Maskininlärning för val av relevanta gener i färdigt klassificerade genetiska data

(Utkast för opponering)

Kandidatavhandling i datavetenskap
Markus Liljebäck, 36637
Handledare: Ion Petre
Övervakare: Sepinoud Azimi
Åbo Akademi
Fakulteten för naturvetenskaper och teknik
6.4.2016

Referat

Inom bl.a. genetik har det uppkommit nya, kraftfulla undersökningsmetoder som gett upphov till mycket mera detaljerade resultat än tidigare. Kostnaden hos dessa undersökningsmetoder leder dock ofta till att endast ett statistiskt sett litet sampel går att undersöka, vilket innebär problem i att utvinna den information som finns i dessa data. Problem förorsakas också att en stor mängd variabler som kommer med är irrelevanta för det som undersöks, vilket oundvikligen händer när mängden variabler stiger till flera tusen. I denna avhandling analyseras utvinningen av information ur denna typ av s.k. högdimensionella data, med fokus på genetiska data från undersökningar med mikromatriser. De data som används är genuttrycksnivåer från bröstcancer- och Alzehimerspatienter. Dessa data är färdigt klassificerade, och analysen går ut på att hitta den minimala mängden variabler (gener) som definierar de enskilda klasserna i dessa data. Problemställningen med att välja den minimala mängden relevanta variabler och metoder för att göra detta presenteras, och maskininlärningsalgoritmen Incremental Forward Feature Selection (IFFS) analyseras och implementeras på dessa data. Stödvektormaskiner för klassificering av data behandlas kort, och algoritmen Smooth Support Vector Machine (SSVM) implementeras för att testa den valda mängden variablers effektivitet som träningsmängd. Den bakgrundsinformation inom bl.a. genetik som är nödvändig för att förstå innehållet gås också igenom.

Nyckelord: genuttrycksdata, variabelval, stödvektormaskiner, IFFS, SSVM

Innehållsförteckning

1. Inledning

Detaljnivån i data som kan samlas in har ökat kraftigt i och med att nya och mer avancerade undersökningsmetoder har uppkommit, bl.a. inom genetik. Detta innebär möjligheter att hitta nya samband och information som man inte förr kunnat hitta, men också flera utmaningar. Dessa typer av undersökningar är ofta mycket dyra att utföra på grund av den utrustning och det expertkunnande som behövs för att utföra dem, vilket resulterar i att endast det ofta är endast ett litet sampel kan undersökas. Bland de data som samlas in finns också en stor mängd irrelevanta variabler, vilket gör det nästintill omöjligt för en expert inom området att utföra en analys av dessa data. Därför måste de variabler som karakteriserar dessa data utvinnas innan en ämnesspecifik analys kan utföras. Att gallra bort irrelevanta variabler är vanligtvis ett förbehandlingssteg för att uppnå bättre resultat med olika algoritmer för datautvinning. Det finns många metoder inom både maskininlärning och statistisk vetenskap för att göra detta. [1]

I denna avhandling implementeras maskininlärningsalgoritmerna Incremental Forward Feature Selection (IFFS) [2] och Smooth Support Vector Machine (SSVM) [3] för analys av denna typ av data, och resultaten de producerar evalueras. De data som analyseras i avhandlingen är genuttrycksvärden hos bröstcancer- och Alzheimerspatienter, som samlats in genom undersökning med mikromatriser. De data som används är färdigt uppdelade i kluster, och analysen går ut på att hitta den minimala mängden relevanta variabler som definierar de enskilda klustren. Algoritmerna är dock generella och inte specifikt skapade för att analysera genuttrycksdata, utan fungerar även för andra datamängder med litet sampel och en stor mängd variabler.

Denna analys är det första steget i förbehandling av data för ett projekt med målet att bättre kunna förstå särdragen hos de enskilda klustren i denna typ av data. Detta steg är att välja de minimala mängder gener som är relevanta för att identifiera de kluster de tillhör. Efter att dessa variabler har identifierats ska de kontinuerliga genuttrycksvärden omvandlas till diskret (binär) form. Det sista steget i förbehandlingen är att säkerställa att klustren är disjunkta. Projektets mål är att kunna hitta en unik signatur i disjunkt normalform (DNF) för varje enskilt kluster. Denna signatur skulle användas för att bättre förstå vad som karaktäriserar ett enskilt kluster och för att förutse vilket kluster ett nytt element tillhör.

Den matematiska notation som används är följande: Vektorer skrivs som bokstäver med en pil ($\vec{x}$), och skalärer som bokstäver utan pil (x). Längden (magnituden, 2-normen) på en vektor $\vec{x}$ betecknas $\|\vec{x}\|$. För att beteckna den i:te kolonn- eller radvektorn i en matris används notationen $\vec{x}^i$. Om det är fråga om en kolonn- eller radvektor specificeras skilt enligt sammanhanget. Matriser skrivs med stora bokstäver och antas vara i formen $A \epsilon \mathbb{R}^{m*n}$ om inte annat anges.

2. Bakgrundsinformation

2.1 Genetiska data

De data som analyseras i denna avhandling består av genuttrycksnivåer hos olika patienters gener. I detta avsnitt presenteras den biologiska bakgrundsinformation som är relevant för att förstå vad geners uttrycksnivåer innebär och varför de analyseras.

