformedData, (int)user, samplingRate));
          }
        }
      }
    } catch (Exception e) {
      /* ... */
    }
  }
}
```

Instansvariablen `ConnectedUsers` #1 indeholder en liste af identifikationsnumre på aktuelt forbundne headset. For hvert headset forespørges der om ny data #2. Variablen `engine` indeholder en reference til den underliggende motor, som Emotiv API'et giver adgang til. Funktionen `GetData` fra Emotiv API'et returnerer en hashtabel med kanalnavne som nøgler, og et array af doubles som værdier. Arrayet indeholder x antal nye målinger for den givne kanal siden senest forespørgsel. Hvis der ikke er nogle nye målinger siden den seneste forespørgsel, så returneres `null`. Vi laver derfor et null-tjek af returværdien, fordi vi er ikke interesseret i at sende mulige null-værdier ud til abonnenterne, da det ville introducere unødvendige null-tjek i alle abonnenter. Ved #3 genererer vi en hændelse, som sendes ud til alle abonnenterne.

5.2.2 Pipes & Filters

Vi har implementeret Pipes & Filters designmønstret vha. C# generics, hvilket er medvirkende til at vi kan sammensætte heterogene filtre. Dvs. typen af output objektet

[1] Den fulde kildekode er tilgængelig via http://www.dhil.net/public/edu/aau/d506e13/

fra pipelinen behøver ikke nødvendigvis at være den samme som input objektet. Selve implementeringen er simpel, og udnytter fleksibiliteten i generiske typeparametre og udvidelsesmetoder i C#. Kildekode 5.2 giver en oversigt over vores implementering af mønstret.

Kildekode 5.2: IFilter og extension methods

```
#1 public interface IFilter<in TIn, out TOut> {
     TOut Invoke(TIn data);
   }

#2 public class Pipe<TIn, TCommon, TOut> : IFilter<TIn, TOut> {
    // Instansvariabler
    private IFilter<TIn, TCommon> source;
    private IFilter<TCommon, TOut> destination;
    /*
     *  Instantieringslogik undladet her...
     **/
     // Kaldslogik
     public virtual TOut Invoke(TIn data) {
       if (source == null || destination == null)
          throw new Exception("The pipe isn't proper initialized, needs both source & dest.");
#3     return destination.Invoke(source.Invoke(data));
     }
   }

   // Udvidelsesmetoder (extension methods) til interfacet IFilter
   public static class FilterExtensions {
     // Samling af to filtre til et
#4   public static IFilter<TIn, TOut> Join<TIn, TCommon, TOut>(this IFilter<TIn, TCommon> source,
                                                                IFilter<TCommon, TOut> filter) {
       return new Pipe<TIn, TCommon, TOut>(source, filter);
     }
   }
```

Interfacet `IFilter` #1 har en metode `Invoke`. Metoden tager et objekt af typen `TIn` og returnerer et objekt af typen `TOut`, som begge er generiske typer. Det er op til en implementerende klasse, at bestemme hvordan`Invoke` skal realiseres, og dermed også bestemme in- og outputtyperne.

Klassen `Pipe` #2 repræsenterer sammensætningen af to filtre. For at to vilkårlige filtre er kompatible, skal de have den generiske type `TCommon` til fælles. Faktisk *skal* outputtypen af det første filter (*source*) være inputtypen til det andet filter (*destination*). `Pipe` simulerer et filter ved først at kalde `Invoke` på instansvariablen `source` med `data` som aktuel parameter #3, og herefter give resultatet heraf som aktuel parameter til `Invoke` på instansvariablen `destination`. På denne måde bliver det oprindelige inputobjekt `data` transformeret gennem et rør.

Udover metoden `Invoke`, så er metoden `Join` #4 også tilgængelig for et filter. `Join` forbinder to filtre ved at lave en konkret instans af klassen `Pipe`. Vi har valgt at implementere denne metode som en udvidelsesmetode. Vi kunne have valgt at placere metoden i interfacet `IFilter`, men det ville kræve at enhver implementerende klasse skal give en implementering af metoden. Det vil ikke være hensigtsmæssig, da vi så vil have koderedundans. For at løse den problemstilling kunne vi have valgt at have en abstrakt klasse mellem interfacet og de konkrete filtre, som giver en fælles implementering af metoden. Men så tvinger vi alle filtre til at nedarve fra samme klasse, og vi vil skulle arbejde på den abstrakte klasses abstraktionsniveau og ikke det øverste abstraktionsniveau som interfacet `IFilter` giver.

5.2.3 Implementering af algoritmerne

Vi har valgt at implementere algoritmerne 4.1, 4.2, 4.3 og 4.4 i Matlab, da algoritmerne kræver en del tunge matrixberegninger. Da vi ikke har kunnet finde et godt open-source C#-bibliotek til lineær algebra, valgte vi at embedde Matlab-koden for algoritmerne i C#-koden. Matlab er de facto standardværktøj til matrixberegninger og understøtter derfor de regneværktøjer indenfor lineær algebra, som vi skal bruge. Kildekode 5.3 viser vores implementering af algoritme 4.1.

Kildekode 5.3: Implementering af algoritme 4.1

```
function [ Exp ] = expmap(P, X)
 % EXPMAP Exponential map from tangent space to manifold
 % input:
 % P : point in P(n)
 % X : tangent vector in S(n)

#1  [u,Lambda] = eig(P);

#2  p_sqrt = u*diag(sqrt(diag(Lambda)))*u';
    p_sqrt_inv = u*diag(diag(Lambda).^(-1/2))*u';

    pXp = p_sqrt_inv * X * p_sqrt_inv;

    [v,Sigma] = eig(pXp);

#3  exped = v*diag(exp(diag(Sigma)))*v';

    Exp = p_sqrt * exped * p_sqrt;
end
```

Kildekoden gengiver stort set algoritme 4.1 til punkt og prikke. Ved `#1` laves egenværdidekomponeringen af inputpunktet $P \in P(n)$. Funktionen `eig` laver dekomponeringen, hvorefter resultatet bliver tildelt variablerne `u` og `Lambda` vha. pattern-matching. Variablen `u` får tildelt den ikke-inverterede matrix med egenvektorer i søjlerne, mens `Lambda` får tildelt diagonalmatricen, hvis diagonal består af egenværdierne. Matlab behandler desværre ikke eksponentiering af diagonalmatricer korrekt for funktionerne `sqrt` og `exp`. I stedet for at lade funktionerne virke på diagonalen, så lader Matlab dem virke på alle indgangene. Derfor benytter vi funktionen `diag` `#2` på `Lambda` til at udtrække diagonalen som en vektor, før vi benytter `sqrt`. For kvadratrodsfunktionen `sqrt` er det ikke et problem, at Matlab lader den virke på alle indgangene i diagonalmatricen, da $\sqrt{0} = 0$, men for `exp` `#3` bliver det straks værre da $\exp(0) = 1$. Hvis vi ikke brugte `diag`, så ville det resultere i en matrix, hvor diagonalen ville være korrekt, men de resterende indgange alle ville indeholde 1.

6

Eksperimenter med EEG-data

Vi har gennemført en række eksperimenter med vores programmel, der klassificerer data ud fra afstanden mellem kovariansmatricen for dataen og middelværdien for klasserne i et Riemannsk rum, som beskrevet i afsnit 4.2. Eksperimenterne er udført på to forskellige datasæt, hvoraf det ene er egne data, mens det andet er fra BCI Competition IV. I afsnit 6.1 giver vi en beskrivelse og analyse af eksperimenter med BCI Competition IV-datasættet[1], mens vi i afsnit 6.4 behandler vores egne data.

6.1 BCI Competition data

BCI Competition IV (BCIC) tilbyder en række BCI datasæt, der består af forskellige motoriske klasser. Vi har benyttet datasæt 2a, som består af fire motoriske klasser:

Klasse 1: Venstre hånd.

Klasse 2: Højre hånd.

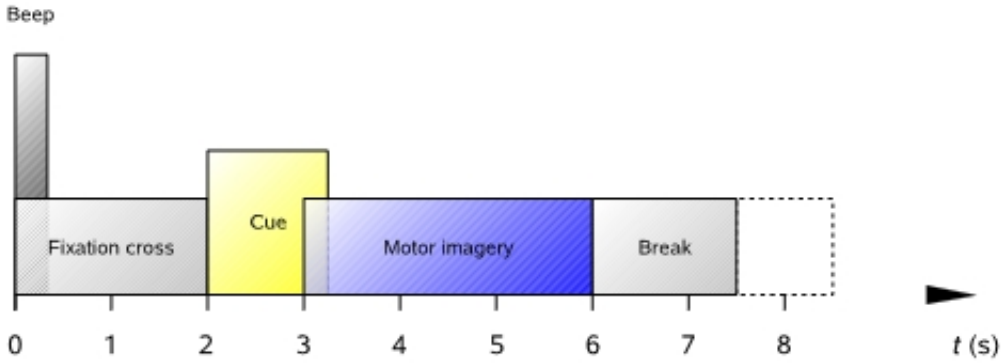
Klasse 3: Begge fødder.

Klasse 4: Tungen.

Formålet med at lave eksperimenter med BCIC datasættet er, at verificere vores implementering af Riemannklassifikationsalgoritmen. Desuden benytter Barachant et al. [2010] også datasæt 2a fra BCI competition IV. Derfor benytter vi samme datasæt, således vi kan sammenligne vores resultater med resultaterne fra artiklen.

Datasættet er baseret på data fra ni testpersoner. Hver testperson fik foretaget målinger på to forskellige dage. Hver dag bestod af af seks forløb med korte pauser i mellem, hvor hvert forløb bestod af 48 målinger eller trials, inddelt så hver af de fire forskellige klasser fik 12 målinger per forløb. [Brunner et al., 2008]

[1] www.bbci.de/competition/iv/



Figur 6.1: Forløbet for en måling i BCI competition IV 2a datasættet [Brunner et al., 2008].

En måling forløb som følger. Testpersonen sad i en lænestol foran en sort computerskærm, og ved starten af hver måling lød der en kort advarselstone samtidig med at et kryds fremkom på skærmen, som testpersonen skulle fokusere på. To sekunder efter at målingen var startet, dukkede der en pil op på skærmen, som enten kunne pege op, ned, til højre eller til venstre, og som indikerede at testpersonen skulle tænke på at bevæge henholdsvis tungen, fødderne, højre hånd eller venstre hånd. Pilen blev på skærmen i 1,25 sekunder, men testpersonen skulle blive ved med at tænke på denne handling indtil krydset forsvandt fra skærmen, hvilket skete seks sekunder efter at målingen var påbegyndt. Efter seks sekunder forsvandt krydset, og skærmen blev sort, hvilket gav testpersonen en kort pause inden næste måling. Hele dette forløb er illustreret i figur 6.1. [Brunner et al., 2008] På figuren kan ses et mellemrum på et sekund fra *cue* til *motor imagery*, hvilket blot angiver at der går noget tid, ikke nødvendigvis et sekund, fra indikationen fremkommer på skærmen til at testpersonen reagerer og begynder at tænke på handlingen.

Målingerne af EEG-signaler blev foretaget med 22 elektroder med en samplingsrate på 250 Hz. Efterfølgende blev signalerne båndpasfiltreret mellem 0,5 og 100 Hz samt spærrefiltreret på 50 Hz for at fjerne støj.

6.2 BCIC eksperiment: Metrikker (2 klasser)

I dette eksperiment undersøger vi korrektheden af klassifikationer under to forskellige metrikker: den Euklidiske og Riemannske metrik. I eksperimentet benytter vi kun højre- og venstreklassen fra BCIC datasættet.

6.2.1 Formål & eksperimentbeskrivelse

Vi ønsker at verificere vores implementering og sammenligne resultaterne med resultaterne fra Barachant et al. [2010]. I artiklen laver de et eksperiment, hvor de undersøger klassificering under to metrikker. Vi forsøger at efterligne dette eksperiment. Derfor er vores forventning til eksperimentet også i lighed med resultaterne i Barachant et al. [2010].

