KU LEUVEN kulak

Groep Wetenschap & Technologie Kulak

Bachelor fysica, informatica, wiskunde
ingenieurswetenschappen

X0B53A – Probleemoplossen en ontwerpen, deel 1
X0A22D – Informaticawerktuigen

Levens redden met machine learning: Support Vector Machines voor tumorclassificatie

Eindverslag van de teamopdacht

Team ∃uler

Vincent Van Schependom, Daan Vanhaverbeke, Jasper Benoit, Lasha Shergelashvili, Marie Taillieu, Zeineb Kharbach, Florian Degraeve, Younes Mebarki

Academiejaar 2023 – 2024

Inhoudsopgave

Inleiding

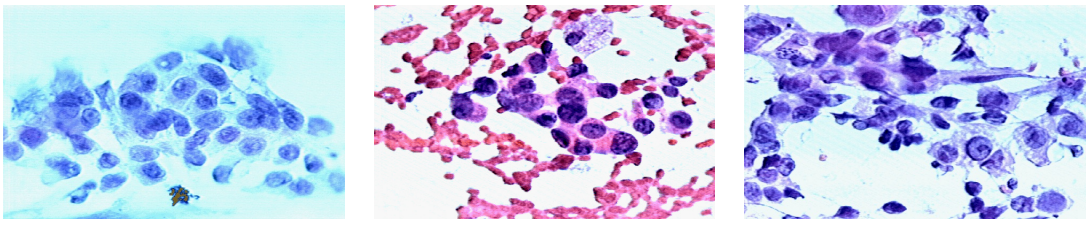
Dit is het eindverslag van de teamopdracht, waar we met alle leden van team ∃uler gedurende negen weken aan gewerkt hebben. We maakten doorheen de sessies kennis met machine learning en verwierven week na week meer kennis over dit onderwerp. We oefenden eerst op het onder de knie krijgen van twee basistechnieken, waarna we in week zes begonnen met het uitwerken van onze eigen machine learning toepassing.

Inhoud van dit eindverslag

We zullen in dit eindverslag de lezer eerst laten ontdekken wat machine learning precies is. Vervolgens bespreken we wat *supervised learning* inhoudt, in vergelijking met *unsupervised learning*. We bespreken ook wat classificatie juist is. Hierna zal het principe achter Support Vector Machines, de toepassing waarvoor we kozen, volledig uitgelegd worden. Hierbij zullen ook de nodige termen zoals *bias*, variantie en *cross-validation* aan bod komen. Daarna zullen we verschillende classificatietechnieken vergelijken en beargumenteren waarom voor de tumor-dataset SVM een geschikte keuze blijkt te zijn. Tot slot evalueren we de resultaten van SVM op onze dataset.

De gegevens

We maken in ons verslag gebruik van de resultaten van een onderzoek naar de verschillende eigenschappen van tumoren bij borstkankerpatiënten. Hierbij onderzochten wetenschappers W. Nick Street, W. H. Wolberg, en O. L. Mangasarian [1] in 1993 welke kenmerken de meeste invloed hadden in het al dan niet kwaadaardig zijn van een tumor. Deze kenmerken werden berekend op basis van een gedigitaliseerd beeld van een fijne naaldaspiraat (FNA) van de borstmassa. Dergelijke beelden zijn te zien in figuur 1. De data die in het onderzoek vergaard werd, werd nadien publiekelijk vrijgegeven [2], wat ons in de mogelijkheid stelt om de data te verwerken met hedendaagse machine learning technologie.



Figuur 1: Gedigitaliseerde beelden van fijne naaldaspiraten van de borstmassa. [1]

Hoofdstuk 1

Situering van SVM binnen machine learning

1.1 Wat is machine learning?

Machine learning is een tak binnen het gebied van de artificiële intelligentie, waarbij we een model trainen op basis van een gegeven dataset. Naarmate de training vordert, zal het model bepaalde verbanden beginnen leggen en bepaalde structuren beginnen herkennen in de data waarop het getraind wordt. In principe neemt de kwaliteit van het model toe wanneer de data, waarop het model zich baseert, toeneemt.

In dit verslag zal een model getraind worden dat verbanden zal zoeken in de dataset met tumoren van borstkankerpatiënten. Het model zal trachten een link te vinden tussen de verschillende kenmerken van deze tumoren en de klasse waartoe deze tumoren behoren; die van de goedaardige tumoren of die van de kwaadaardige.

1.2 Terminologie en notatie

De dataset beschikt over n verschillende datapunten. Deze individuele datapunten x_i hebben elk een eindig aantal kenmerken, die we *features*, *inputs* of *onafhankelijke variabelen* zullen noemen. We duiden de hoeveelheid *features* aan met p. Elk individueel datapunt x_i uit onze set van datapunten $x_1, x_2, ..., x_n$ is dus van de vorm $x_i(x_1, x_2, ..., x_p)$ (met $1 \leq i \leq n$). Deze features van de dataset kunnen we voorstellen aan de hand van de matrix X, die als volgt gedefinieerd wordt:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1p} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2p} \\ x_{31} & x_{32} & x_{33} & \dots & x_{3p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{np} \end{bmatrix}$$

Hierbij is X_{ij} de meetwaarde van de j-e feature voor het i-e datapunt met met $i = 1, 2, 3, ..., n$ en $j = 1, 2, 3, ..., p$. De tumordataset bevat gegevens over 569 tumoren, die elk 30 kenmerken hebben. In de dataset n dus gelijk aan 569 en p gelijk aan 30.