2.1.1 DNA och RNA

DNA är den molekyl som innehåller levande organismers genetiska information (arvsmassa eller genom), och finns som en kopia i varje cell hos organismen. Stommen för DNA-molekylen är två kedjor av socker- och fosfatmolekyler som länkas samman och bildar en spiral. Det är i molekylerna som länkar samman de två kedjorna som informationen DNA bär på lagras. Dessa molekyler är adenin (A), guanin (G), cytosin (C) och tymin (T). Det är ordningen på dessa molekyler i DNA-strängen som utgör arvsmassans kod. Följder bestående av tre av dessa molekyler bildar ”ord” (kodon), som i sin tur bildar aminosyror som reglerar cellens aktivitet. [4]

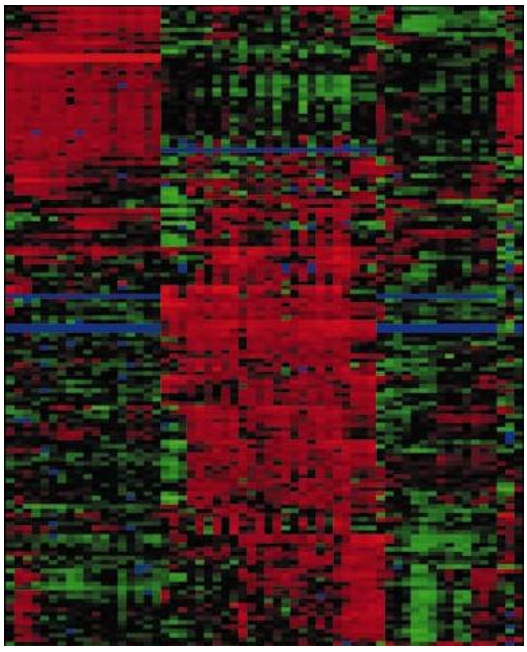
RNA är uppbyggt på samma sätt som DNA, men består endast av en stomme av socker och fosfat som ensamt bildar en spiral, till skillnad från DNA:s två stommar. Detta gör att RNA är en mindre stabil molekyl än DNA. RNA har flera funktioner, och RNA-molekyler brukar namnges enligt vilken funktion de uppfyller för tillfället. De typer som är mest relevanta för att förstå genuttrycksnivåer är budbärar-RNA (mRNA) och ribosom-RNA (rRNA). mRNA är den molekyl som cellerna använder för att kommunicera med ribosomerna, där proteiner bildas. rRNA är den ribosomernas huvudsakliga beståndsdel. [4]

2.1.2 Gener och genuttryck

En gen är en sekvens av DNA som innehåller den kod som behövs för att skapa ett visst protein, och därmed reglera verksamheten hos den cell som den är en del av. Processen att skapa protein består av två faser: transkription och translation. Under transkriptionsfasen skapas en mRNA-molekyl från strängen av DNA. Detta görs genom att DNA-strängen delar på sig tillfälligt och används som en mall för att skapa den nya mRNA-molekylen. Denna mRNA molekyl lämnar sedan cellens kärna, och används av ribosomerna för att skapa protein i translationsfasen. [5]

I en cell sker tusentals transkriptioner per sekund, men alla övergår inte till translationsfasen eftersom celler har flera avancerade regleringsmekanismer för att snabbt kunna anpassa sig till nya förhållanden. Exempelvis måste celler arbeta på olika sätt beroende på hur mycket och vilken typ av närning som finns tillgänglig. Regleringen kan dock även ske under senare stadier av processen. När celler delas så får de nya regleringsmekanismer för denna process, vilket har inverkan på vilken typ av cell som skapas och vilka celltyper som den nya cellen kan dela sig i. Mängden RNA och protein i en cell indikerar hur aktiv cellen är, och hur mycket RNA och protein som skapas ur en specifik gen kallas för genens uttrycksnivå. [5] På grund av detta är det av intresse att undersöka sambandet mellan uttryckta gener hos en viss cell och sjukdomar som t.ex. cancer och Alzheimers.

Geners uttrycksnivåer kan mätas i stor skala genom att använda mikromatriser. En mikromatris är ett mätinstrument där en spjälkt DNA-sträng jämförs med RNA-molekyler från en cell från ett provexemplar och en från ett kontrollexemplar. Dessa RNA-molekyler färgas vanligtvis så att RNA från provexemplaret är rött och RNA från kontrollexemplaret är grönt. Uttrycksnivåerna mäts sedan genom att analysera färgerna i matrisen. En röd färg innebär att provexemplaret har mycket RNA från en viss gen och en grön färg innebär att kontrollexemplaret har mycket RNA från genen. En gul färg innebär att båda har stor koncentration RNA från en viss gen och en svart färg innebär att varkendera har RNA för genen. Ett exempel på data från en mikromatrisundersökning kan ses i figur 1. Mängden RNA från en viss gen står inte i direkt korrelation med mängden protein som skapats från genen i fråga, men korrelationen är hög nog för att dra slutsatser om en gens uttrycksnivå från denna typ av undersökning. [6]



Figur 1: Exempel på ett segment av data från en mikromatrisundersökning. I denna bild utgör kolumnerna samplet och raderna generna, och de blåa punkterna indikerar data som saknas. Gränserna mellan samplets olika kluster kan tydligt ses i figuren. **Källa:** [6]

2.2 Högdimensionella data

Högdimensionella data har börjat uppkomma i stor grad i och med att mera avancerade undersökningsmetoder och möjligheter till insamling av data uppkommer inom olika områden. Undersökning med mikromatriser är ett exempel på en undersökningsmetod med hög genomströmning, dvs. att en stor mängd egenskaper kan analyseras på en och samma gång. Detta ger upphov till s.k. högdimensionella data, vilka kallas så på grund av hur ett element med n stycken variabler representeras av en n-dimensionell vektor i en matris när samplet analyseras. Högdimensionella data behandlas även inom många andra områden, exempelvis vid analys av textdokument eller finansiella data.[7]