For hver metrik trænes klassificeringsalgoritmen på et datasæt. Det er det samme datasæt for begge metrikker. En træning resulterer i en middelværdi per klasse. For enhver måling

beregnes afstanden mellem målingen og de respektive klassers middelværdier. Den mindste afstand bestemmer klassificeringen af målingen.

Eksperimentet er inddelt i to dele. I den første del trænes klassificeringsalgoritmen for én testperson ad gangen, og evalueres derefter for samme testperson. I den anden del trænes klassificeringsalgoritmen derimod på data fra alle testpersoner på én gang, og evalueres derefter enkeltvis for hver testperson.

> **Hypotese 6.1**
> Vi forventer, at klassificering ved den Riemannske metrik resulterer i markant færre fejlklassificeringer end ved den Euklidiske metrik.

6.2.2 Dataanalyse

I dette afsnit vil vi fremlægge vores resultater og kommentere på nogle af resultaterne. Først behandler vi scenariet, hvor klassificeringsalgoritmen er trænet individuelt per testperson, hvorefter vi behandler tilfældet, hvor algoritmen er trænet på data fra alle testpersoner.

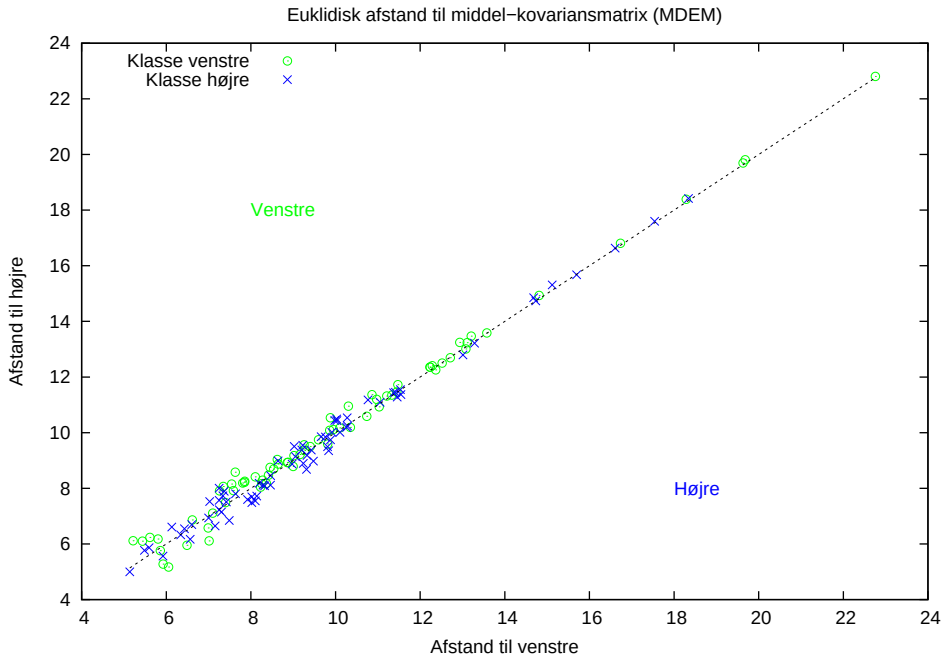
Træning på individuel data

Figur 6.6 giver en visuel repræsentation af resultaterne fra et eksperiment med den Euklidiske metrik. Dataen der ligger til grund for figuren er fra BCIC-sættet for forsøgsperson 1. Den stiplede linje i figuren repræsenterer beslutningsgrænsen, der er afgørende om en måling klassificeres som højre eller venstre. Cirklerne repræsenterer målinger, der *skal* klassificeres som venstre, mens krydserne repræsenterer målinger der *skal* klassificeres som højre. Det er tydeligt at se, at der er mange fejlklassificeringer. De fleste af målingerne ligger enten på eller lige i omregnen af beslutningsgrænsen. Der er ikke nogen klar adskillelse af klassificeringerne.

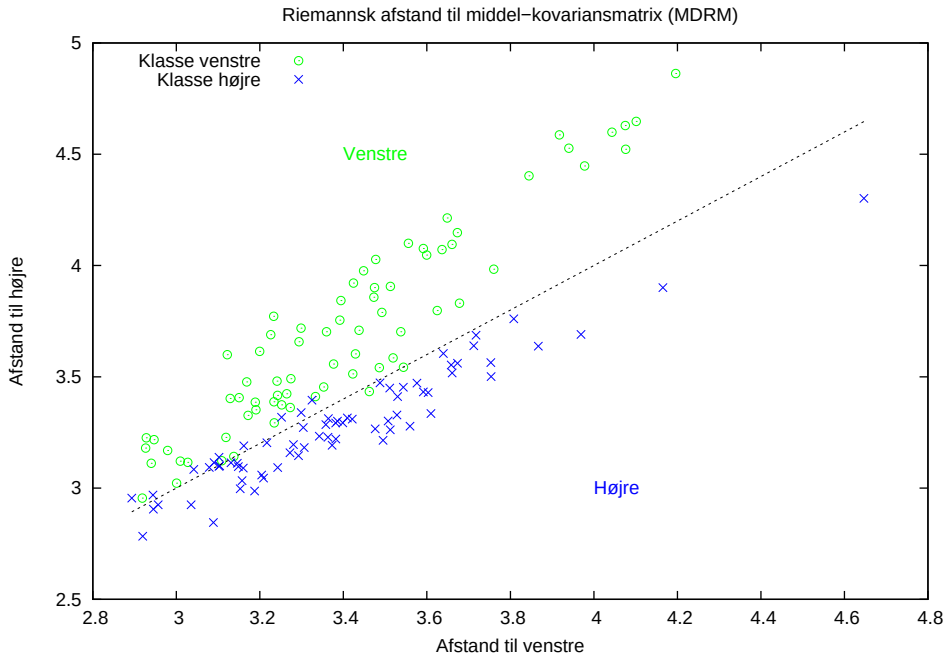
Figur 6.3 viser derimod resultaterne fra eksperimentet med den Riemannske metrik. Vi har her en hel klar adskillelse af klassificeringerne og tydeligvis langt færre fejlklassificering. Der er umiddelbart meget få venstremålinger, der bliver klassificeret som højre, mens der er nogen flere højremålinger der bliver klassificeret som venstre, hvilket giver indtrykket af at vores klassificeringsalgoritme er bedre til korrekt at klassificere venstre fremfor højre.

Tabel 6.1 viser effektiviteten af den Riemannske metrik kontra den Euklidiske metrik per testperson. "Middel %"-kolonnerne angiver de gennemsnitlige korrekte klassificeringer i procent per testperson, mens kolonnen "Forskel (pp)" angiver forskellen mellem de gennemsnitlige korrekte klassificeringer i procentpoint (pp). Resultaterne fra testperson 8 og 9 er bemærkelseværdige. Her oplever vi for begge personer en stigning i korrekte klassificeringer på ≈ 40 pp under den Riemannske metrik. Vi kan se, at antallet af korrekte klassificeringer af venstre er meget lav under den Euklidiske metrik, og at der er stor procentvis forskel mellem antallet af korrekte klassificeringer af venstre og højre. Denne forskel bliver for begge testpersoner udlignet under den Riemannske metrik. Særligt for testperson 8 får vi omtrentlig en lige god procentvis klassificering af højre og venstre.

Derimod oplever vi for testperson 2, 5 og 7 "kun" en stigning på ≈ 10 pp. Testperson 2 og 5 er hver med en stigning på 9,7 pp de mindste stigninger vi har oplevet. For testperson 5 er



Figur 6.2: Klassificering af højre og venstre under den Euklidiske metrik for testperson 1.



Figur 6.3: Klassificering af højre og venstre under den Riemannske metrik for testperson 1.

der både stor forskel mellem de procentvise korrekte klassificeringer under den Euklidiske og Riemannske metrik. Hvor vi før så, at forskellen blev udlignet ved anvendelse af den Riemannske metrik, så lader den til at have en meget lille betydning her. Problemet kan ligge i testdataen. Det er muligt at testpersonen har svært ved at adskille højre og venstre, således at hjernen udsender et lignende signal for begge klasser. Derved bliver klassernes overlap større, og derfor forekommer der flere fejlklassificeringer. Testperson

	Euklid			Riemann			
Person	Venstre %	Højre %	Middel %	Venstre %	Højre %	Middel %	Forskel (pp)
1	75,0	50,7	62,9	97,2	85,9	91,6	+28,7
2	80,6	21,1	51,1	76,4	45,1	60,8	+9,7
3	76,1	59,7	67,8	84,5	100,0	92,3	+24,5
4	65,3	43,1	54,2	63,9	79,1	71,5	+17,3
5	13,7	84,5	48,6	21,9	95,8	58,3	+9,7
6	50,7	56,3	53,5	69,9	73,3	71,5	+18,0
7	34,7	76,4	55,6	44,4	90,3	67,4	+11,8
8	26,4	81,9	54,2	97,2	95,8	96,5	+42,3
9	24,7	76,1	50,0	87,7	95,8	91,7	+41,7
Middel	49,7	61,1	55,3	71,5	84,6	78,0	+22,7

Tabel 6.1: Trænet kun på data for hver enkelt testperson.

2 har med den Euklidiske metrik ca. samme totale success for klassificeringerne som testperson 8 bortset fra at fejlklassificeringsprocenten er byttet om for højre og venstre. I modsætning til testperson 8 har testperson 2 dog kun et 9,7 pp fald i fejlklassificeringer når der skiftes til den Riemannske metrik, mens for testperson 8 er faldet på 42,3 pp. En forbedring i klassificeringer kan derfor ikke forudses ud fra forskellen mellem antallet af fejlklassificeringer på højre og venstre med den Euklidiske metrik. Testpersonerne 1, 3, 4 og 6 har alle en stigning inden for 10 pp af de 22,7 pp, som er den gennemsnitlige stigning. Fælles for både testperson 1, 3, 4 og 6 er også, at de står for de fire mindste forskelle mellem antallet af fejlklassificeringer på højre og venstre i den Euklidiske metrik med højst 25 pp i forskel. Der syntes derfor at være en tendens til at større forskelle mellem antallet af fejlklassificeringer på højre og venstre i den Euklidiske metrik giver større udsving på forbedringen under den Riemannske metrik.

Træning på aggregeret data

	Euklid			Riemann			
Person	Venstre %	Højre %	Middel %	Venstre %	Højre %	Middel %	Forskel (pp)
1	16,7	88,7	52,5	100,0	26,8	63,6	+11,1
2	80,6	21,1	51,0	80,6	23,9	52,5	+1,5
3	38,0	56,9	47,6	98,6	84,7	91,6	+44,0
4	1,4	100,0	50,7	63,9	80,6	72,2	+21,5
5	9,6	91,6	50,0	98,6	1,4	50,7	+0,7
6	42,5	53,5	47,9	2,7	100,0	50,7	+2,8
7	6,9	93,0	50,0	80,6	63,9	72,2	+22,2
8	98,6	5,6	52,1	77,8	97,2	87,5	+35,4
9	35,6	60,6	47,9	13,7	100,0	56,3	+8,4
Middel	36,7	63,4	50,0	68,5	64,3	60,7	+10,7

Tabel 6.2: Trænet på aggregeret testdata for alle personer, testet på individer.