1.3 Supervised learning

Machine learning kan worden opgesplitst in twee grote categorieën; *supervised learning* enerzijds en *unsupervised learning* anderzijds. De naam van beide vormen geeft eigenlijk al weg wat deze vormen precies inhouden.

Bij *unsupervised learning* is er voor elk datapunt in de dataset zowel features als outputs voor handen. Er is sprake van zogenaamde 'input-outputparen'. We kunnen ons model trainen op het vinden van een verband tussen de x-waarden en de y-waarden in onze trainingsdataset. Het uiteindelijke doel is dan dat het model voor een nieuwe dataset, op basis van de *features* in die dataset, een output zal voorspellen.

Bij *unsupervised learning* zal de data enkel uit inputs of x-waarden bestaan en zijn er dus geen afhankelijke variabelen. Een *unsupervised learning*-model zal zelf patronen of structuren trachten te vinden in de data en vervolgens de data zelf onderverdelen in klassen. De details achter *unsupervised learning* liggen buiten de scope van dit rapport en zullen we dus achterwege laten.

1.3.1 Uitbreiding van de notatie

Aangezien we aan *supervised learning* doen, hebben we voor elke input x_i in de dataset ook een bijhorende *output* y_i. Er zijn dus n y-waarden $y_1, y_2, ..., y_n$, die - net zoals de *features* - matricieel kunnen worden voorgesteld als volgt:

$$Y = \begin{bmatrix} y_1 & y_2 & y_3 & \dots & y_n \end{bmatrix}$$

1.4 Classificatie

Classificatie in machine learning is een vorm van supervised learning waarbij het doel is om een input toe te wijzen aan een van de vooraf gedefinieerde klassen. De klassen worden vaak ook aangeduid als *label* of *categorie*. Het ultieme doel van classificatie is om te bepalen in welke categorie nieuwe gegevens zullen vallen.

We passen dit even toe op onze tumordataset. We willen aan de hand van classificatie onderzoeken wat de aard van een tumor is. Er zijn dus twee mogelijke categorieën waartoe een tumor behoort: die van de goedaardige tumoren of die van de kwaadaardige. Elke individuele output y_i is dus ofwel gelijk aan 1 (goedaardig), ofwel gelijk aan -1 (kwaadaardig). Omdat er slechts twee mogelijke klassen zijn, is er sprake van *binaire classificatie*. Er bestaan echter ook classificatietechnieken waar meerdere klassen aan bod komen.

Hoofdstuk 2

Het *maximum margin*-principe

Het SVM model zal trachten onze datapunten te scheiden in twee klassen. In onze dataset heeft elk datapunt waarde -1 of 1, afhankelijk van de klasse waartoe het punt behoort. Indien elk punt in de dataset p verschillende *features* heeft, zullen er $(p-1)$-dimensionele objecten, genaamd hypervlakken (of *hyperplanes*), berekend worden om de dataset in deze twee te groepen verdelen.

We construeren twee zo'n hypervlakken die de twee gegevensklassen scheiden, zodat de afstand daartussen zo groot mogelijk is. Het gebied dat wordt begrensd door deze twee hypervlakken wordt de marge genoemd. Deze marge zal zo groot mogelijk gemaakt worden door het model. Het hypervlak tussen de marge noemen we de beslissingsgrens. In het geval dat elk datapunt drie *features* heeft, zal zo'n hypervlak een 2-dimensionaal vlak zijn. In ons geval heeft elk datapunt echter twee *features*, dus kan de dataset gescheiden worden door een rechte, dat strikt genomen een 1-dimensioneel hypervlak is.

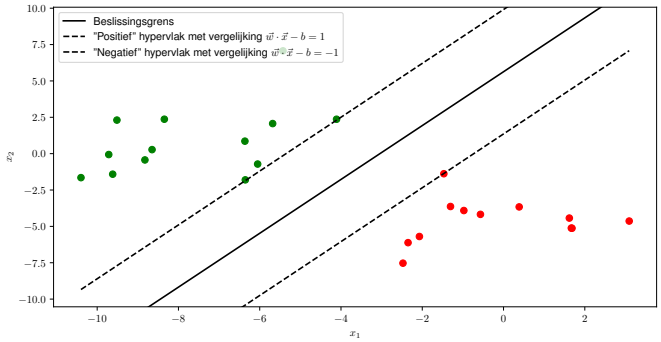
2.1 De hypervlakken

We stellen een voorschrift op voor de twee scheidingsrechten, die te zien zijn in figuur 2.1, waartussen zich een marge bevindt. Als $\vec{w}$ de normaalvector is op de hypervlakken, kunnen we de vergelijkingen van deze rechten schrijven als $\vec{w} \cdot \vec{x} - b = 1$ en $\vec{w} \cdot \vec{x} - b = -1$. Hierbij is b dan de *intercept* van de rechte die de beslissingsgrens beschrijft. De *intercept* is de y-waarde voor $x = 0$. Alle punten boven het eerste hypervlak worden geclassificeerd als horend tot de ene klasse. De punten onder het tweede hypervlak worden geclassificeerd als horende tot de andere klasse.

De marge is het gebied tussen deze twee hypervlakken. De breedte van hiervan is gelijk aan $\frac{2}{\|\vec{w}\|}$ en aangezien we die breedte willen maximaliseren, zullen we met andere woorden dus $\|\vec{w}\|$ trachten te minimaliseren.