Flera problem uppstår vid analys av högdimensionella data, som försvårar användningen av traditionella analysmetoder. Dessa problem brukar i en stor mängd verk sammanfattas som ett fenomen som kallas ”dimensionalitetens förbannelse” (eng. the curse of dimensionality). De problem som uppstår är exempelvis när data klassificeras för exakt för att ha någon praktisk nytta (eng. overfitting), och motsatta fall där data klassificeras alltför dåligt (underfitting). Traditionella analysmetoder är ofta skapade för att analysera datamängder där samplets storlek är större än antalet variabler hos de enskilda elementen. Högdimensionella data består av en stor mängd irrelevanta och högt korrelerade variabler, dock betyder inte en hög korrelation mellan variablerna nödvändigtvis att någon av dem är irrelevant. Traditionella metoder brukar ofta göra missklassificeringar på grund av att grupper av högt korrelerande

variabler sticker ut i en stor mängd irrelevanta variabler. För att handskas med högdimensionella data finns flera analysmetoder som inte påverkas av datamängdens dimensionalitet, exempelvis stödvektormaskiner.[7][8]

2.3 Presentation av datamängderna

De data som analyseras i denna avhandling består av genuttrycksnivåer i gener hos både sjuka och friska personer. Dessa data har samlats in genom undersökning med mikromatriser, och publicerats som fritt tillgängliga. Genuttrycksnivåerna i databaserna representeras av positiva, reella tal.

Den första datamängden som analyseras består av genuttrycksnivåer i tumörer hos bröstcancerpatienter. Samplet består av 47 patienter, och för varje patient finns genuttrycksnivåer för över 54000 gener. Datamängden är delad i fyra kluster: sporadiska basalliknande tumörer, BRCA1-associerade tumörer, icke-basalliknande tumörer och normal vävnad. Basalliknande bröstcancer är en aggressiv typ av cancer som utgör 10-15% av all sporadisk, dvs. icke-ärftlig, bröstcancer hos människor. BRCA1-typen av cancer är en ärftlig form av basalliknande cancer som förorsakas av mutationer i BRCA1-genen, och har påvisade samband med den sporadiska formen av basalliknande cancer. [9]

Den andra datamängden som analyseras består av genuttrycksnivåer från hjärnor hos Alzheimerspatienter och friska personer. Samplet består av 160 patienter, och för varje patient finns genuttrycksnivåer för över 54000 gener. För denna analys beaktas endast om patienten varit frisk eller lidit av Alzheimers sjukdom, vilket delar datamängden i två kluster. Det finns dock även information som bl.a. patienternas ålder, kön och etnicitet tillgänglig i datamängden. Mätningarna har gjorts på vävnader från 6 olika delar av hjärnan för att täcka hjärnans olika funktionella områden och de områden som påverkas mest av Alzheimers sjukdom. [10]

3. Problemställning och metoder

I denna avhandling testas en algoritm för att välja ut den minimala mängden variabler som definierar klustren i färdigt klassificerad data. Denna algoritm kräver en klassificeringsalgoritm för att avgöra om den valda mängden variabler är minimal, och för detta används en algoritm som hör till *stödvektormaskiner*-gruppen av algoritmer. I detta avsnitt presenteras problemställningen med val av variabler, metoder för att göra detta samt

bakgrundsinformation om stödvektormaskiner och motiveringar för att använda dem för detta ändamål.

3.1 Att välja den minimala mängden relevanta variabler

Att välja de relevanta variablerna i en datamängd är ett förbehandlingssteg för att göra en datamängd lättare att arbeta med. Det ökar snabbheten och korrektheten hos maskininlärningsalgoritmer, och motverkar problemet där data klassificeras i för stor detalj för att vara användbart i undersökningssammanhang. Datamängden blir också lättare att visualisera och göra förståelig för de som ska arbeta med den. [7] Det är dock viktigt att vara försiktig när variabler plockas bort. I flera fall, exempelvis med genetiska data, är det inte acceptabelt om en viktig variabel går förlorad under förbehandlingen. Om en gen som är relevant för att beskriva en sjukdom försvinner så vilseleder det forskaren som ska arbeta med dessa data, och leder troligtvis till att felaktiga slutsatser dras. [1]

För att lösa problemet med att hitta den minimala mängden variabler har det uppkommit en stor mängd olika metoder. De metoder som finns för att välja den minsta möjliga mängden relevanta variablerna kan huvudsakligen delas in i tre kategorier. De två traditionella grupperna av metoder är wrapper- och filtermetoderna, och en nyare grupp av metoder som visat sig vara effektiva kallas för inbäddade metoder. Algoritmen som används för val av variabler i denna avhandling, IFFS, hör till wrapper-klassen av metoder [2], och därför behandlas denna klass mera djupgående än de andra metoderna.