Tabel 6.2 viser effektiviteten af den Riemannske metrik kontra den Euklidiske metrik ved aggregeret testdata. Ligesom i tabel 6.1 angiver "Middel %"-kolonnen de genemsnitlige korrekte klassificeringer i procent per testperson og "Forskel (pp)"-tabellen angiver forskellen mellem de gennemsnitlige korrekte klassificeringer i pp. Med den Euklidiske

metrik er den aggregerede data kun 5,3 pp lavere end på den individuelle data. Den væsentlige forbedring fra aggregeret data til individuel data ses dog under den Riemannske metrik, hvor der er en forbedring på 12 pp. I den aggregerede testdata har antallet af fejlklassificeringer det med at stige væsentligt for den ene klasse og falde for den anden mellem den Euklidiske og Riemannske metrik, hvor fejlklassificeringer for den individuelle testdata aldrig stiger med mere end 5 pp fra den Euklidiske til den Riemannske metrik. Derudover er et tilfælde som testperson 5 værd at fremhæve, hvor der for den individuelle data er meget lav fejlklassificering for højre og relativ høj fejlklassificering for venstre, men omvendt efter træning på den aggregerede data. Dette kan skyldes, at højre- og venstreklasserne er meget overlappende klasser hos denne testperson, hvormed det bliver svært for algoritmen at kende forskel.

6.2.3 Sammenligning af resultater

I dette afsnit vil vi sammenligne vores resultater på BCIC datasættet med resultaterne givet i Barachant et al. [2012]. Vores resultater benytter kun to af klasserne i BCIC datasættet, mens resultaterne i Barachant et al. [2012] benytter alle fire klasser, så vores sammenligning kan ikke bruges til at konkludere, om den ene eller den anden implementering virker bedre. Vi vil i stedet benytte denne sammenligning til at se, om der er en vis overensstemmelse mellem de to implementeringer af algoritmen, med det forbehold at resultaterne kunne være væsentligt anderledes hvis vi brugte alle fire klasser.

Person	MDRM
1	77,8
2	44,1
3	76,8
4	54,9
5	43,8
6	47,1
7	72,0
8	75,2
9	76,6
Middel	63,2 ± 15,2

Tabel 6.3: Præcision for klassifikation i 30-folds kryds-validering [Barachant et al., 2012].

Tabel 6.3 viser Barachant et al. [2012]'s resultater for algoritmen Minimum Distance to Riemannian Mean (MDRM), som Riemann-algoritmen i denne rapport bygger på. Resultatet her vises dog kun som det procentmæssige gennemsnit af korrekte klassificeringer for alle dele af datasættet. Da de ikke er delt ud i korrekte klassificeringer for højre og venstre kan det være svært præcis at bestemme, om de to algoritmer har samme problemer. For det meste syntes vores algoritme at have samme udsving som MDRM. Testperson 1, 3, 8 og 9 ligger alle med 75% eller højere grad af korrekte klassificeringer og 90% eller højere for vores, hvilket i begge algoritmers tilfælde er langt over gennemsnit. Testperson 2 og 5 har under 45% korrekte klassificeringer for MDRM og omkring 60% for os, hvilket igen i begge tilfælde er langt under gennemsnit. Testperson 4 ligger for begge algoritmers vedkommende kun en smule under gennemsnittet.

Det er kun for testperson 6 og 7, at resultatet i forhold til gennemsnittet afviger væsentligt fra hinanden for algoritmerne. Testperson 6 har for MDRM kun 47,1% korrekte klassificeringer, hvor vores har 71,5% korrekte klassificeringer. I begge tilfælde ligger det under middel, men for MDRM afviger det med 25 pp, hvor vores kun afviger med 8 pp. Testperson 7 har med MDRM 72% korrekte klassificeringer, hvilket foruden at være væsentligt over MDRM's middel, også er den eneste testperson, der klassificeres bedre af MDRM end af vores algoritme. For vores algoritme er det kun 67,4% af klassificeringerne der er korrekte, hvilket er et stykke under middel for den.

For gennemsnittet syntes vores algoritme at være en væsentligt forbedring af MDRM med en middel på 78% korrekte klassificeringer sammenlignet med MDRM's 63,2%. MDRM har højest 15,2 pp (24%) afvigelse fra middel, hvor vores algoritme højest har 19,7 pp (25%) afvigelse fra middel, så på det punkt er der ingen væsentlig forskel fra den ene til den anden.

6.2.4 Konklusion

Dataanalysen bekræfter vores forventning. Der sker betydeligt færre fejlklassificeringer under den Riemannske metrik. Desuden stemmer vores resultater nogenlunde overens med resultaterne fra [Barachant et al., 2012].

6.3 BCIC eksperiment: Metrikker (3 klasser)

Dette eksperiment er en udvidelse af vores første eksperiment, hvor vi inkluderer en tredje motorisk klasse.

6.3.1 Formål & eksperimentbeskrivelse

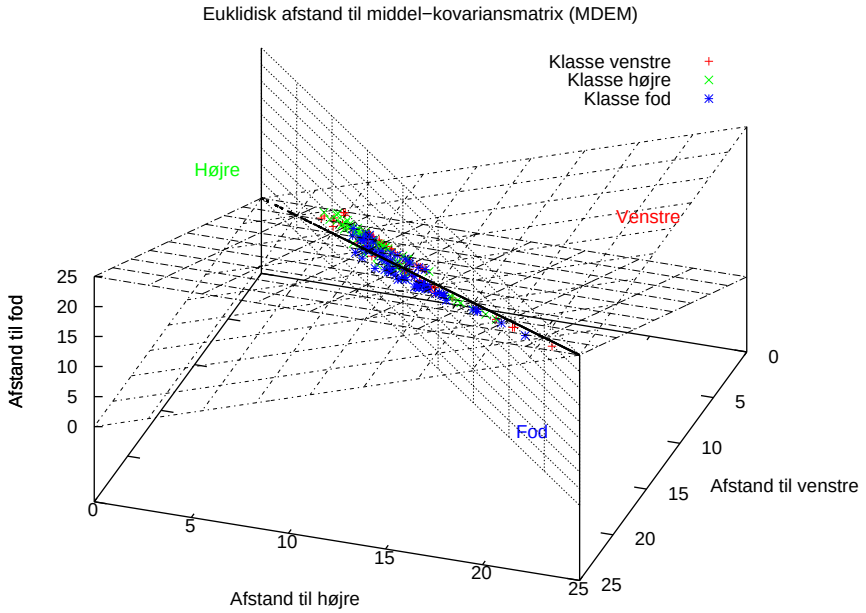
Riemannklassificeringsalgoritmen kan teoretisk håndtere et arbitrært antal klasser. BCIC datasættet indeholder fire motoriske klasser, men vi har valgt kun at se på tre af klasserne, da vi maksimalt kan visualisere tre klasser. Derfor kommer vi ikke til at kunne konkludere, hvordan vores algoritme skalerer i forhold til antallet af klasser, men eksperimentet er interessant ud fra hvad der sker, når vi tilføjer en tredje dimension. Den tredje klasse, som vi har valgt at inkludere, er begge fødder.

Eksperimentet foreløber nøjagtigt som det foregående, dog med den ændring, at vi har inkluderet en tredje klasse. Vi har følgende forventning til eksperimentet.

> **Hypotese 6.2**
> Vi forventer, at klassificeringsalgoritmen er i stand til at håndtere de tre klasser, og dermed have et lavt antal fejlklassificeringer.

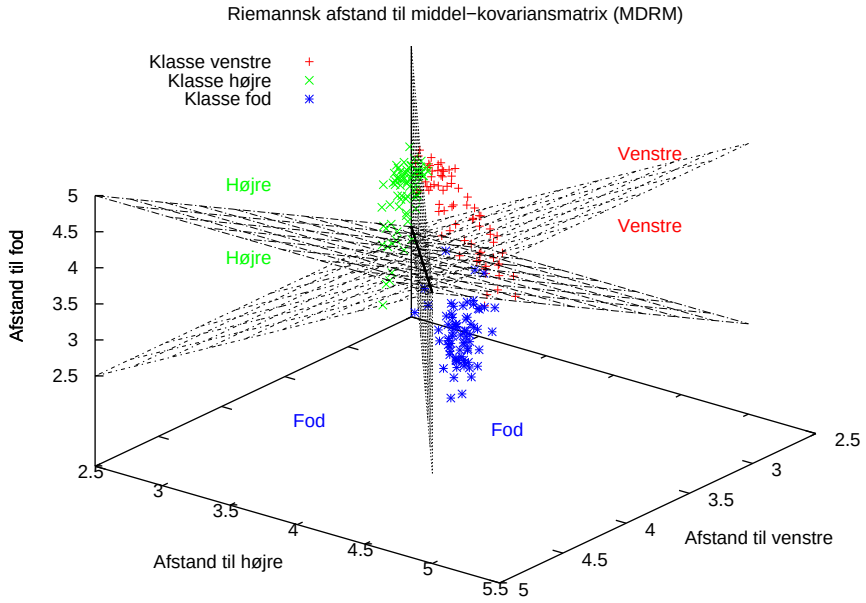
6.3.2 Dataanalyse

Figur 6.4 og 6.5 viser resultaterne under henholdsvis den Euklidiske og Riemannske metrik, igen for forsøgsperson 1 i BCIC-sættet. Resultaterne minder meget om resultaterne fra det foregående eksperiment med to klasser. Algoritmen har svært ved at skelne mellem de tre forskellige klasser under den Euklidiske metrik, som det kan ses på figur 6.4. Umiddelbart



Figur 6.4: Klassificering af højre, venstre og fødder under den Euklidiske metrik for testperson 1.

ser det ikke ud til, at der er en klasse, den bedre kan klassificere end de andre. Den lader til at være lige dårlig for alle klasserne. Under den Riemannske metrik ser vi derimod en klar adskillelse af klasserne, som vist på figur 6.5. Der er dog nogle fejlklassificeringer mellem højre- og venstreklassen og venstre- og fødderklassen.



Figur 6.5: Klassificering af højre, venstre og fødder under den Riemannske metrik for testperson 1.

6.3.3 Konklusion

Vores implementering opfører sig som forventet efter inkluderingen af fødderklassen. Dog kan vi ikke endeligt konkludere hvordan vores algoritme skalerer i forhold til antallet af klasser for et større antal klasser ud fra dette eksperiment.

6.4 Eksperiment med egne data

I dette afsnit vil vi beskrive eksperimenter foretaget med vores klassificeringsalgoritme på data vi selv har optaget med Emotivs headset. Headsettet er beskrevet i afsnit 2.3, og vores data er optaget gennem proceduren beskrevet i afsnit 6.1, dog med den forskel at vores forløb generelt bestod af 100 trials og målingerne blev foretaget i et mindre kontrolleret miljø.

6.4.1 Formål & eksperimentbeskrivelse

Vi foretog to typer målinger: tænkt bevægelse og faktisk bevægelse. I målinger med tænkt bevægelse tænkte forsøgspersonen på at foretage den foreskrevne handling, uden egentlig at udføre den, mens i målinger med faktisk bevægelse foretog forsøgspersonen handlingen. Vi lavede disse to typer af målinger for at undersøge, om der er en væsentlig forskel på algoritmens præcision mellem de to typer. Vi forestiller os at det at kombinere tanke med handling resulterer i flere ens signaler og dermed bedre nøjagtighed, da det kan være svært at tænke på at bevæge højre eller venstre arm uden egentlig at udføre bevægelsen.

Vi har desuden udarbejdet en indlæringskurve for en af testpersonerne, for at se hvor meget data algoritmen kræver for at kunne give ordentlige resultater. Dette er beskrevet i afsnit 6.4.2.

> **Hypotese 6.3**
> Vi forventer, at klassificering ved den Riemanniske metrik resulterer i færre fejlklassificeringer end ved den Euklidiske metrik. Dog forventer vi, at resultaterne er ringere end resultaterne fra BCIC datasættet. Desuden forventer vi, at resultaterne fra "faktisk bevægelse"-datasættet er bedre end resultaterne fra "tænkt bevægelse"-datasættet.

6.4.2 Dataanalyse

Vi præsenterer og diskuterer først resultaterne fra datasættet "tænkt bevægelse", dernæst viser vi resultaterne fra datasættet "faktisk bevægelse" og endeligt undersøger vi indlæringskurven for egne data.

Tænkt bevægelse

I tabel 6.4 ses resultaterne for vores klassificeringsalgoritme brugt på vores data med tænkt bevægelse. Klassificeringen er sket på baggrund af et træningssæt bestående af 198 trials i hver klasse for hver person, og algoritmen er trænet på hver enkelt person før denne persons data blev klassificeret.