2.2 De marge maximaliseren

Als de dataset twee lineair scheidbare 'wolken' vormt en er geen *outliers* - punten die niet tot de juiste wolk behoren - zijn, is het bepalen van de maximale marge vrij makkelijk. Het wordt moeilijker wanneer er wel *outliers* zijn en de twee wolken dus niet meer perfect lineair scheidbaar zijn, zonder dat punten aan de verkeerde kant van de twee hyperplanes belanden. Daarom voeren we een soort foutterm in.



Figuur 2.1: Twee lineair scheidbare wolken van punten. De kleuren van de punten duiden aan tot welke klasse ze behoren.

2.2.1 *Hinge loss*

De *hinge loss* is een soort foutterm die we toevoegen aan punten die niet goed of met geen grote zekerheid worden geclassificeerd. Deze *hinge loss* wordt berekend door de formule

$$\max\left(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b)\right)$$

Deze foutterm laat ons toe om de fouten te beperken, terwijl we de marge maximaliseren. Voor elk punt hebben we dan twee mogelijkheden wat de waarde van de *hinge loss* betreft:

1. Het punt met afhankelijke variabele y_i werd correct geclassificeerd.

   Dan ligt het punt dus aan de juiste kant van een van de twee hyperplanes. In dit geval is $y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1$ en zal $1 - y_i(\vec{w} \cdot \vec{x}_i - b) \leq 0$. Omdat de *hinge loss* het maximum neemt van 0 en dit laatste getal - dat ofwel ook 0, ofwel negatief is - zal de *hinge loss* in dit geval 0 zijn.

2. Het punt ligt in de marge of het punt ligt aan de verkeerde kant van de beslissingslijn.

   De *hinge loss* wordt dan $1 - y_i(\vec{w} \cdot \vec{x}_i - b)$. Indien het datapunt binnen de marge ligt, zal de hinge loss tussen 0 en 1 liggen. Het punt wordt dan met lagere zekerheid geclassificeerd, maar we willen het ook niet te hard bestraffen. Indien een datapunt buiten de marge, maar aan de verkeerde kant van de beslissingslijn ligt, zal de *hinge loss* groter dan 1 zijn. We willen het model wel hard bestraffen, want zo'n fouten willen we miniem houden.

2.2.2 De kostfunctie

Om het SVM-model te trainen, moeten we nu dus met twee zaken rekening houden. Enerzijds willen we $\|\vec{w}\|$ minimaliseren om een zo groot mogelijke marge te bekomen anderzijds willen we de *hinge loss* of strafterm zo klein mogelijk houden. We voeren een kostfunctie J in en zoeken het minimum $\min_{w,b} J$ van deze functie. Het minimalisatieprobleem wordt dan gegeven door de formule

$$\min_{w,b} J = \min_{w,b} \left[\frac{1}{n} \sum_{i=1}^{n} \max\left[0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b)\right] + \lambda \cdot ||\vec{w}||^2 \right]$$

Hierbij is $\frac{1}{n}\sum_{i=1}^{n}\max\left[0, 1-y_i(\vec{w}\cdot\vec{x}_i-b)\right]$ de gemiddelde *hinge loss*, dus de gemiddelde foutterm die werd toegekend over alle n datapunten in de dataset. De tweede term $\lambda\cdot||\vec{w}||^2$ regelt het evenwicht tussen enerzijds het minimaliseren van de fouten op de trainingsdata en anderzijds het maximaliseren van de marge. λ is een metaparameter van het model en deze parameter vertelt aan het model hoeveel belang we hechten aan het minimaliseren van $\|\vec{w}\|$.

Hoofdstuk 3

De regularisatieparameter λ

3.1 Wat is een metaparameter?

Machine learning modellen hebben over het algemeen een of meerdere *metaparameters* die de flexibiliteit van het model beïnvloeden. Het model zal zich meer of minder aanpassen aan de trainingsdata naar gelang de waarden van deze metaparameter(s). Deze parameter wordt voor SVM ook wel de *regularisatieparameter* genoemd en wordt aangeduid met λ. Deze λ zal uiteindelijk de breedte van de marge - en dus de variabiliteit van het model - beïnvloeden (zie later).

In het geval van SVM noemen we de metaparameter λ ook wel de *regularisatieparameter*. Deze parameter bepaalt de breedte van de marge en dus ook de variabiliteit van het model:

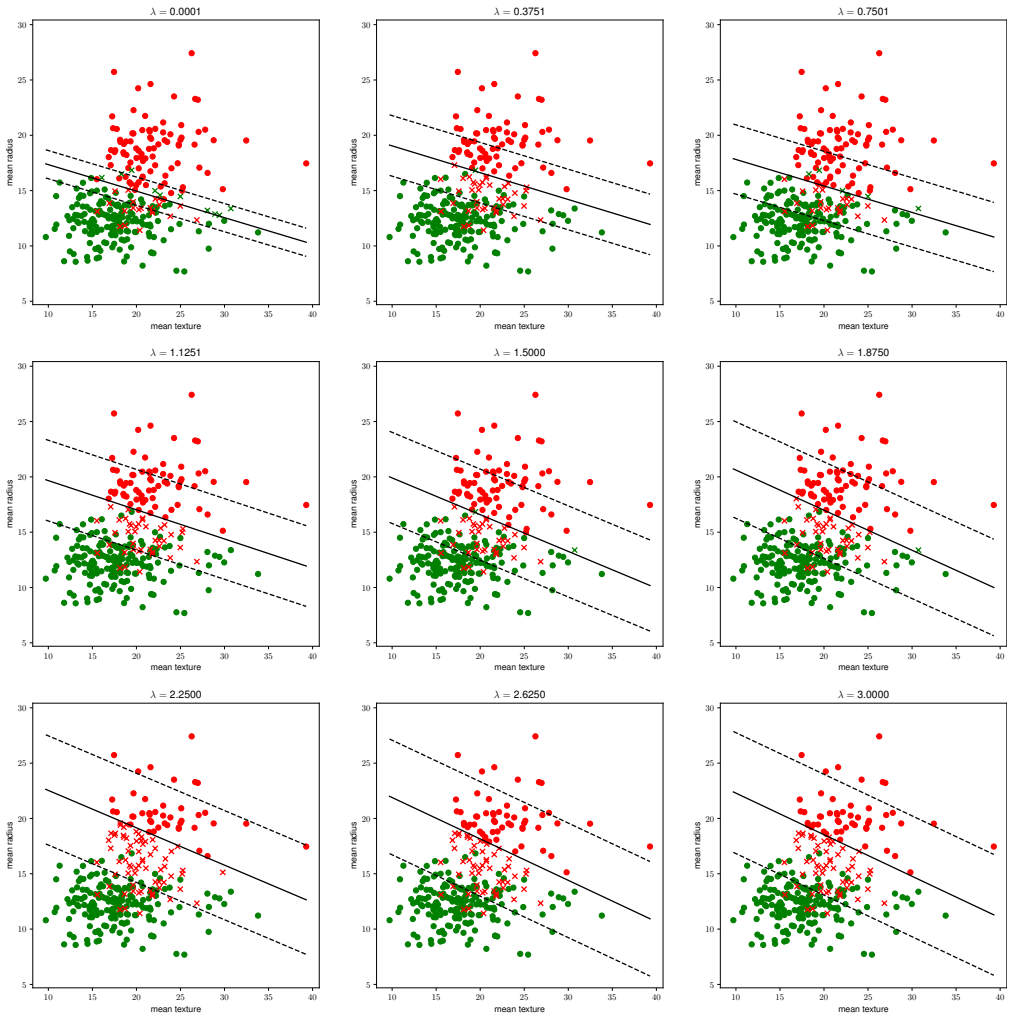
- Als λ klein - en eventueel zelfs 0 - is, komt het berekenen van een minimale kostfunctie J neer op het op nul zetten van zo veel mogelijke *hinge losses* van zo veel mogelijk punten. Dit resulteert in een kleine marge.
- Als λ groot is, ligt er meer nadruk op het minimaliseren van de grootte van $\|\vec{w}\|$ en dan spelen de *hinge losses* een minder grote rol. Hoe kleiner $\|\vec{w}\|$, hoe groter de marge, want de marge is $\frac{2}{\|\vec{w}\|}$ breed. We concluderen dat λ resulteert in een grotere marge, zoals te zien is in figuur 3.1.

3.2 Het belang van de regularisatieparameter

Een metaparameter in machine learning - bij ons dus de regularisatieparameter λ - regelt zowel de correctheid als de flexibiliteit van het model. Die twee eigenschappen noemen we respectievelijk *bias* en variantie.

3.2.1 Bias

In machine learning verwijst *bias* naar de systematische fout die door een model wordt geïntroduceerd wanneer het de onderliggende patronen in de trainingsgegevens consequent verkeerd weergeeft. Dit kan leiden tot onnauwkeurige voorspellingen en onbetrouwbare prestaties op nieuwe datasets.



Figuur 3.1: De invloed van de metaparameter λ op het SVM-model. Een grote λ resulteert in een grote marge, een kleine λ resulteert in een kleine marge.

Als een classificatiemodel wordt getraind op vertekende gegevens die overwegend één klasse vertegenwoordigen, kan het model leren nauwkeuriger te zijn voor die groep, terwijl het slecht presteert voor andere groepen. Als we ons SVM model bijvoorbeeld bijna uitsluitend trainen op gevallen waar de tumor kwaadaardig is, zal het hoogstwaarschijnlijk niet goed presteren voor nieuwe datapunten waar de tumor goedaardig is. Het model is namelijk niet getraind op tumoren van die aard en zal dus geen nauwkeurige predicties kunnen doen. De *bias* is in dit geval zeer hoog.

Het kan echter ook gebeuren dat de trainingsgegevens niet representatief zijn voor de populatie waarnaar ze willen generaliseren. Dit probleem treedt op wanneer bepaalde groepen in de populatie onder- of oververtegenwoordigd zijn in het trainingspakket. Als we ons SVM model bijvoorbeeld enkel trainen op tumoren van vrouwelijke borstkankerpatiënten, is de kans dat het model slecht presteert voor een nieuwe dataset van mannelijke patiënten vrij groot. We trainen ons model dan niet per se voornamelijk op één classificatiegroep, zoals in het vorige voorbeeld, maar de trainingsdata is eveneens niet representatief.

3.2.2 Variantie

De variantie is de maat voor de gevoeligheid van het model. Het geeft inzicht over de flexibiliteit van een model, met name hoe nauwkeurig de voorspellingen zijn bij verschillende datasets.