3.1.1 Filtermetoder

Filtermetoderna använder sig av en funktion för att räkna ut vikten för alla variablerna i datamängden, och sorterar sedan variablerna utgående från denna vikt. Det är huvudsakligen funktionen som räknar ut denna vikt som skiljer åt de olika metoderna i denna grupp. Filtermetoderna är beräkningsmässigt effektiva, eftersom de endast består av en enkel beräkning per variabel och ett sorteringssteg. De har dock vissa krav på oberoende mellan variablerna för att ge användbara resultat, vilket minskar deras användningsområden. Filtermetoderna är helt åtskilda från den algoritm som senare ska utnyttja datamängden som förbehandlas. [7] [2]

3.1.2 Inbäddade metoder

De inbäddade metoderna kallas så för att gallringen av irrelevanta variabler är "bäddat in" i själva maskininlärningsalgoritmen. Konceptet att förenkla beräkningar på detta sätt är inte nytt i sig, men dess användning för att behandla data för maskininlärning är relativt nytt. De är starkt beroende av maskininlärningsalgoritmen de bäddas in i, vilket leder till att en ny implementation måste skapas för varje algoritm de används med. Detta leder till att de ofta är svåra att använda, trots sin påvisade effektivitet. [7]

3.1.3 Wrapper-metoder

Wrapper-metoderna tar med klassificeringsalgoritmen de behandlar datamängden för som en del av processen att välja ut variabler, men till skillnad från de inbäddade metoderna så är de två algoritmerna helt åtskilda. Generellt, så prövar en wrapper-metod med hjälp av klassificeringsalgoritmen hur mycket vissa variabler förbättrar klassificeringen, och väljer de variabler som bidrar mest. Att prova alla olika kombinationer av variabler är dock ett problem som är NP-hårt, dvs. som inte kan göras i polynomiell tid. På grund av detta är det viktigt att avgränsa sökningen på ett sätt som inte skadar resultatets korrekthet, vilket utgör den huvudsakliga åtskiljande faktorn hos olika metoder inom wrapper-klassen. [7] [2]

Avgränsningen sker vanligtvis genom att variabler jämförs i en ordning bestämd utgående från något kriterium, och att sökningen stoppas när bättre resultat inte längre kan nås. Hur bra en klassificering är testas genom att låta maskininlärningsalgoritmen klassificera data utgående från en mängd av variabler och jämföra resultatet med datamängdens riktiga klassificering. Detta gör att wrapper-algoritmerna för val av variabler är universella; de använder maskininlärningsalgoritmerna endast som verktyg för att testa klassificeringen av data och därför kan godtyckliga klassificeringsalgoritmer och wrapper-algoritmer kombineras. [7] [2]

3.2 Stödvektormaskiner

Stödvektormaskiner (eng. Support Vector Machines, SVM) är en grupp av algoritmer för klassificering av data. De hör till kategorin övervakade inlärningsalgoritmer (eng. supervised learning algorithms), vilket innebär att de måste tränas med en färdigt klassificerad datamängd före användning. SVM-algoritmerna populära eftersom de är effektiva, ger exakta resultat och fungerar även bra på datamängder utan att man behöver anpassa dem skilt för de data som undersöks. SVM-algoritmerna påverkas heller inte av dimensionaliteten hos data, vilket gör att de lämpar sig väl för analys av högdimensionella data. Nackdelarna med

stödvektormaskiner är att de i sin grundform endast klassificerar data binärt, och att de ofta är matematiskt mycket mera komplicerade än flera andra klassificeringsmetoder. Det finns en mycket stor mängd olika SVM-algoritmer för linjära och icke-linjära datamängder, som hanterar problem som exempelvis för noggrann klassificering på olika sätt. [11][12]

SVM-algoritmerna skapar en binär klassificering av data genom att hitta det optimala separerande hyperplanet mellan klustren utgjorda av datamängdens element, representerade som punkter i n-dimensionell rymd. För att hitta detta hyperplan används elementen som ligger på gränsen till det kluster de tillhör, och vektorerna som representerar dessa element kallas för stödvektorer. Det optimala separerande hyperplanet är det hyperplan som har största möjliga marginal till de båda klustren, vilket gör att element i klassificeringsskedet kan ligga närmare det motsatta klustret än det egna klustrets stödvektorer gör, och fortfarande bli korrekt klassificerade. Det att stödvektormaskinerna endast använder elementen på klustrens gränser som stödvektorer gör att de lämpar sig väl för att hitta den minimala mängden relevanta variabler hos en datamängd. [11][12]

Nedan presenteras de grundläggande matematiska koncepten för en generell SVM, samt vad hur icke-linjärt separerbara datamängder hanteras med kärnfunktioner och genom att skapa en SVM med mjuk marginal. Alla steg som tas för att uppnå en viss formel presenteras dock inte, utan målet är att ge en överblick av hur SVM-algoritmerna formuleras matematiskt. För en full förståelse av stödvektormaskiner krävs kunskaper om vektor- och matrisberäkningar, hyperplan, optimering, Lagrangemetoder och kärnmetoder. Eftersom avhandlingens fokus ligger på val av variabler ur färdigt klassificerade datamänger så behandlas dessa områden inte mer djupgående.