	Euklid			Riemann			
Person	Venstre %	Højre %	Middel %	Venstre %	Højre %	Middel %	Forskel (pp)
1	100,0	33,3	66,7	89,9	100,0	95,0	+28,3
2	2,0	96,0	49,0	11,1	90,1	50,6	+1,6
3	56,6	40,4	48,5	5,1	84,9	45,0	-3,5
4	21,2	78,8	50,0	70,7	75,8	73,3	+22,3
Middel	45,0	62,1	53,6	44,2	87,7	66,0	+12,2

Tabel 6.4: Trænet kun på data for hver enkelt testperson.

Eksperimenterne viser at algoritmens nøjagtighed svinger meget fra person til person. Med den Riemannske metrik er algoritmen god til at klassificere for både testperson 1 og 4, mens den er dårlig for testperson 2 og 3. En forklaring på de dårlige resultater for testperson 2 og 3 kan muligvis skyldes, at de har svært ved adskille højre- og venstretanker og dermed bliver det svært for algoritmen at klassificere for disse to. Det kan også ses at algoritmen generelt er bedre med den Riemannske metrik end med den Euklidiske metrik. Dog er den Riemannske metrik faktisk værre for testperson 3, selvom resultaterne for denne person er dårlige både med Euklidisk og Riemannsk metrik.

Algoritmen er desuden bedre til at klassificere højre end venstre klasse i dette datasæt for begge metrikker.

Faktisk bevægelse

Tabel 6.5 viser resultaterne for vores klassificeringsalgoritme brugt på dataen med faktisk bevægelse. Der er brugt 198 trials i hver klasse per person til træningen af klassificeringsalgoritmen, og algoritmen blev trænet på hver testpersons data inden personens data blev klassificeret af algoritmen.

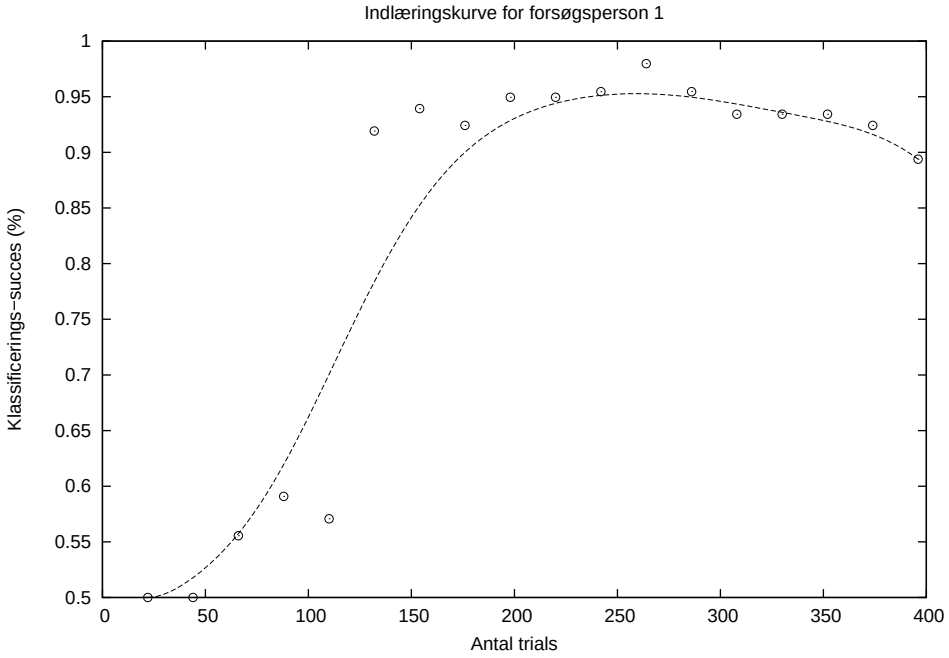
	Euklid			Riemann			
Person	Venstre %	Højre %	Middel %	Venstre %	Højre %	Middel %	Forskel (pp)
1	100,0	100,0	100,0	100,0	99,0	99,5	-0,5
2	0,0	33,3	16,7	99,0	0,0	49,5	+32,8
3	46,5	59,6	53,1	58,6	87,9	73,3	+20,2
Middel	48,8	64,3	56,6	85,9	62,3	74,1	+17,5

Tabel 6.5: Trænet kun på data for hver enkelt testperson.

Disse resultater viser en stor varians i hvor god algoritmen er til at klassificere. For testperson 1 har algoritmen tæt på 100% nøjagtighed både for den Euklidiske og den Riemannske metrik, mens testperson 2 har meget dårlige resultater med begge metrikker. Umiddelbart lader det til signalerne for højre og venstre hos testperson 1 er tilstrækkeligt forskellige, og det derfor er nemt for algoritmen at klassificere under begge metrikker. Dermed vindes der intet ved at klassificere ved Riemann geometri for denne testperson. Dog ses der samlet set en forbedring ved brug af den Riemannske metrik fremfor den Euklidiske.

For testperson 2 klassificeres venstre korrekt 0% af tiden ved brug af den Euklidiske metrik, mens det er højre, der klassificeres korrekt 0% af tiden for samme bruger ved brug af den Riemannske metrik.

Indlæringskurve



Figur 6.6: Indlæringskurve for optaget data. Den stiplede linje er en midlet Bezier-kurve der viser tendensen.

Figur 6.6 viser en indlæringskurve for testperson 1. Som det kan ses begynder grafen allerede at flade ud ved omkring 200 trials efter at være steget kraftigt fra 100. Herefter er der kun mindre svingninger i grafen, så det er begrænset hvor meget bedre klassificering man får per tilføjet trial efter 200. Efter ca. 260 trials viser der sig endda en lille stigning i fejlklassificeringer. Forklaring på stigningen skal måske findes i mangel på koncentration under datalogning, da dataen brugt i denne indlæringskurve er ordnet i tid efter optagelse.

Indtil de første 100 trials stiger grafen ikke meget, men mellem 100 og 150 sker der et meget markant spring op til 93% klassificerings-success, hvor det vil begynde at være brugbart til at løse opgaver som eksempelvis genoptræning. Præcis hvor kurven flader ud kan variere fra bruger til bruger, men ud fra denne indlæringskurve er det realistisk at forvente at en ny bruger vil skulle optage et par hundrede trials før algoritmen vil kunne give gode resultater.

6.4.3 Konklusion

Eksperimenterne i dette afsnit er udført på et meget lille datasæt, så det er ikke muligt at konkludere noget generelt, men vi kan se nogle af de samme tendenser som vi observerede i afsnit 6.2.2. Det ser derfor ud til at dataene fra Emotivs headset kan fungere i praksis som et alternativ til laboratorieudstyr. Desuden viser eksperimenterne på vores begrænsede datasæt at der ikke er stor forskel i nøjagtighed på at klassificere tænkt bevægelse og faktisk bevægelse.

7

Konklusion

Vi vil i dette afsnit konkludere på vores problemformulering, som blev fremsat i afsnit 1.3. Den lød som følger:

Hvordan kan man implementere et stykke software, der ved brug af consumer-grade EEG-teknologi assisterer en patient med genoptræning?

For at besvare dette har vi i kapitel 2 gennemgået dele af hjernens opbygning og inddeling for at få indblik i, hvad EEG-apparater måler.

I kapitel 3 har vi beskrevet BCI-systemer, og hvordan disse kan bruges sammen med EEG-apparater til at genkende forskellige typer af hjerneaktivitet, samt hvordan man forbereder EEG-målingerne til at blive brugt i sådanne systemer.

Herefter har vi i kapitel 4 givet en klassificeringsalgoritme til brug i et BCI-system, som er baseret på Riemannsk geometri. Vi har derfor også i kapitlet beskrevet en del af teorien om Riemannsk geometri.

I kapitel 5 har vi dokumenteret opbygningen af det delvise BCI-system, som vi har implementeret, og i kapitel 6 har vi fremlagt resultaterne af de eksperimenter, vi lavede med vores implementering for at teste implementeringens effektivitet.

På baggrund af vores problemformulering fremsatte vi tre spørgsmål, som vi vil besvare i de næste tre afsnit.

7.1 Indsamling, fortolkning og analysering af EEG-data

Det første af vores tre spørgsmål var

Hvordan kan vi indsamle, fortolke og analysere EEG-data?

Til dette har vi brugt Emotivs headset, som er et consumer-grade EEG-apparat, der kan indsamle EEG-data. Headsettet bruger 10-20 systemet til at beskrive, hvilke dele af skalpen, det måler på.

For at analysere den data, som indsamles med headsettet, benytter vi os af et BCI-system, der lærer en måde at klassificere på ud fra noget data, som den på forhånd ved, hvordan skal klassificeres. Efter at have trænet på dette data, kan systemet så bruges til at klassificere ny data og give feedback til brugeren på baggrund af denne klassificering. Selve klassificeringen foregår ved hjælp af en maskinindlæringsalgoritme, men inden denne algoritme køres på dataen, er det vigtigt at bearbejde dataen først, for blandt andet at fjerne støj. Dette gøres i præprocesseringsfasen.

Den klassificeringsalgoritme, vi har valgt at bruge er baseret på Riemannsk geometri. Ideen i algoritmen er at se på kovariansmatricerne for dataen, og betragte det Riemannske rum, som kovariansmatricerne udgør. I dette rum har vi defineret en afstandsfunktion, som gør os i stand til at beregne afstanden mellem to punkter i rummet, og en metode til at finde middelværdien for en mængde punkter i rummet. På baggrund af dette har vi givet en klassificeringsalgoritme, som udregner middelværdien for kovariansmatricerne for hver klasse. Når klassificeringsalgoritmen så modtager ny data, der skal klassificeres, udregner den kovariansmatricen for den nye data og klassificerer derefter ud fra hvilken klasses middelværdi, der er tættest på den nye datas kovaransmatrix.

7.2 Software

Vores andet spørgsmål var

Hvordan kan man designe softwaren på en hensigtsmæssig måde?

Softwaren brugt til at opbygge de delvise implementering af et BCI-system består overordnet af en EEG-motor, en præprocessor og en klassificeringsalgoritme.

EEG-motoren har en event-dreven arkitektur, hvilket giver mulighed for at have flere abonnenter uafhængigt af hinanden tilsluttet til motoren.

Præprocesseringsdelen bruges til at filtrere dataen modtaget fra headsettet. Her benyttes "Pipes & filters"-integrationsmønstret, som gør det nemt at integrere og fjerne de forskellige filtre for at gøre testning af forskellige filtre så let som mulig.

Klassificeringsalgoritmen bruger herefter den filtrerede information til at lave klassificering med enten en Euklidisk eller Riemannsk metrik.

7.3 Resultater

Det tredje spørgsmål var

Hvilke kriterier er der for succes af softwaren? Og hvordan kan vi måle dem?

Kriteriet for success af software er succesraten for klassificeringer. Det måles for tre forskellige datasæt: BCIC-data, "faktisk bevægelse"-data optaget med Emotiv headsettet og "tænkt bevægelse"-data optaget med Emotiv headsettet. For at forsøge at løse problemet omkring implementation af et stykke software til brug af consumer-grade EEG-teknologi for at assistere en patient med genoptræning er der i rapporten blevet produceret nogle forsøg. Forsøgene har indeholdt to forskellige metrikker til klassificering og en sammenligning af disse for at kunne udvælge den bedste af de to løsninger til problemet.

Resultaterne for den Riemannske metrik giver gennemsnitligt bedre resultater end den Euklidiske på alle tre datasæt, selvom den i få tilfælde giver værre resultater for den enkelte testperson. Ved brug af Riemann-metrikken på BCIC-data blev der i forsøgene opnået en bedre grad af korrekte klassificeringer med en forbedring på henholdsvis 41% for den individuelle data og 21% for den aggregerede data i forhold til Euklid-metrikken. Det kan dermed konkluderes, at Riemann-metrikken er væsentligt bedre til at klassificere denne type problemer, og vil være en brugbar forbedring i forhold til den Euklidiske metrik i løsningen til rapportens problem.