Bij een hoge variantie is het model sterk aangepast aan de trainingsdata. Dit betekent dat zelfs kleine veranderingen in de trainingsdata een grote invloed zullen hebben op het model, wat resulteert in een fenomeen genaamd *overfitting*. In het geval van tumorclassificatie zou een hoge variantie betekenen dat het model zeer nauwkeurig kan voorspellen of een tumor goedaardig is bij de trainingsdata. Maar doordat het model te sterk is afgestemd op de trainingsdata, zal het moeite hebben met generaliseren en levert dit geen precieze voorspellingen op voor de andere datasets, met name de validatie- en testdatasets.

Aan de andere kant kan een model ook een te lage variantie hebben. Dit betekent dat het model te veel generaliseert en weinig verandert indien de trainingsdata verandert. Een te lage variantie leidt tot *underfitting*. Hierdoor levert het ook geen betrouwbare voorspellingen op. Toegepast op tumorclassificatie zal een te lage variantie leiden tot een te sterk gegeneraliseerd model, waarbij het model niet goed presteert op de drie verschillende datasets.

3.3 *Bias-variance trade-off*

We willen enerzijds de *bias* minimaliseren door te fouten die ons model maakt te beperken. Dit kunnen we doen door ons model flexibeler te maken. Een grotere flexibiliteit resulteert onvermijdelijk echter in een grotere variantie. Een grote variantie resulteert dan in *overftting*, wat we natuurlijk ook willen vermijden, aangezien ons model anders geen accurate voorspellingen zal doen op nieuwe data. Het vinden van een evenwicht tussen de *bias* en de variantie is dus cruciaal voor het trainen van ons model. We noemen dit probleem vaak ook de *bias-variance trade-off*. Aangezien zowel de *bias* als de variantie worden beïnvloed door de regularisatieparameter, zullen we de beste waarde voor λ zoeken.

3.4 *Cross-validation*

3.4.1 De data opsplitsen

Voor het vinden van de optimale waarde λ_{opt} voor de regularisatieparameter, maken we gebruik van een principe genaamd *cross-validation*. Hiervoor splitsen we eerst de data die we voor handen hebben op in 3 delen: 50% trainingsdata, 25% validatiedata, 25% testdata:

Trainingsdata : 285 tumoren

Validatiedata : 142 tumoren

Testdata : 142 tumoren

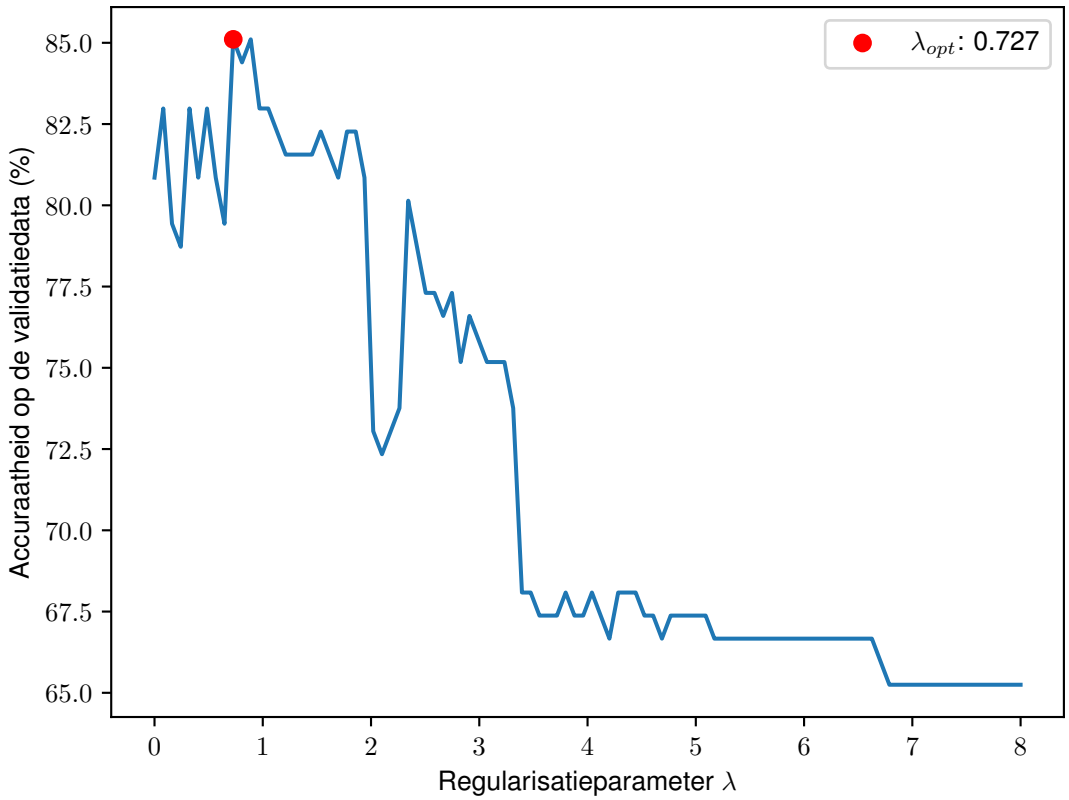
Voor *cross-validation* zullen we enkel gebruik maken van de eerste twee datasets. De testdata komt helemaal op het einde van dit verslag aan bod, wanneer we de accuraatheid van ons model zullen evalueren op nieuwe data.