3.2.1 Hyperplan

Ett hyperplan i n dimensioner är ett objekt bestående av $n-1$ dimensioner, exempelvis är ett hyperplan i 2 dimensioner en (1-dimensionell) linje och ett hyperplan i 3 dimensioner ett (2-dimensionellt) plan. Ekvationen för ett hyperplan i godtycklig dimension är $\vec{w}^T\vec{x} + b = 0$, där vektorernas dimensionalitet avgör vilken dimension detta hyperplan är i. I denna ekvation är vektorn $\vec{w}$ vinkelrät mot hyperplanet, och anger därmed hyperplanets lutning. Skalären b anger hyperplanets position utefter linjen som anges av $\vec{w}$, och genom att ändra på b så fås alla hyperplan som är parallella till detta. Den okända vektorn $\vec{x}$ går från origo till en godtycklig punkt på hyperplanet, vilket innebär att alla värden på $\vec{x}$ som tillfredställer ekvationen är punkter på hyperplanet. [12][13]

Målet vid klassificering är att hitta ett hyperplan som uppfyller kraven $\vec{w}^T\vec{x}_p + b \geq +1$ för alla element t i klassen $+1$, och $\vec{w}^T\vec{x}_p + b \leq -1$ för alla element i klassen -1, för $p = 1, \dots, m$. Klasserna ges identifierarna $+1$ och -1 för att dessa krav ska kunna sammanfattas till $y_p(\vec{w}^T\vec{x}_p + b) \geq 1$, där y_p är klassen som element $\vec{x}_p$ hör till. Det att förtecknet är det enda som skiljer klasserna åt möjliggör denna förenkling. Det finns antingen oändligt många hyperplan som uppfyller detta krav, eller inget alls om datamängden inte är linjärt separerbar. [12][13]

3.2.2 Optimeringsproblem

Optimeringsproblem är matematiska problem som går ut på att hitta den bästa möjliga lösningen för ett problem ur mängden av alla möjliga lösningar. Metoder för att göra detta effektivt använder sig av egenskaper som problemet i fråga har. Att hitta det optimala separerande hyperplanet är ett optimeringsproblem som går ut på att hitta det hyperplan som bäst separerar datamängden ur den oändliga mängden hyperplan som uppfyller kravet $y_p(\vec{w}^T\vec{x}_p + b) \geq 1$. De egenskaper som kan utnytjas för att hitta en lösning på detta problem är att det är *konvext*, och att det går att formulera som ett *kvadratiskt programmeringsproblem* (eng. Quadratic Programming, QP). [12]

Att en funktion är konvex innebär att om en graf skapas utifrån den så kan man dra en linje mellan två valfria punkter på grafen så ligger funktionens minsta punkt antingen under eller på denna linje. Grafen utgör då en "dal" utan kurvor, och har bara en lokal minsta punkt. Antagandet man kan göra utifrån detta är att varje lokalt minimum är ett globalt minimum, vilket utnyttjas vid lösandet av detta optimeringsproblem. QP-problem är en klass av optimeringsproblem där funktionen som ska optimeras är kvadratisk, men har linjära begränsningar. Det finns flera metoder för att lösa denna typ av problem, vilket olika varianter av stödvektormaskiner utnyttjar. [12]

3.2.3 Det optimala separerande hyperplanet

För att hitta det optimala separerande hyperplanet måste marginalen till de båda klustren maximeras. Detta kan göras genom att minimera längden på $\vec{w}$ under de begränsningar som ställs av stödvektorerna. Ett sätt att skriva detta är: minimera $\frac{1}{2}\|\vec{w}\|^2$ så att $y_p(\vec{w}^T\vec{x}_p + b) \geq 1$ för $p = 1, \dots, m$. Ur denna formulering kan den s.k. *grundformuleringen* (eng. primal formulation) för SVM-algoritmer utvinnas, och den blir: minimera $\frac{1}{2}\sum_{i=1}^{n}\vec{w}_i^2$ så att $y_p(\vec{w}^T\vec{x}_p + b) \geq 1$ för $p = 1, \dots, m$. I denna formulering är dock optimeringsproblemet

beroende av mängden variabler i datamängden, vilket är opraktiskt vid analys av högdimensionella datamängder. [12][13]

Det finns en alternativ formulering som kallas den *dubbla formuleringen* (eng. dual formulation) för SVM-algoritmer, som är beroende av sampelstorleken istället för antalet variabler per element. Denna egenskap gör att den lämpar sig därför bättre för högdimensionella datamängder. Detta uppnås genom att formulera om problemet så att elementvektorernas skalära produkter utnyttjas istället för att variabelvektorerna skulle behandlas enskilt. Formuleringen skapas genom att omforma grundformuleringen med hjälp av Lagrangemultiplikatorer. Denna formulering är: maximera $\sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j$ så att $\alpha_i \geq 0$ och $\sum_{i=1}^{m} \alpha_i y_i = 0$. I denna Lagrangefunktion $\Delta(\vec{\alpha})$ är $\vec{\alpha}$ en m-dimensionell vektor bestående av Lagrangemultiplikatorerna som används för att definiera $\vec{w}$. [12][13]

3.2.4 Icke-linjär klassificering

Alla datamängder är dock inte linjärt separerbara, och för att kunna klassificera dessa datamängder kan antingen en SVM med *mjuk marginal* eller, *kärnfunktioner* (eng. kernel functions) användas för klassificering av datamängden. En SVM med mjuk marginal lämpar sig för datamängder där klustren endast överlappar lite, medan kärnfunktioner används för fullständigt icke-linjärt separerbara datamängder. Det är önskvärt att en linjär klassificering används på linjärt separerbar data, eftersom en icke-linjär klassificering har större risk att bli för noggrann. [12]