BCIC-dataen er ikke lavet med consumer-grade EEG-teknologi, og der kan derfor ikke alene ud fra resultaterne af klassificering på den data udfærdiges en løsning på rapportens problem. Det bliver i stedet sat op imod datamålinger fra Emotiv-headsettet for dermed, at kunne konkludere om consumer-grade EEG-teknologi, vil være brugbart som løsning til rapportens problem.

Dataen optaget med Emotiv headsettet har med 66% en 12 pp værre klassificerings-success end BCIC-dataen. For faktisk bevægelse stiger klassificerings-succesen til 74,1%. Både for tænkt og faktisk bevægelse er der også stor forskel på klassifications-succesen fra en bruger til en anden. Til brug i systemer, hvor pålidelighed er meget vigtigt, vil 74,1% klassificeringssuccess sandsynligvis være alt for lidt. Men givet problemformuleringen vil det muligvis kunne bruges.

7.4 Perspektivering

Projektet kan udvides på mange måder. En oplagt måde er at se på forbedringer til Riemann-klassificeringsalgoritmen. I Barachant et al. [2012] giver de to varianter af Riemann-algoritmen. Den ene variant er den, vi har implementeret i dette projekt, og den anden variant benytter sig af projektioner ind i tangentrummet og reducering af antallet af dimensioner. I artiklen giver den anden variant bedre resultater, og man kunne derfor prøve at implementere denne for consumer-grade hardware og se om dette giver forbedrede resultater. I artiklen bruger de desuden en klassificeringsalgoritme kaldet *Linear Discriminant Analysis* (LDA) til den anden variant. Her kunne man også prøve at bruge en mere sofistikeret algoritme end LDA.

I vores implementering kunne man prøve forskellige forbedringer, såsom at bruge medianen i stedet for middelværdien for klasserne, hvilket i teorien burde være mere resistent overfor outliers. Man kunne også i stedet for at bruge kovariansmatricen Σ prøve at bruge præcisionsmatricen Σ^{-1}, hvor man sætter de indgange, der er meget tæt på 0 til præcis 0. På denne måde kan man reducere dimensionaliteten af dataen og derved muligvis forbedre klassificeringsnøjagtigheden.

En anden tilgangsvinkel til udvidelse af projektet er at bygge en form for brugergrænseflade. På nuværende tidspunkt bliver klassificeringen af ny data ikke brugt til noget, men man kunne forestille sig at klassificeringen kan bruges til at vise en grafisk repræsentation af den tænkte bevægelse, eller bruges til at styre andre programmer som eksempelvis computerspil.

Litteratur

Morteza Alamgir, Moritz Grosse-Wentrup, and Yasemin Altun. Multitask learning for brain-computer interfaces. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 9, 2010.

Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2nd edition, 2010.

Alexandre Barachant, Stéphane Bonnet, Marco Congedo, and Christian Jutten. Common spatial pattern revisited by Riemannian geometry. In *Multimedia Signal Processing (MMSP), 2010 IEEE International Workshop on*, pages 472–476. IEEE, 2010.

Alexandre Barachant, Stéphane Bonnet, Marco Congedo, and Christian Jutten. Multiclass brain–computer interface classification by Riemannian geometry. *Biomedical Engineering, IEEE Transactions on*, 59(4):920–928, 2012.

Benjamin Blankertz, Ryota Tomioka, Steven Lemm, Motoaki Kawanabe, and Klaus-Robert Müller. Optimizing spatial filters for robust EEG single-trial analysis. *IEEE Signal Processing Magazine*, pages 41–56, January 2008.

Chris Brown. Informal introduction to signal processing and the frequency domain. http://ftp.cs.rochester.edu/u/nelson/courses/csc_160/2011_spring/readings/signal_proc.pdf, July 2011.

Clemens Brunner, Robert Leeb, Gernot R. Müller-Putz, Alois Schlögl, and Gert Pfurtscheller. BCI competition 2008 - Graz data set A. 2008.

Stephen Butterworth. On the theory of filter amplifiers. *Experimental wireless & the wireless engineer*, October 1930. http://www.gonascent.com/papers/butter.pdf.

György Buzsáki. *Rhythms of the Brain*. Oxford University Press, 2006.

Ward Cheney and David Kincaid. *Numerical Mathematics and Computing*. Thomson, 6th edition, 2008.

Database Center for Life Science, 2012. https://commons.wikimedia.org/wiki/File:Precentral_gyrus_animation.gif, 2012.

Emotiv. http://emotiv.com/upload/manual/EEGSpecifications.pdf.

Simone Fiori. Learning the fréchet mean over the manifold of symmetric positive-definite matrices. *Cognitive Computation*, 1(4):279–291, 2009.

P. Thomas Fletcher and Sarang Joshi. Principal geodesic analysis on symmetric spaces: Statistics of diffusion tensors. In *Computer Vision and Mathematical Methods in Medical and Biomedical Image Analysis*, pages 87–98. Springer, 2004.

Alan Gittis. The measurement of brain waves. http://www.psych.westminster.edu/psybio/BN/Labs/Brainwaves.htm.

Scott T. Grafton, Michael A. Arbib, Luciano Fadiga, and Giacomo Rizzolatti. Localization of grasp representations in humans by positron emission tomography. *Experimental brain research*, 112(1):103–111, 1996.

Gyldendals åbne encyklopædi. hjerne. http://www.denstoredanske.dk/Krop,_psyke_og_sundhed/Sundhedsvidenskab/Sammenlignende_anatomi_og_fysiologi/hjerne.

Hjernesagen. Faktaoplysninger om apopleksi. http://www.hjernesagen.dk/apopleksi/fakta-til-er-det-mon-en-prop-i-hjernen, 2009.

Valer Jurcak, Daisuke Tsuzuki, and Ipperita Dan. 10/20, 10/10, and 10/5 systems revisited: Their validity as relative head-surface-based positioning systems. *NeuroImage*, 34, 2007.

Fabien Lotte and Cuntai Guan. Regularizing common spatial patterns to improve BCI designs: Unified theory and new algorithms. *Biomedical Engineering, IEEE Transactions on*, 58(2):355–362, 2011.

Paul Lutus. The discrete fourier transform – a practicum on fourier analysis and signal processing. http://www.arachnoid.com/signal_processing/dft.html, 2008.

Scott Makeig, Christian Kothe, Tim Mullen, Nima Bigdely-Shamlo, Zhilin Zhang, and Kenneth Kreutz-Delgado. Evolving signal processing for brain-computer interfaces. *Proceedings of the IEEE*, 100:1567–1584, May 2012.

Microsoft, 2013. http://www.xbox.com/da-DK/Kinect/GetStarted, 2013.

Maher Moakher. A differential geometric approach to the geometric mean of symmetric positive-definite matrices. *SIAM Journal on Matrix Analysis and Applications*, 26(3): 735–747, 2005.

Paul L. Nunez and Ramesh Srinivasan. Electroencephalogram. http://www.scholarpedia.org/article/Electroencephalogram, 2007.

Ariel Ortiz. Pipes and filters architectural pattern. http://webcem01.cem.itesm.mx:8005/apps/s200911/tc3003/notes_pipes_and_filters/, 2013.

Gitte Gundtoft Pasgaard. Robot styrkedragt. http://www.hjernesagen.dk/genoptrening/genoptrening/robot-styrkedragt, 2011.

Gert Pfurtscheller, Gernot R. Müller-Putz, Reinhold Scherer, and Christa Neuper. Rehabilitation with brain-computer interface systems. *Computer*, 41(10):58–65, 2008.

Dorte Malig Rasmussen. Velfærdsteknologivurdering af virtuel genoptræning. Technical report, Teknologisk Institut, 2012.

Fátima Rodrigues-de-Paula and Lidiane Oliveira Lima. Physical therapy - exercise and parkinsons disease. 2010. http://cirrie.buffalo.edu/encyclopedia/en/article/336/.

Paul Shirkey. Iir filtering with butterworth filters. http://www.centerspace.net/blog/tag/c-butterworth-filter/, 2013.

Social-, Børne-, og Integrationsministeriet. Genoptræning og vedligeholdelsestræning. http://www.sm.dk/temaer/sociale-omraader/handicap/hjaelp-og-stoette/genoptr%C3%A6ning-og-vedligeholdelsestr%C3%A6ning/Sider/Start.aspx.

Surjo R. Soekadar, Niels Birbaumer, and Leonardo G. Cohen. Brain-computer-interfaces in the rehabilitation of stroke and neurotrauma. In Kenji Kansaku and Leonardo G. Cohen, editors, *Systems Neuroscience and Rehabilitation*. 2011.

Danmarks Statistik. Befolkningen 1. januar 2010. *Nyt fra Danmarks Statistik*, 2010. http://www.dst.dk/pukora/epub/Nyt/2010/NR059.pdf.

Danmarks Statistik. Genoptræning og vedligeholdende træning, udgifter og effekt, efter indikator og område. http://www.statistikbanken.dk/statbank5a/SelectVarVal/Define.asp?Maintable=SUIK21&PLanguage=0, 2013.

Sterman Kaiser Imaging Laboratory. http://www.skiltopo.com/BA/myBrodmannsAreas2-802.jpg, a.

Sterman Kaiser Imaging Laboratory. http://www.skiltopo.com/1/index.htm, b.

Teknologi-Rådet. Sådan virker hjernen. http://www.tekno.dk/undervisning/materiale/artikel1.pdf.

WelfareTech. Honda tester sundhedsrobotter i danmark. http://www.welfaretech.dk/nyheder/honda-tester-sundhedsrobot-i-danmark/, 2013.

Sherif M. Yacoub. Composite filter pattern. *Publishing Systems and Solutions Labatory*, May 2001.

A

Appendiks

A.1 Sandsynlighedsteori

Inputtet til en maskinindlæringsalgoritme kan anses som en stokastisk variabel X, da værdien af inputtet kan ses som en hændelse i et givet udfaldsrum. Ud fra denne betragtning kan vi bruge gængse sandsynlighedsteoretiske begreber til at beskrive de statistiske egenskaber af vores data.

Definition A.1 (Middelværdi)
Middelværdien $\mathrm{E}[X]$ for en stokastisk variabel X er givet ved

$$\mathrm{E}[X] = \sum_{k=1}^{\infty} x_k p(x_k),$$

hvis X er en diskret variabel, hvor X har udfaldsrum $\{x_1, x_2, \dots\}$ og sandsynlighedsfunktion p. Hvis X er en kontinuert variabel, så har vi

$$\mathrm{E}[X] = \int_{-\infty}^{\infty} x f(x) \mathrm{d}x,$$

hvor f er tæthedsfunktionen for X.

Definition A.2 (Varians)
Variansen for en stokastisk variabel X er givet ved

$$\mathrm{Var}[X] = \mathrm{E}\Big[(X - \mathrm{E}[X])^2\Big].$$

Definition A.3 (Kovarians)
Kovariansen mellem to stokastiske variable X og Y er givet ved

$$\mathrm{Cov}[X, Y] = \mathrm{E}\Big[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])\Big].$$

Hvis $X = Y$ er kovariansen blot variansen af variablen, $\mathrm{Cov}[X, X] = \mathrm{Var}[X]$.

A.2 Matricer og matrixrum

Punkterne på de Riemannske mangfoldigheder vi arbejder med er symmetriske, positiv definitte (SPD) matricer (defineret nedenunder). Kovariansmatricer som bruges i BCI-klassifikation er garanteret at være SPD, og de udgør også en Riemannsk mangfoldighed, på hvilken vi kan definere afstanden mellem matricer.