3.4.2 Itereren over verschillende waardes voor λ

Na deze opsplitsing kunnen we ons model trainen op de validatiedata, telkens met een verschillende waarde voor λ. Bij elke iteratie zullen we de accuraatheid van het model voor deze specifieke waarde van de regularisatieparameter evalueren aan de hand van de trainingsdata. We herhalen onderstaand proces voor 100 verschillende waarden van 0.0001 tot 8 voor λ:

- Train het model op basis van onze trainingsdata voor een bepaalde waarde van λ. Deze trainingsdata bestaat uit 285 datapunten (met elk 30 *features*) en 285 *outputs*.
- Maak een voorspelling met dit model op basis van de *inputs* van de validatiedataset.
- Vergelijk de voorspellingen voor de y-waarden van de validatiedata met de effectieve y-waarden van de validatiedataset en bereken op basis daarvan de accuraatheid van het model voor de huidige waarde van λ. Hiervoor delen we het aantal goed geclassificeerde punten in de validatiedata door het aantal verkeerdelijk geclassificeerde punten in de validatiedata. Door dit getal te vermenigvuldigen met 100, krijgen we een percentage dat dan de accuraatheid van het model beschrijft.

We kunnen de accuraatheid van het getrainde model de trainingsdata uitzetten in functie van een metaparameter met behulp van een grafiek, zoals te zien is in figuur 3.2. We kiezen de λ die ons de hoogste accuraatheid oplevert: $\lambda_{\text{opt}} = 0.727$. Met deze waarde voor de regularisatieparameter was de accuraatheid van het model tegenover de validatiedata meer dan 85%.



Figuur 3.2: De accuraatheid van het model t.o.v. de validatiedata, uitgezet in functie van de metaparameter λ voor ons SVM-model.

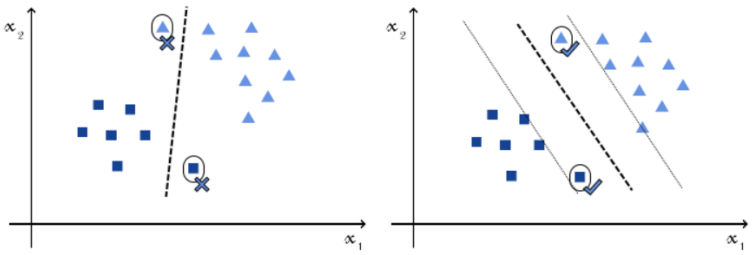
Hoofdstuk 4

Vergelijking met andere classificatietechnieken

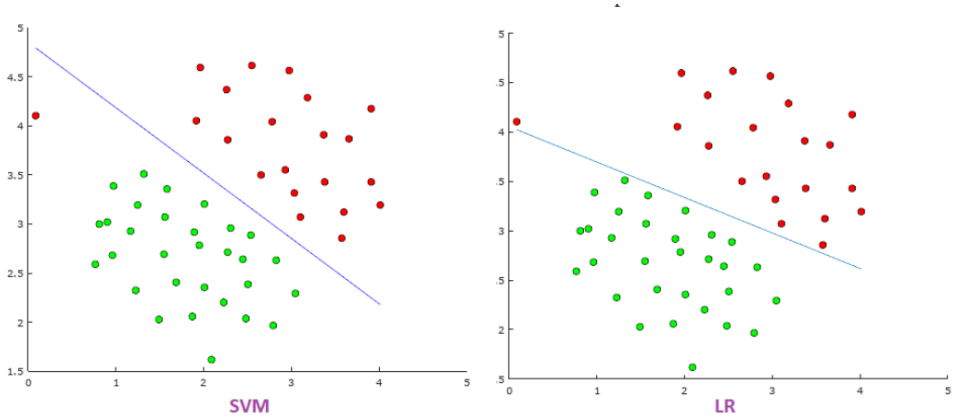
Waar Support Vector Machines in uitblinken, is het verwerken van datasets waarvan de datapunten elk zeer veel *features* hebben. SVM werkt namelijk uitermate goed in hoger dimensionale ruimtes, die we helaas dus niet visueel kunnen illustreren. Zo zouden we de tumoren met zeer hoge accuraatheid kunnen classificeren indien we de volledige lijst kenmerken in acht nemen.

Daarnaast is SVM minder gevoelig voor *overfitting* in vergelijking met complexere modellen zoals neurale netwerken. Dit is vooral belangrijk wanneer de data die voorhanden is, eerder beperkt is in grootte. SVM zal dus voor kleinere datasets heel goed presteren, met name als het aantal *features* groter is dan het aantal datapunten. Ook zal SVM in vergelijking met neurale netwerken bij het toevoegen van extra punten minder snel de mist in gaan zoals op figuur 4.1. Dit is waarom de hyperplane zo belangrijk is bij SVM.

Indien er heel veel variabiliteit is in de dataset, zal SVM ook een voordeel hebben tegenover andere classificatietechnieken: juist door het feit dat de marge tussen de scheidingshypervlakken zo groot mogelijk wordt gehouden, zullen *outliers* of andere vormen van ruis in de dataset niet in acht worden genomen. SVM is dus totaal niet gevoelig voor *overfitting*. Dit wordt nog eens extra benadrukt aan de hand van figuur 4.2.