Att en SVM har mjuk marginal innebär att den tillåter mindre felklassificeringar. Detta görs genom att tilldela en slackvariabel ξ_i till varje element. Den modifierade grundformuleringen för SVM-algoritmer blir då: minimera $\frac{1}{2}\sum_{i=1}^{n} \vec{w}_i^2 + c\sum_{i=1}^{m} \xi_i$ så att $y_p\left(\vec{w}^T \vec{x}_p + b\right) \geq 1 - \xi_p$ för $p = 1 \ldots m$, och den dubbla formuleringen blir: minimera $\sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j$ så att $\alpha_i \leq 0 \leq c$ och $\sum_{i=1}^{m} \alpha_i y_i = 0$ för $i = 1 \ldots m$. I dessa formuleringar är c en konstant som används för att reglera hur stor inverkan slackvariablerna har på hyperplanets position. Ett stort värde på c gör klassificeringen lik den som skulle skapas med en fullständigt linjär SVM, och ett litet värde på c gör att hyperplanet flyttas kraftigt om klustren överlappar varandra. Märk att ξ_i faller bort vid skapandet av den dubbla formuleringen, och endast c är kvar för att påverka marginalen. [12][13]

Kärnfunktioner är kapabla av att skapa icke-linjära klassificeringar av data utan att tvinga användaren att acceptera missklassificeringar. Kärnfunktionerna fungerar genom att

länka punkterna representerade av datamängdens n-dimensionella elementvektorer till punkter i dimension $n + 1$, så att en linjär klassificering kan utföras på denna omvandlade datamängd. Denna linjära klassificering motsvarar en icke-linjär klassificering i den ursprungliga dimensionen. Det finns en stor mängd olika kärnfunktioner som kan användas för detta, vilket vidare ökar mängden variationer som kan skapas av SVM-algoritmer. [12][13] I denna avhandling används den gaussiska kärnfunktionen, vilken beskrivs närmare i avsnittet där klassificeringsalgoritmen SSVM behandlas.

4. IFFS

Algoritmen Incemental Forward Feature Selection(IFFS) är en algoritm för att lösa problemet med att välja ut de variabler hos en mängd data som karakteriserar datamängden utan att relevanta variabler går förlorade (eng. the feature selection problem). Algoritmen är skapad av Y-J. Lee, C-C. Chang och C-H. Chao. IFFS hör till wrapper-kategorin av algoritmer för detta ändamål och använder sig av en klassificeringsalgoritm för att evaluera den valda mängden variabler. Detta gör att resultatet som IFFS producerar kan därför variera beroende på vilken klassificeringsalgoritm som används, även om datamängden som behandlas är den samma. Klassificeringsalgoritmen som väljs bör vara en övervakad inlärningsalgoritm, eftersom målet med IFFS är att skapa en minimal träningsmängd för denna typ av algoritmer.[2]

4.1 Grundidé

Grundidén bakom IFFS är välja variabler genom att mäta hur mycket information de tillför till klassificeringen. Variablerna representeras av vektorer med de värden som de tar för varje element i samplet. Hur mycket information en viss variabel tillför mäts genom att mäta distansen från denna vektor till kolonnrummet av den matris som innehåller de variabler som hittills valts. För att avgöra om denna variabel tillsätts så används ett värde δ som definieras av användaren, vilket utgör den minsta distansen för att en variabel skall tillsättas. Efter att alla variabler gåtts igenom med detta värde på δ så testas klassificeringssäkerheten med hjälp av klassificeringsalgoritmen som används. Detta upprepas med minskande värden på δ tills klassificeringens felprocent inte längre minskar.

4.2 Matematisk beskrivning

Indexmängden S innehåller indexen för de variabler som väljs av algoritmen. Matrisen $A \in \mathbb{R}^{m*n}$ representerar datamängden som algoritmen behandlar och $A^S \in R^{m*|S|}$ är den

matris som består av kolonnvektorerna ur A som definieras av indexen i S. Notera att kolonnvektorerna representerar variablerna och att radvektorerna representerar elementen i samplet. Detta krävs för att uppfylla kravet på en bred matris, som ställs av de matrisoperationer som utförs på datamängden. I detta avsnitt betecknar A^j den j:te kolonnvektorn i A. För att begränsa sökningen IFFS gör används ett värde $\delta > 0$. δ och S kan manipuleras av användaren, vilket möjliggör expertinput så att algoritmen kan optimeras för en viss datamängd.

Informationen som tillförs av varje enskild variabel med indexet $j \notin S$ testas genom att beräkna distansen r_j från vektorn A^j till kolonnrummet för A^S. Detta beräknas som $\left\|A^S\vec{\beta}^* - A^j\right\|$, där $\vec{\beta}^*$ är den närmaste möjliga lösningen för $\vec{\beta}$ i minsta kvadrater problemet $A^S\vec{\beta} - A^j$. Ett minsta kvadrater problem går ut på att hitta den närmaste möjliga lösningen $\vec{x}^*$ för en vektor $\vec{x}$ i ett linjärt ekvationssystem $A\vec{x} = \vec{b}$ som saknar lösning. Att denna typ av ekvationssystem saknar lösning innebär att $\vec{x}$ inte är i kolonnrummet för A. Ekvationen $A^S\vec{\beta} - A^j$ kan genom att addera A^jtill båda sidorna omvandlas till $A^S\vec{\beta} = A^j$, vilket är ett linjärt ekvationssystem i formen $A\vec{x} = \vec{b}$. Att hitta $\vec{\beta}^*$ för detta system innebär då att hitta den punkt i kolonnrummet för A^S som är närmast A^j. Att lösa $\left\|A^S\vec{\beta}^* - A^j\right\|$ kan då sammanfattas som att hitta längden på den vektor som går från A^j till den närmaste punkten $\vec{\beta}^*$ i kolonnrummet för A^S. Det är denna längd som används för att bedöma hur mycket information variabeln tillför. Det finns flera metoder för att hitta lösningar till minsta kvadrater problem, men de behandlas dock inte i denna avhandling.