Definition A.4 (Symmetriske matricer)
En kvadratisk matrix S er *symmetrisk* hvis $S^T = S$. Rummet af alle symmetriske $n \times n$ matricer er angivet som $S(n)$,

$$S(n) = \big\{S \mid S \text{ er } n \times n \text{ og } S^T = S\big\} \tag{A.1}$$

De interessante objekter, og dem som danner Riemann-rummet, er symmetriske positiv definitte matricer.

Definition A.5 (SPD matricer)
En symmetrisk $n \times n$ matrix P er positiv definit hvis $\boldsymbol{v}^T P \boldsymbol{v} > 0$ for alle $\boldsymbol{v} \neq \mathbf{0} \in \mathbb{R}^n$. Mængden af alle symmetriske positiv definitte (SPD) $n \times n$ matricer er angivet som $P(n)$,

$$P(n) = \big\{P \in S(n) \mid \boldsymbol{v}^T P \boldsymbol{v} > 0, \forall \boldsymbol{v} \neq \mathbf{0}, \boldsymbol{v} \in \mathbb{R}^n\big\}. \tag{A.2}$$

For alle SPD-matricer $P_1 \in P(n)$, har vi egenskaberne at

- alle egenværdier af P_1 er positive og reelle: $\lambda_i > 0$ for alle egenværdier λ_i der tilfredsstiller $P_1 \boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$ for tilhørende egenvektorer $\boldsymbol{v}_i$,
- $\det(P_1) > 0$, så P_1 er invertibel,
- den inverse $P_1^{-1} \in P(n)$,
- $\mathrm{Tr}(P_1) > 0$,
- og hvis $P_2 \in P(n)$, så er $P_1 P_2 \in P(n)$.

For at forsimple beregningerne bruger vi at symmetriske matricer altid er diagonaliserbare. Specielt bruger vi deres egenværdidekomponering, og skriver dem ud fra deres egenværdier og egenvektorer.

Definition A.6 (Egenværdidekomponering)
Lad A være en $n \times n$ matrix med n uafhængige linære egenvektorer $\boldsymbol{q}_i$, $i = 1, \cdots, n$ og tilhørende egenværdier λ_i, der opfylder $A\boldsymbol{q}_i = \lambda_i \boldsymbol{q}_i$. Så kan A entydigt faktoriseres på formen

$$A = Q\Lambda Q^{-1}, \tag{A.3}$$

hvor $\Lambda = \text{diag}(\lambda_1, \cdots, \lambda_n)$ er en diagonalmatrix af egenværdier og Q er en matrix af egenvektorer, med $\boldsymbol{q}_i$ i den i'te række.

For en symmetrisk matrix S, kan egenvektorerne vælges som værende ortogonale. Derfor er Q en ortogonal matrix, $Q^{-1} = Q^T$, og S kan faktoriseres som

$$S = Q\Lambda Q^T. \tag{A.4}$$

A.3 Eksponential- og logaritmefunktionen for matricer

Den matematiske ramme kræver at vi kan bruge eksponental- og logaritmefunktionen på matricer. Dette er *ikke* det samme som blot at benytte funktionerne på de individuelle indgange i en matrix.

For reelle tal $x \in \mathbb{R}$ kan vi definere eksponentialfunktionen gennem dens Taylor-udvikling

$$\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!}. \tag{A.5}$$

Generaliseres dette, kan vi definere funktionen $\exp(A)$ af en *matrix* A som

$$\exp(A) = \sum_{k=0}^{\infty} \frac{A^k}{k!}, \tag{A.6}$$

hvor A^k betegner multiplikation af A med sig selv k gange. Som i det reelle tilfælde konvergerer ligning (A.6) altid. Derefter definerer vi $\ln(A)$ som den inverse funktion til det ovenstående, således at

$$\exp(\ln(A)) = \ln(\exp(A)) = A. \tag{A.7}$$

Ligning (A.6) er meget besværlig at arbejde med, men for symmetriske matricer antager den en meget enklere form. Dekomponeringen af en symmetrisk matrix S som i ligning (A.4) giver

$$\exp(S) = \exp(Q\Lambda Q^T). \tag{A.8}$$

En egenskab ved eksponentialfunktionen for matricer er at $\exp(ABA^{-1}) = A\exp(B)A^{-1}$ hvis A er invertibel, så

$$\exp(S) = Q\exp(\Lambda)Q^T = Q\exp(\text{diag}(\lambda_1, \cdots, \lambda_n))Q^T = Q\text{diag}(e^{\lambda_1}, \cdots, e^{\lambda_n})Q^T. \tag{A.9}$$

Derfor kan eksponentialfunktionen for matricer beregnes med den almindelige eksponentialfunktion for egenværdierne i den diagonale matrix Λ hvis matricen er symmetrisk. Ligeledes tager matrix-logaritmefunktionen for symmetriske matricer formen

$$\ln(S) = Q\text{diag}(\ln\lambda_1, \cdots, \ln\lambda_n)Q^T, \tag{A.10}$$

hvor ln er den reelle logaritmefunktion.

Kvadratroden af en matrix indgår også i den teoretiske del. Svarende til for reelle tal, hvor vi siger at $b = a^{1/2}$ hvis $bb = a$, definerer vi kvadratrødderne $B = A^{1/2}$ af en matrix A som dem, der opfylder $BB = A$. Generelt er der mange sådanne kvadratrødder, men en vælges som den principielle rod som for relle kvadratrødder. Kvadratrødderne kan findes ved at løse den definerende ligning $BB = A$.

A.4 Indre produkter og normer

At beskrive krumme flader med Riemannsk geometri kræver en generalisering af koncepterne om indre produkter og normer i Euklidiske rum. I $\mathbb{R}^n$ er det indre produkt $\langle \boldsymbol{x}, \boldsymbol{y}\rangle = \boldsymbol{x}\cdot\boldsymbol{y}$ og normen er $||\boldsymbol{x}|| = x = \sqrt{\boldsymbol{x}\cdot\boldsymbol{x}}$. Disse og deres generaliseringer skal opfylde nogle generelle egenskaber, og danner grundlaget for geometri i Euklidiske rum og tangentrummet for Riemannske flader, der begge er vektorrum.

Definition A.7 (Indre produkt)
Et *indre produkt* på et vektorrum V er en funktion $\langle\cdot,\cdot\rangle : V \times V \to \mathbb{R}$ som for alle vektorer $u, v, w \in V$ og skalarer $a \in \mathbb{R}$ opfylder følgende egenskaber:

- Symmetri: $\langle u, v\rangle = \langle v, u\rangle$
- Linearitet under multiplikation: $\langle au, v\rangle = \langle u, av\rangle = a\langle u, v\rangle$
- Linearitet under addition: $\langle u+v, w\rangle = \langle u, w\rangle + \langle v, w\rangle$ og $\langle u, v+w\rangle = \langle u, v\rangle + \langle u, w\rangle$
- Positiv definit: $\langle u, u\rangle \geq 0$ og $\langle u, u\rangle = 0$ hvis og kun hvis $u = 0$.

Et vektorrum med et indre produkt kaldes et indre produktrum, og det har også en induceret norm defineret ud fra det indre produkt.

Definition A.8 (Norm)
En *norm* på et indre produktrum V er en funktion $\|\cdot\| : V \to \mathbb{R}$ defineret ved $\|u\| = \sqrt{\langle u, u\rangle}$ for alle $u \in V$. For alle vektorer $u, v \in V$ og skalarer $a \in \mathbb{R}$, opfylder den følgende:

- Homogenitet: $\|au\| = |a|\|u\|$
- Trekantsuligheden: $\|u + v\| \leq \|u\| + \|v\|$
- Positiv definit: $\|u\| \geq 0$ og $\|u\| = 0$ hvis og kun hvis $u = 0$.

Alle egenskaberne ovenfor følger direkte af egenskaberne for det indre produkt.

De generelle betingelser ovenfor tillader os at definere indre produkter og normer for matricer, eftersom rummet for matricer over $\mathbb{R}^{n\times m}$ selv udgør et vektorrum. Der er ingen entydig måde at gøre dette på, men for $n \times n$-matricer i det Euklidiske rum $\mathbb{R}^{n\times n}$, er *Frobeniusproduktet* et naturligt valg.

Definition A.9 (Frobeniusprodukt)
For to reelle matricer $A, B \in \mathbb{R}^{n\times n}$, er Frobeniusproduktet $\langle\cdot,\cdot\rangle_F : \mathbb{R}^{n\times n} \to \mathbb{R}$ defineret ved

$$\langle A, B\rangle_F = \mathrm{Tr}(A^T B). \tag{A.11}$$

Dette produkt opfylder alle de krævede egenskaber i definition (A.7).

Eftersom $(AB)^T = B^T A^T$, $\mathrm{Tr}(A^T) = \mathrm{Tr}(A)$ og sporet er invariant under permutationer $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$, er Frobenius-produktet også givet ved hver af

$$\langle A, B\rangle_F = \mathrm{Tr}(A^T B) = \mathrm{Tr}(AB^T) = \mathrm{Tr}(B^T A) = \mathrm{Tr}(BA^T), \tag{A.12}$$

og derfor er eksempelvis $\langle A, B\rangle_F = \langle B, A\rangle_F$ som krævet.

Dette indre produkt inducerer en norm på $\mathbb{R}^{n\times n}$, kaldet Frobenius-normen $\|\cdot\|_F$.

Definition A.10 (Frobenius-norm)
For enhver matrix $A \in \mathbb{R}^{n\times n}$, er Frobenius-normen $\|\cdot\|_F : \mathbb{R}^{n\times n} \to \mathbb{R}$ givet ud fra Frobenius-indre produktet som

$$\|A\|_F = \sqrt{\langle A, A\rangle_F} = \sqrt{\mathrm{Tr}(A^T A)}, \tag{A.13}$$

og det opfylder alle egenskaberne i definition A.8.

Normen på et rum tillader os at definere et begreb for *afstand* mellem punkter, og dermed gøre rummet til et metrisk rum. De intuitive krav til hvad der forventes af et afstandbegreb kan formaliseres eksakt.

Definition A.11 (Metrisk rum)
Et *metrisk rum* er et par (M, d) hvor M er en mængde af punkter og $d : M \times M \to \mathbb{R}$ er en afstandsfunktion, også kaldet metrik, som giver afstanden mellem punkterne i M. For alle $x, y, z \in M$, skal d opfylde følgende egenskaber:

- Ikke-negativitet: $d(x, y) \geq 0$
- Entydighed af punkterne: $d(x, y) = 0$ hvis og kun hvis $x = y$
- Symmetri: $d(x, y) = d(y, x)$
- Trekantsuligheden: $d(x, z) \leq d(x, y) + d(y, z)$

Normer i vektorrum gør det let at definere metriske rum for disse rum. For vektorer $\boldsymbol{x}, \boldsymbol{y}$ i $\mathbb{R}^n$ har vi den sædvanlige Euklidiske afstandsfunktion

$$d_E(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|, \tag{A.14}$$

og Frobenius-normen tillader os at definere afstanden mellem matricerne A, B i $\mathbb{R}^{n \times n}$ ligeledes som

$$d_F(A, B) = \|A - B\|_F. \tag{A.15}$$

Den *Riemannske* afstand mellem punkter på en Riemann-flade gør også denne til et metrisk rum, men for at definere denne kræves mere teoretisk baggrund (se næste afsnit).

A.5 Riemannske mangfoldigheder

En note om sprogkonventionen; i denne rapport og i litteraturen kaldes Riemannske mangfoldigheder også Riemann-flader og Riemann-rum. Disse begreber betegner alle det samme matematiske objekt. Flader bruges oftest om todimensionelle objekter, mens mangfoldigheder er generaliseringen af dette begreb til arbitrær dimensionalitet.

En Riemannsk mangfoldighed er den matematisk præcise formulering for det intuitive koncept om en krum flade, for eksempel overfladen på en kugle (en todimensionel flade). Euklidiske rum er også eksempler på Riemann-mangfoldigheder, som dog er trivielle da de har ingen krumning.