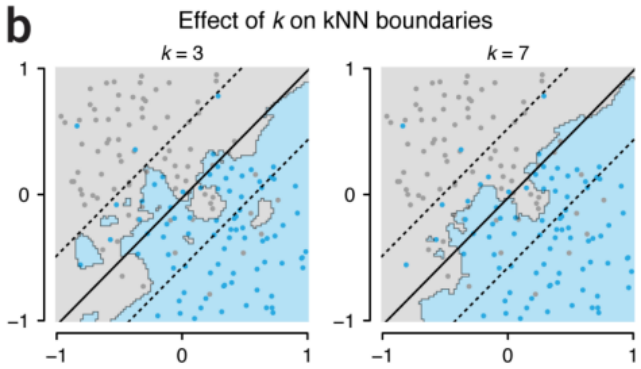
Figuur 4.1: Deze figuur uit bron [3] geeft een belangrijk verschil tussen een neuraal netwerk en een SVM model weer. Links wordt de werking van een neuraal netwerk geïllustreerd, rechts die van een SVM model. Hierbij valt op dat de kans dat een nieuw datapunt juist geclassificeerd zal worden, veel groter is bij SVM. Deze figuur benadrukt dus ook belang van het *maximum margin* hypervlak bij het voorkomen van *overfitting*.



Figuur 4.2: Deze figuur uit bron [4] toont de vergelijking tussen SVM en logistische regressie. Hierbij valt het duidelijk op dat wanneer er sprake is van een *outlier*, logistische regressie zich heel hard aanpast aan deze *outlier*. Dit toont dan aan dat Support Vector Machines veel beter bestendig zijn tegen ruis in de dataset.

Wanneer SVM bij de classificatie van punten van twee *features* geplot wordt, is deze visuele voorstelling ook eenvoudig te begrijpen. Het is namelijk zeer duidelijk of een punt nu tot de ene of andere klasse behoort volgens het model, door naar de eenvoudige scheidingslijn te kijken. Deze interpretatie is moeilijker bij andere classificatietechnieken zoals KNN-classificatie. Deze classificatietechniek berekent een scheidingslijn door naar de K dichtstbijzijnde buren van elk datapunt te kijken, wat vaak resulteert in een ietwat chaotische en minder makkelijk te begrijpen figuur, zoals je kan zien in figuur 4.3.

Tot slot is SVM zeer geheugenefficiënt: het model houdt enkel rekening met de dichtste punten tot de hypervlakken en zal dus, in tegenstelling tot bijvoorbeeld logistische regressie, niet met elk punt rekening moeten houden in de berekening van de beste beslissingsgrens.



Figuur 4.3: Deze figuur uit bron [5] vergelijkt KNN-classificatie met SVM-classificatie. Bij de eerste van de twee technieken ontstaan er twee twee gebieden, die worden bepaald door naar de k dichtste buren te kijken van elk datapunt. Op de linkerfiguur is $k = 3$, op de rechterfiguur is $k = 7$. Met behulp van *cross-validation* werd bepaald dat $k = 7$ de optimale waarde is voor de metaparameter k, met een accuraatheid van 87%. De rechte die door de grafiek gaat, is de beslissingsgrens van SVM en de stippellijnen zijn hypervlakken. Hierbij valt het duidelijk op dat SVM veel makkelijker te interpreteren is in vergelijking met KNN, terwijl de accuraatheid van SVM nog steeds gelijk is aan 85% in dit voorbeeld. We boeten, door gebruik te maken van SVM, dus een slechts een klein beetje in op accuraatheid, maar dit is in het voordeel van de interpreteerbaarheid van het model.