Efter att informationen som tillförs har mätts för alla variabler som inte redan finns i S så testas felprocenten för den klassificering som kan skapas utgående från de valda variablerna. Efter att omklassificeringen gjorts av den algoritm som valts för detta ändamål av användaren är beräkningen av dess felprocent en enkel process. Den egentliga klassificeringen och den nya klassificeringen jämförs och antalet felklassificerade element divideras med det totala antalet element. Om felprocenten har förändrats så minskas δ med 1 och distansberäkningarna utförs på nytt för alla variabler. När felprocenten inte längre sjunker så har den minimala mängden variabler som krävs för att korrekt klassificera mängden funnits.

4.3 Pseudokod

Nedan i tabell 1 presenteras algoritmen IFFS i pseudokod. Att ge ett initialt värde på S är valfritt, men S kan inte vara tom för beräkningen av r_j i steg 5, därför är kontrollen i steg 1 nödvändig. Den första slingan i steg 2 kan implementeras som en for-loop där δ minskas varje iteration. Den andra slingan kan också implementeras som en for-loop som går igenom alla värden på j, men en extra kontroll behövs för att hoppa över de värden på j som redan finns i S. I steg 7 tränas och körs den valda klassificeringsalgoritmen, vilket är SSVM för implementationen i denna avhandling. Felprocenten räknas ut genom att jämföra resultatet från klassificeringen med den ursprungliga klassificeringen.

Input: Datamängden, en matris $A \in \mathbb{R}^{m*n}$
Den ursprungliga klassificeringen
Den initiala indexmängden av valda variabler

Output: Den slutliga indexmängden av valda variabler
Den slutliga felprocenten för klassificeringen

1. Om S är tom, tillsätt 0 till S.
2. Upprepa stegen 3-7 tills felprocenten inte längre minskar eller alla värden för δ testats.
3. Skapa submatrisen A^S bestående av de variabler vars index finns i S.
4. Upprepa stegen 5-6 för alla $j = 0, \dots, n$, där $j \notin S$.
5. Beräkna distansen r_j från j till kolumnrummet för A^S.
6. Om r_j är större än δ, tillsätt j till S och uppdatera A^S.
7. Skapa en klassificering utgående från A^S och beräkna felprocenten.
8. Returnera den slutgiltiga indexmängden S och felprocenten.

Tabell 1: Pseudokod för IFFS.

4.3 Komplexitetsanalys

IFFS består av två slingor, med den ena nästlad i den andra. Den första minskar på δ varje iteration, och körs δ gånger eller tills felprocenten inte längre minskar. δ är ett användardefinierat värde, och är således oberoende av datamängdens storlek. Den andra slingan testar informationsvärdet hos enskilda variabler, och är därför beroende av datamängdens storlek. För varje δ körs den inre slingan $n - |S|$ gånger, dvs. en gång för varje

variabel som inte redan valts. I det bästa fallet, dvs. att den minimala variabelmängden hittas under första värdet på δ, kommer den yttre slingan att köras 2 gånger. I det värsta fallet, dvs. att en minimal mängd variabler inte kan hittas, körs den yttre slingan δ gånger. Detta ger en körtid på $O(n)$ i både bästa och värsta fallet, när klassificeringsalgoritmen inte tas i beaktande.

Att IFFS använder sig av en godtycklig klassificeringsalgoritm gör komplexiteten hos IFFS beroende av den använda algoritmens komplexitet. Klassificeringsalgoritmen körs inom den yttre slingan, och därför påverkar inte datamängdens storlek antalet gånger den körs. Algoritmens input är dock beroende av datamängden. Algoritmen kommer först att tränas med datamängden A^S bestående av de för tillfället valda variablerna, och sedan köras med den fulla datamängden A. Träningsmängden A^S kommer alltid att vara mindre än eller lika stor som A, och består i alla praktiska fall av endast en trivial mängd data jämfört med den fulla datamängden. Körtiden för IFFS kommer att bestämmas av klassificeringsalgoritmen om dess komplexitet är större än $O(n)$.

5. SSVM

Algoritmen Smooth Support Vector Machine (SSVM) är en stödvektormaskin för klassificering av data. Algoritmen är skapad av J-Y. Lee och O.L. Mangasarian.[3] Algoritmen presenteras här kortfattat och ytligt, mer detaljerad information och bevis för beräkningarna finns i artikeln där SSVM presenterats. Alla matematiska koncept som används i algoritmen behandlas inte heller i denna avhandling.

5.1 Grundidé

Grundidén bakom SSVM är att använda utjämning (eng. smoothing) för att bättre klassificera data. Genom utjämning generaliseras datamängdens egenskaper och att en för exakt klassificering skapas undviks. Att skapa en generell klassificering gör att algoritmen ger större korrekthet i egentliga klassificeringssituationer, även om hyperplanet som skapas i träningsskedet inte är det bästa möjliga för träningsdatamängden som används. SSVM har formulerats i både en linjär och icke-linjär form.

...

5.2 Matematisk beskrivning

5.2.1 Linjär SSVM

...

5.2.2 Newton-Armijo algoritmen

...