En funktion er *glat* hvis den er differentiabel uendeligt mange gange, dvs. at afledte af funktionen eksisterer til alle ordner. Beskrivelsen af geometrien af en Riemannsk mangfoldighed kræver at man kan benytte differential- og integralregning, og derfor kræves det at mangfoldigheden er glat, i samme forstand som for funktioner. Dette svarer intuitivt til at man kan gå kontinuerligt fra et punkt til et andet på fladen, uden "spring".

Da mangfoldigheden kan være krum, kan man generelt *ikke* definere vektorer der peger fra et punkt til et andet. Til gengæld er der i hvert punkt på fladen defineret et tangentrum. Intuitivt peger vektorerne i tangentrummet i retninger man kan bevæge sig i væk fra punktet, og denne intuition er formaliseret i eksponential- og logaritmeafbildningen.

Definition A.12 (Tangentrum)
For ethvert punkt p på en glat mangfoldighed $\mathcal{M}$ betegnes tangentrummet i p $T_p\mathcal{M}$. Tangentrummet er et vektorrum af samme dimension som selve mangfoldigheden.

Bemærk at det generelt *ikke* giver mening at sammenligne vektorer i tangentrum for forskellige punkter på mangfoldigheden.

Riemannske mangfoldigheder er karakteriseret ved indre produkter af vektorer i tangentrummet. Hvis vektoren i tangentrummet er en funktion af position på mangfoldigheden, kaldes den et vektorfelt.

Definition A.13 (Vektorfelt)
Et *vektorfelt* X på en mangfoldighed $\mathcal{M}$ er en funktion $X : \mathcal{M} \to T_pM$ som tildeler en vektor i tangentrummet $T_p\mathcal{M}$ i p for alle punkter $p \in \mathcal{M}$.

Et vektorfelt kan være glat på samme måde som funktioner. Hvis vektorerne opløses i en basis, vil vektorfeltet være glat hvis alle komponentfunktionerne er glatte. Vi kan nu give definitionen af Riemannske mangfoldigheder.

Definition A.14 (Riemannsk mangfoldighed)
En *Riemannsk mangfoldighed* er et par $(\mathcal{M}, g)$ hvor $\mathcal{M}$ er en glat mangfoldighed og g_p er et indre produkt defineret på tangentrummet $T_p\mathcal{M}$ i ethvert punkt $p \in \mathcal{M}$. Det indre produkt g_p skal variere jævnt med p, hvilket vil sige at funktionen $\phi : \mathcal{M} \to \mathbb{R}$ defineret for alle glatte vektorfelter $X(p), Y(p)$ på $\mathcal{M}$ ved

$$\phi(p) = g_p(X(p), Y(p)) \tag{A.16}$$

er glat.

Det indre produkt $g_p(x, y)$ skrives også $\langle x, y \rangle_p$, og er generelt en funktion af punktet p. Det indre produkt kaldes også en metrik, som ikke skal forveksles med afstandsfunktionen for metriske rum.

Tangentrummene i alle punkter på mangfoldigheden er nu også indre produktrum. Dermed kan vi definere en norm for vektorer x i tangentrummet $T_p\mathcal{M}$ ud fra

$$||x||_p = \sqrt{\langle x, x \rangle_p}. \tag{A.17}$$

Det næste skridt, at gøre Riemann-mangfoldigheden til et metrisk rum, kompliceres af at tangentrummene er forskellige for forskellige punkter. For krumme flader er afstanden mellem to punkter *ikke* nødvendigvis afstanden af en lige linje mellem dem.

A.6 Afstande mellem punkter og geodætiske kurver

Afstanden mellem punkter er længden af den korteste sti mellem dem. I Euklidiske rum er den korteste sti en lige linje, men generelt er det ikke sandt. En sti er en parametriseret kurve der starter i et punkt og ender i et andet.

Definition A.15 (Parametriserede kurver)
En parametriseret kurve mellem punkter p_1, p_2 på en Riemannsk mangfoldighed $\mathcal{M}$ er en glat funktion $\gamma : [0, 1] \to \mathcal{M}$, hvor $\gamma(0) = p_1$ og $\gamma(1) = p_2$.

Længden af en kurve defineres som for parametriserede kurver i Euklidiske rum, ved et integral over infinitesimale længder langs kurven som findes fra normen.

Definition A.16 (Kurvelængde)
Kurvelængden $L[\gamma]$ af en kurve $\gamma(\lambda)$ er

$$L[\gamma] = \int_0^1 \|\dot{\gamma}(\lambda)\|_{\gamma(\lambda)} \, \mathrm{d}\lambda, \tag{A.18}$$

hvor $\dot{\gamma}(\lambda) = \frac{\mathrm{d}\gamma}{\mathrm{d}\lambda}$.

Bemærk at $\dot{\gamma}(\lambda)$ er en vektor i tangentrummet, og at normen generelt varierer fra punkt til punkt. For ethvert par af punkter er der uendeligt mange kurver mellem dem. For at definere afstande bruges de kurver med minimal kurvelængde, der kaldes geodæter. Den geodætiske afstand mellem to punkter er således længden af den korteste sti mellem dem. Det er ikke garanteret at geodæterne for en Riemannsk mangfoldighed er entydige, men for SPD-rum er det sandt.

Fra afstanden af alle geodæter mellem par af punkter på Riemann-mangfoldigheden kan den gøres til et metrisk rum, hvor afstandsfunktionen er den såkaldte geodætiske metrik.

Definition A.17 (Geodætisk metrik)
Givet ethvert par af punkter x, y i en Riemannsk mangfoldighed $\mathcal{M}$, er afstandsfunktionen

$$d_R(x, y) = \min_{\gamma \in \Gamma} L[\gamma] = \min_{\gamma \in \Gamma} \int_0^1 \|\dot{\gamma}(\lambda)\|_{\gamma(\lambda)} \, \mathrm{d}\lambda, \tag{A.19}$$

hvor Γ er mængden af alle stier γ hvor $\gamma(0) = x$ og $\gamma(1) = y$.

Det er generelt ikke nemt at finde en geodætisk metrik ud fra det ovenstående.

A.6.1 Eksponential- og logaritmeafbildningen

Eksponential- og logaritmeafbildningen er funktioner, der afbilder mellem selve Riemann-mangfoldigheden $\mathcal{M}$ og tangentrummet $T_p\mathcal{M}$. De er de værktøjer, der er nødvendige for at bevæge et punkt rundt på mangfoldigheden.

Betragt et startpunkt p med tilhørende tangentrum $T_p\mathcal{M}$, og et andet punkt p_1. Antag at der findes en geodæt mellem p og p_1. Så er eksponentialafbildningen i p en funktion $\mathrm{Exp}_p : T_p\mathcal{M} \to \mathcal{M}$ og logaritmeafbildningen en funktion $\mathrm{Log}_p : \mathcal{M} \to T_p\mathcal{M}$. Disse er hinandens inverse.

Intuitionen bag afbildningerne er:

- Begyndende i p og givet en retningsvektor $x \in T_p\mathcal{M}$, hvilket punkt p_2 ender man i hvis man begynder at bevæge sig i retning x? (Eksponentialafbildningen)
- Begyndende i p og givet et slutpunkt p_1, hvilken retning $x \in T_p\mathcal{M}$ skal man begynde at bevæge sig langs for at ende i p_1?

Se også figur 4.1 og diskussionen i afsnit 4.1.1.

A.7 Den Euklidiske og Riemannske middelværdi

Den almindelige middelværdi for N tal $x_1, \cdots, x_N \in \mathbb{R}$

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{A.20}$$

kan generaliseres direkte til at gælde for vektorer og matricer, så vi for N matricer $A_1, \cdots, A_N \in \mathbb{R}^{n \times m}$ har den aritmetiske matrixmiddelværdi i en Euklidisk metrik

$$\mu = \frac{1}{N} \sum_{i=1}^{N} A_i. \tag{A.21}$$

For at generalisere dette begreb benyttes Fréchet-middelværdien, der er defineret for alle metriske rum, og dermed kan bruges til at definere middelværdier i Riemannske rum. For Euklidiske rum er definitionen i overensstemmelse med den ovenstående.

Definition A.18 (Fréchet middelværdi [Fiori, 2009])
For et metrisk rum (M, d) og en mængde af N punkter $x_1, \cdots, x_N$ i M er middelværdien $\mu \in M$ af punkterne det punkt der minimerer den vægtede kvadrerede afstand til alle punkter,

$$\mu = \arg\min_{p \in M} \sum_{i=1}^{N} w_i d^2(x_i, p), \tag{A.22}$$

hvor $w_1, \cdots, w_N$ er ikke-negative vægte.

Hvis mængden af punkter i M er et udfaldsrum for en stokastisk variabel, er vægtene w_i den tilhørende sandsynlighed $p(x_i)$, og vi får den sandsynlighedsteoretiske middelværdi.

Hvis det metriske rum består af matricer, alle vægtene $w_i = 1$ og afstandsfunktionen er givet ved den Euklidiske Frobenius-norm $d_E(A, B) = \|A - B\|_F$, kan det vises at punktet μ der minimerer funktionen i ovenstående definition er præcis den givet i (A.21). Dette betyder at vi alternativt kan definere den Euklidiske middelværdi af en mængde af matricer i M ved

$$\mu_E(A_1, \cdots, A_N) = \arg\min_{P \in M} \sum_{i=1}^{N} d_E^2(A_i, P) = \arg\min_{P \in M} \sum_{i=1}^{N} \|A_i - P\|_F^2 \tag{A.23}$$

$$= \frac{1}{N} \sum_{i=1}^{N} A_i. \tag{A.24}$$

Med udgangspunkt i dette udtryk generaliserer vi og *definerer* den Riemannske middelværdi for en mængde af punkter $p_1, \cdots, p_N$ i et Riemann-rum $\mathcal{M}$ som

$$\mu_R(p_1, \cdots, p_N) = \arg\min_{p \in \mathcal{M}} \sum_{i=1}^{N} d_R^2(p_i, p), \tag{A.25}$$

hvor den metriske funktion d_R er defineret ved den geodætiske metrik (A.17). I det ovenstående har vi antaget at der findes et entydigt punkt der opfylder kriteriet. Dette er *ikke* nødvendigvis sandt generelt, men for både Euklidiske rum og rum af SPD-matricer er det sandt.

A.8 CSP-algoritmen

Common Spatial Pattern (CSP) er en superviseret algoritme, som virker på båndpasfiltreret flerkanalsdata med to klasser, fx EEG-målinger hvor testpersonen tænkte på henholdsvis højre og venstre. Algoritmen antager at dataen i forvejen er centreret, således at middelværdien $\mathrm{E}[X] = 0$. Vores beskrivelse af CSP tager udgangspunkt i Blankertz et al. [2008].

Algoritmen laver en featurisering $\boldsymbol{x}_{\mathrm{CSP}}(t) \in \mathbb{R}^C$ af en måling $\boldsymbol{x}$, hvor t er et tidspunkt og C er antallet af kanaler. Denne featurisering er givet ved

$$\boldsymbol{x}_{\mathrm{CSP}} = W^T \boldsymbol{x}(t),$$

hvor W er en matrix og hver kolonne $\boldsymbol{w}_j \in \mathbb{R}^C$ for $j = 1, 2, \ldots, C$ i W er et spatialt filter. Målet med algoritmen er at finde de spatiale filtre $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_C$, der maksimerer variansen af signalet for den ene klasse og samtidig minimerer variansen for den anden klasse, for på denne måde lettere at kunne diskriminere mellem de to klasser.

Vi vil altså gerne finde de filtre $\boldsymbol{w}$ som opfylder at

$$\boldsymbol{w} = \arg\max_{\boldsymbol{w} \in \mathbb{R}^C} \frac{\boldsymbol{w}^T X_1^T X_1 \boldsymbol{w}}{\boldsymbol{w}^T X_2^T X_2 \boldsymbol{w}}.$$

For at gøre dette, estimeres de to kovariansmatricer $\Sigma^+ \in \mathbb{R}^{C \times C}$ og $\Sigma^- \in \mathbb{R}^{C \times C}$ for de to klasser. Da vi har at $E[X] = 0$, forsvinder disse led i kovariansen, og estimatet er derfor givet ved

$$\Sigma^c = \frac{1}{|\mathcal{I}_c|} \sum_{i \in \mathcal{I}_c} X_i X_i^T \ , \quad c \in \{+, -\},$$

hvor $\mathcal{I}_c$ er mængden af indekser svarende til den data, der er i klassen $c = \{+, -\}$, og $X \in \mathbb{R}^{C \times T}$ er en måling med C kanaler over et tidsinterval med T tidspunkter. Vi får nu at

$$\boldsymbol{w} = \arg\max_{\boldsymbol{w} \in \mathbb{R}^C} \frac{\boldsymbol{w}^T \Sigma^+ \boldsymbol{w}}{\boldsymbol{w}^T \Sigma^- \boldsymbol{w}}.$$

Dette problem har mere end én løsning [Lotte and Guan, 2011], så for at finde en entydig løsning, begrænser vi problemet så det skal opfylde at $\boldsymbol{w}^T \Sigma^- \boldsymbol{w} = 1$. Ved at bruge Lagranges multiplikatormetode, svarer problemet til at maksimere funktionen

$$L(\lambda, \boldsymbol{w}) = \boldsymbol{w}^T \Sigma^+ \boldsymbol{w} - \lambda(\boldsymbol{w}^T \Sigma^- \boldsymbol{w} - 1).$$

Dette kan gøres ved at tage den afledte med hensyn til $\boldsymbol{w}$ og sætte lig med 0 [Lotte and Guan, 2011].

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{w}} &= 2\boldsymbol{w}^T\Sigma^+ - 2\lambda\boldsymbol{w}^T\Sigma^- = 0 \\ &\Leftrightarrow \Sigma^+\boldsymbol{w} = \lambda\Sigma^-\boldsymbol{w} \\ &\Leftrightarrow (\Sigma^-)^{-1}\Sigma^+\boldsymbol{w} = \lambda\boldsymbol{w}. \end{aligned} \tag{A.26}$$

Ud fra ligning (A.26) kan det ses, at løsningen til maksimeringsproblemet er givet ved et egenværdiproblem. Efter at have løst dette egenværdiproblem, vil egenvektorerne $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_C$ udgøre de filtre, vi ønskede at finde. I praksis vælges dog kun de egenvektorer, der hører til de k største og k mindste egenværdier. Hvis k vælges for lille, vil diskriminantfunktionen ikke kunne diskriminere ordentligt mellem de to klasse. Omvendt, hvis k vælges for stort, kan overfitting blive et stort problem. Algoritmen bruges derfor ofte med $k = 3$ [Blankertz et al., 2008].

A.9 Gyldne Snit Søgning

Gyldne snit-søgning (GSS) er en metode til at finde maksimum og minimum på en *unimodal* kontinuert reel funktion. En unimodal funktion har kun ét minimum eller maksimum på en kompakt mængde $[a, b]$ hvor $a, b \in \mathbb{R}$. Minimum og maksimum-tilfældene for algoritmen er analoge, derfor beskæftiger vi os kun med tilfældet for minimum i denne diskussion. Desuden lader vi x^* betegne minimummet for en vilkårlig reel funktion $f(x)$.

GSS er blevet navngivet efter sin sammenhæng med forholdstallet φ, også kendt som det gyldne snit [Cheney and Kincaid, 2008]

$$\varphi = \frac{1}{2}\left(1 + \sqrt{5}\right) \approx 1,6180339887.$$

Det gælder om φ at det opfylder ligningen $\varphi^2 - \varphi - 1 = 0$, som har rødderne $\frac{1}{2}\left(1 + \sqrt{5}\right)$ og $\frac{1}{2}\left(1 - \sqrt{5}\right)$.

GSS inddeler successivt et interval $[a, b]$ i mindre delintervaler og dermed skaber en monoton aftagende følge af intervaller med respekt til relationen $\supset$:

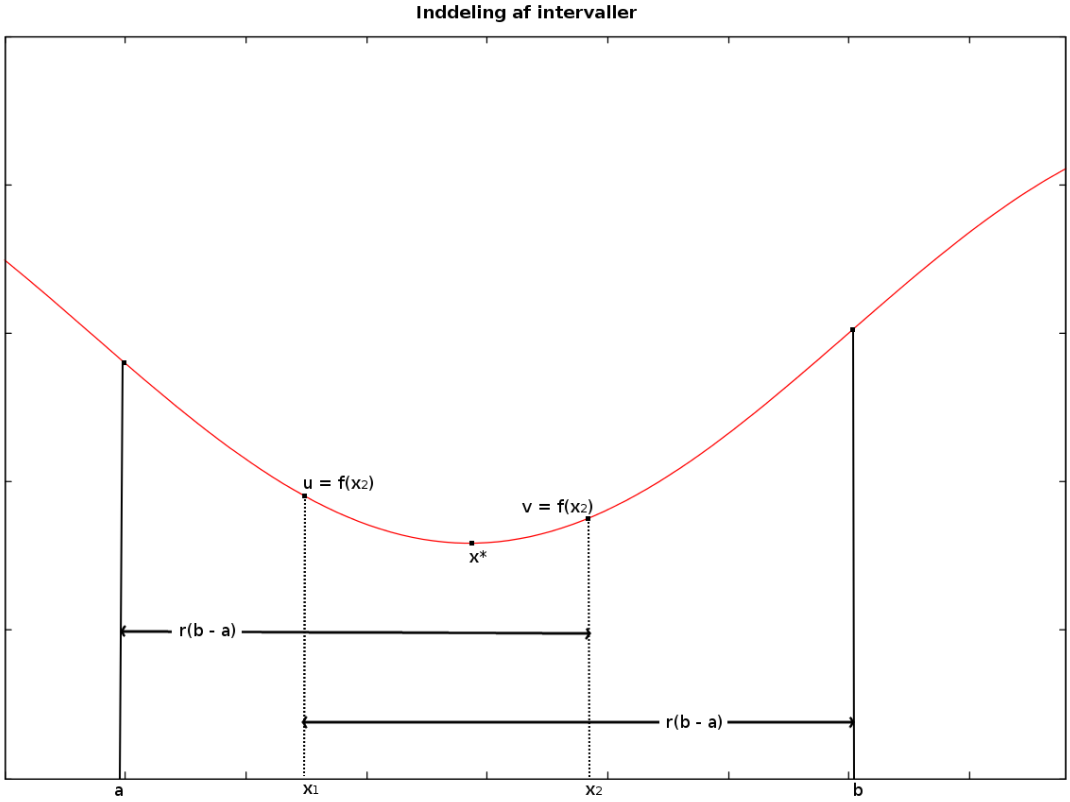
$$[a_0, b_0] \supset \cdots \supset [a_i, b_j], \text{ for } i, j \in \mathbb{N}.$$

Lad os betegne denne følge af intervaller $\{I_n\}_{n\in\mathbb{N}}^{\infty}$, hvor n angiver det n'te delinterval. Algoritmen instantieres med intervallet $I_0 = [a_0, b_0]$ og en objektfunktion $f : I_0 \subset \mathbb{R} \to \mathbb{R}$, hvor det gælder at der eksisterer et $x \in I_0$ således at $f(x) = x^*$. Det gælder desuden at $x^* \in I_n$ for alle $n \in \mathbb{N}$, med andre ord sikrer algoritmen at intervallet, hvori minimummet eksisterer, indsnævres. Ideen er således at $\lim_{n=\infty} |I_n| = 0$, dvs. vi tilnærmer os et interval med kun ét element, og dette element må nødvendigvis være x^* ifølge overstående argument. I hvert trin skal algoritmen opdele et interval I_n i to delintervaler, og derefter vælge hvilket af disse to der er det næste interval I_{n+1}, som der skal arbejdes videre med. Lad os antage at

algoritme skal inddele intervallet $[a, b]$ som på figur A.1. Objektfunktionen f skal evalueres i to punkter x_1 og x_2 [Cheney and Kincaid, 2008]

$$\begin{cases} x_1 = a + r(b-a) & u = f(x_1) \\ x_2 = a + r^2(b-a) & v = f(x_2) \end{cases}$$

hvor $r = 1/\varphi$ og $r^2 + r - 1 = 0$, som har rødderne $\frac{1}{2}\left(-1+\sqrt{5}\right)$ og $\frac{1}{2}\left(-1-\sqrt{5}\right)$. Der er to tilfælde at betragte: enten gælder det at $f(x_1) > f(x_2)$ eller $f(x_1) \leq f(x_2)$. Lad os betragte det første, som på figur A.1, det andet tilfælde er analogt med det første. Da f er kontinuert og unimodal, så må x^* nødvendigvis ligge i intervallet $[a, x_2]$. Det interval bliver dermed startintervallet i næste iteration, og således fortsætter algoritmen indtil et tilstrækkeligt lille interval er nået. Bemærk at i intervallet $[a, x_2]$ har vi allerede



Figur A.1: Inddeling af intervallet $[a, b]$

en evaluering af f tilgængelig, nemlig $f(x_1)$. Bemærk at

$$a + r(x_2 - a) = x_1$$

da $x_1 - a = r(b-a)$, dermed kan vi genbruge en evaluering fra den tidligere iteration. Derfor skal vi nu kun evaluere f i punktet $a + r^2(x_2 - a)$. Det følger herfra, at vi i hver iteration blot skal udføre nogle værdierstatninger i en bestemt rækkefølge [Cheney and Kincaid, 2008]:

$$b := x_1,\ x_1 := x_2, u := v,\ x_2 := a + r^2(b-a) \text{ og } v := f(x_2).$$

Input: Objektfunktionen $f : \mathbb{R} \to \mathbb{R}$, der optimeres på.
Input: Nedre- og øvregrænseværdier a og b for et interval, hvor $x^* \in [a, b] \subset \mathbb{R}$.
Output: x^*

$\varphi := \frac{1}{2}\left(3 - \sqrt{5}\right)$;
$r := 1/\varphi$;
$x_1 := a + r(b-a), x_2 := a + r^2(b-a)$;
$u := f(x_1), v := f(x_2)$;
while $\neg\left(|x_2 - x_1| < \tau\left(|a| + |b|\right)\right)$ **do**
    if $u > v$ **then**
        $b := x_1,\ x_1 := x_2, u := v,\ x_2 := a + r^2(b-a)$ og $v := f(x_2)$;
    else
        $a := x_2,\ x_2 := x_1,\ v := u,\ x_1 := a + r(b-a)$ og $u := f(x_1)$;
    end
end
$x^* := (a+b)/2$;

Algoritme A.1: Pseudokode for GSS

Algoritme A.1 viser pseudokoden for hele algoritmen. Her er τ en positiv konstant, der agerer fejltolerance-parameter i algoritmen. Betingelsen $|x_2 - x_1| < \tau\left(|a| - |b|\right)$ tjekker størrelsen af intervallet relativt til midterværdien i intervallet, og giver dermed en god indikation af hvor god en approksimation af x^* vi har. Typisk kan τ vælges således $\tau = \sqrt{\varepsilon}$, hvor ε er den påkrævede præcision.

Gyldne snit søgning kan med fordel benyttes som linjeafsøgningsmetode i højere-ordens optimeringsproblemer, da den ikke kræver information fra gradienten, som potentielt kan være tidskrævende at udregne. Evalueringerne af f kan potentielt også være tidskrævende at udregne, men som vi har set kan mindst én evaluering genbruges per iteration, dvs. en implementering kan vha. caching realisere en effektiv linjeafsøgningsmetode med respekt til tiden for evalueringerne af f.