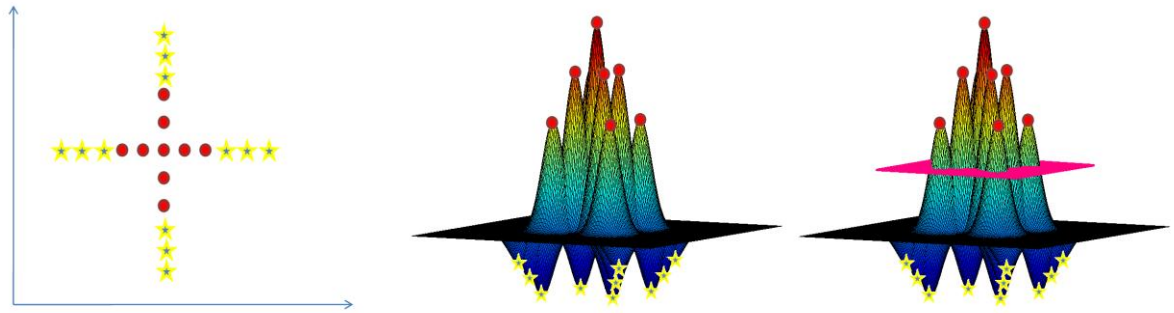
5.2.3 Icke-linjär SSVM

...

5.2.4 Den gaussiska kärnfunktionen

Den gaussiska kärnfunktionen är skapad av och namngiven efter Carl Friedrich Gauss. Intuitivt fungerar den genom att placera datamängdens element i en högre dimension och "höja upp" elementen i ena klassen och "sänka ner" elementen i den andra klassen. En illustration av hur detta fungerar för en 2-dimensionell datamängd kan ses i figur 2. När detta gjorts kan en linjär klassificering av data utföras i den högre dimensionen. Denna klassificering ses som icke-linjär i den lägre dimensionen, och dess gränser utgörs av platserna där det separerande hyperplanet konvergerar med det modifierade planet som utgörs av datamängdens element.

...



Figur 2: Exempel på hur den gaussiska kärnfunktionen fungerar på en 2-dimensionell, icke-linjärt separerbar datamängd. Punkterna i 2-dimensionell rymd länkas till punkter i 3 dimensionell rymd, och en linjär klassificering utförs på denna transformerade datamängd. Den transformation som skapats i träningsskedet avgör vart punkter länkas i en egentlig klassificeringssituation. **Källa: [12]**

5.3 Pseudokod

...

5.4 Komplexitetsanalys

…

6. Resultat och diskussion

…

7. Avslutning

…

Källhänvisningar

[1] C. Wang, J. Gevertz, C.-C. Chen och L. Auslender, *Finding Important Genes from High-Dimensional Data: An Appraisal of Statistical Tests and Machine-Learning approaches,* arXiv:1205.6523 [stat.ML], 2012.

[2] Y.-J. Lee, C.-C. Chang och C.-H. Chao, ”Incremental Forward Feature Selection with Application to Microarray Gene Expression Data,” *Journal of Biopharmaceutical Statistics,* pp. 827-840, 10 September 2008.

[3] Y.-J. Lee och O. Mangasarian, ”SSVM: A Smooth Support Vector Machine for Classifiaction,” *Computational Optimizations and Applications,* vol. 20, nr 1, pp. 5-22, 2001.

[4] ”University of Leicester - Virtual Genetics Education Centre,” [Online]. Available: http://www2.le.ac.uk/departments/genetics/vgec/schoolscolleges/topics/dna-genes-chromosomes. [Använd 25 2 2016].

[5] ”Scitable by Nature Education,” 2014. [Online]. Available: http://www.nature.com/scitable/topicpage/gene-expression-14121669. [Använd 25 2 2016].

[6] A. Brazma och J. Vilo, ”Gene expression data analysis,” *FEBS Letters,* vol. 480, nr 1, pp. 17-24, 2000.

[7] I. Guyon och A. Elisseeff, ”An Introduction to Variable and Feature Selection,” *Journal of Machine Learning Research,* vol. 3, nr 1, pp. 1157-1182, 2003.

[8] R. Clarke, H. W. Ressom, A. Wang, J. Xuan, M. C. Liu, E. A. Gehan och Y. Wang, ”The properties of high-dimensional data spaces: implications for exploring gene and protein expression data.,” *Nature Reviews Cancer,* vol. 8, nr 1, pp. 37-49, 2008.

[9] A. L. Richardson, Z. C. Wang, A. De Nicolo, X. Lu, M. Brown, A. Miron, X. Liao, J. D. Iglehart, D. M. Livingston och S. Ganesan, ”X chromosomal abnormalities in basal-like human breast cancer,” *Cancer Cell,* vol. 9, nr 2, pp. 121-132, 2006.

[10] W. S. Liang, T. Dunckley, T. G. Beach, A. Grover, D. Mastroeni, D. G. Walker, R. J. Caselli, W. A. Kukull, D. McKeel, J. C. Morris, C. Hulette, D. Schmechel, G. E. Alexander, E. M. Reiman, J. Rogers och D. A. Stephan, ”Gene expression profiles in anatomically and functionally distinct regions of the normal aged human brain,” *Physiol Genomics*, vol. 28, nr 3, pp. 311-322, 2007.

[11] P. Harrington, Machine Learning in Action, Shelter Island, NY: Manning Publications Co., 2012.

[12] A. Statnikov, D. Hardin, I. Guyon och F. C. Aliferis, ”A Gentle Introduction to Support Vector Machines in Biomedicine,” i *AMIA*, San Francisco, 2009.

[13] E. Alpaydin, Introduction to Machine Learning, Second Edition, Cambridge, Massachusetts: The MIT Press, 2010.