Hoofdstuk 5

Resultaten op onze dataset

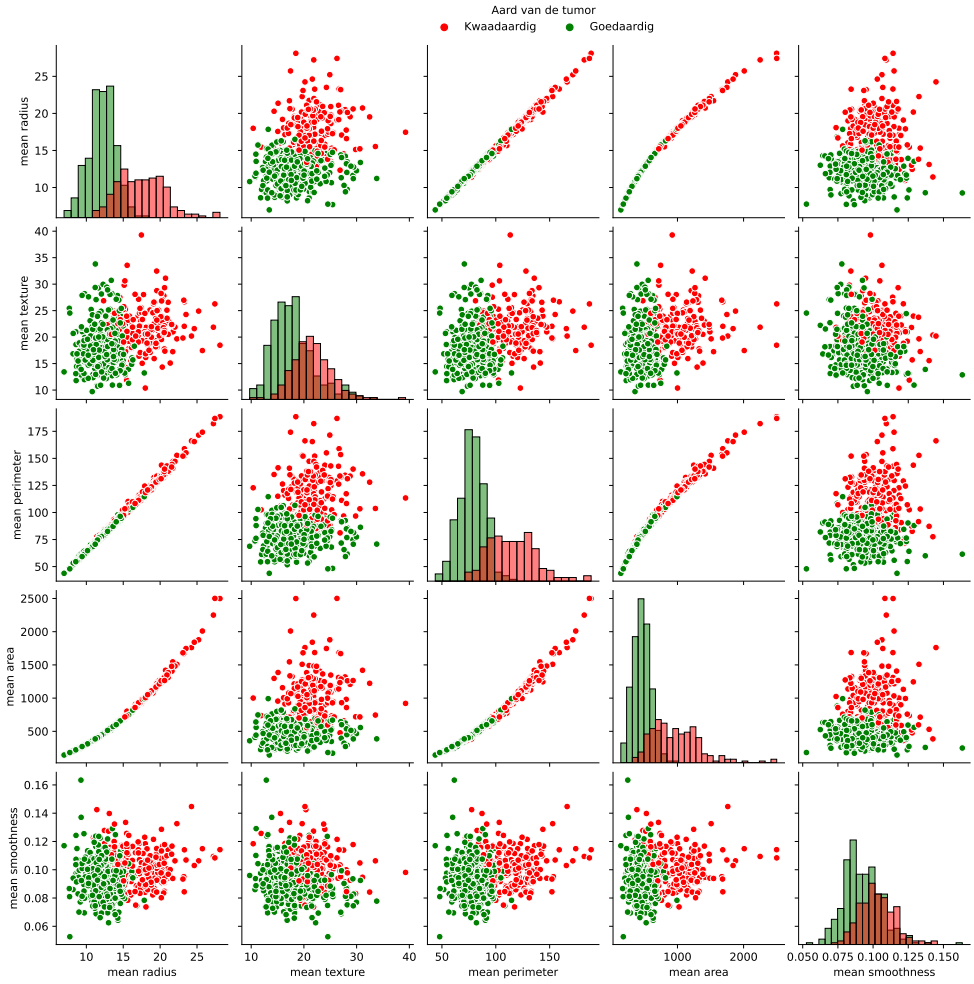
5.1 *Features*

Zoals in de inleiding reeds vermeld werd, bestaat onze dataset uit een 500-tal datapunten met elk 30 *features*. Hieronder lijsten we de eerste *features* op om de lezer een idee te geven van de data die voor handen is:

- Straal
- Textuur (standaardafwijking van grijswaarden)
- Omtrek
- Oppervlakte
- Gladheid (lokale variatie in straallengtes)
- Compactheid ($omtrek^2/oppervlakte - 1.0$)
- Concaafheid (ernst van concave delen van de contour)
- Concave punten (aantal concave delen van de contour)
- Symmetrie
- ...

5.1.1 Limitaties

Zoals eerder reeds vermeld werd, botsen we tegen een limitatie wat betreft het aantal *features* die we kunnen gebruiken voor ons model. Indien we het aantal *features* opdrijven, worden de wiskundige berekeningen namelijk een pak complexer. Daarnaast wordt het visueel voorstellen van de dataset ook des te moeilijker bij een toenemend aantal *features*. Wegens deze twee limitaties, zullen we ons dus beperken tot slechts twee *features* per tumor.



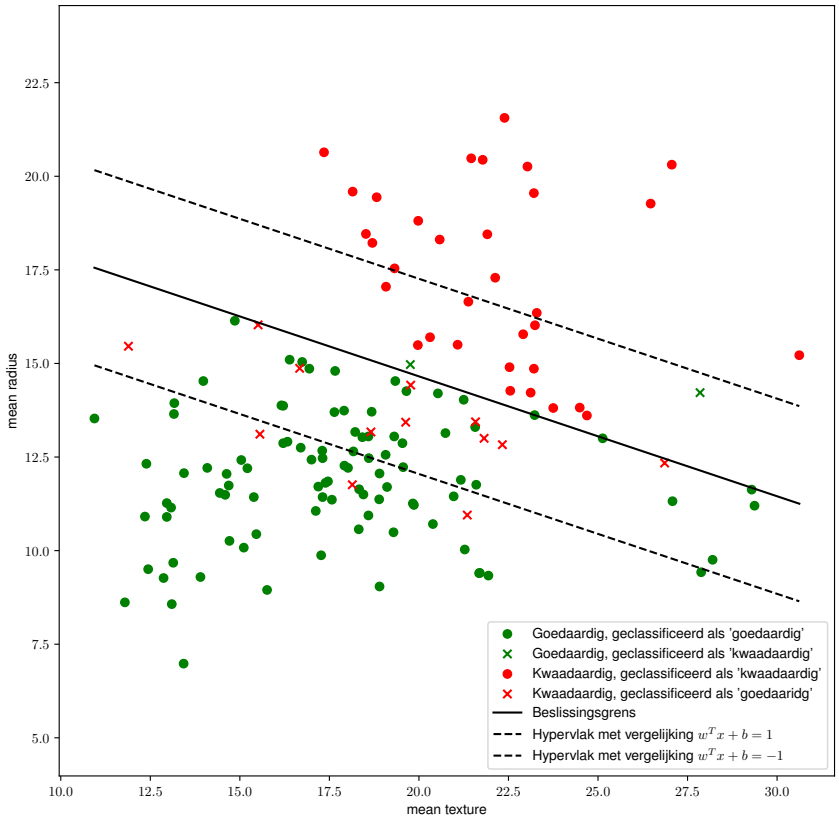
Figuur 5.1: Een *pairplot* van de eerste 5 *features* in de tumordataset.

5.1.2 Selectie

Hoe selecteren we nu de meest interessante *features*? We kunnen daarvoor gebruik maken van een *pairplot*, zoals er een te zien is in figuur 5.1. Een *pairplot* is een visuele weergave van de relatie tussen paren van *features* in een dataset. In machine learning worden dit soort voorstellingen vaak gebruikt om patronen en verbanden tussen de verschillende *features* van een dataset te identificeren. Op basis van deze pairplot bepaalden wij dat *mean texture* (textuur) en *mean radius* (straal) twee interessante *features* waren om te bestuderen.

5.2 Evaluatie van de accuraatheid

We trainden ons model op onze trainingsdata, beperkt tot de twee *features* die we hierboven kozen. Via *cross-validation* bepaalden we de optimale waarde voor de regularisatieparameter λ. De laatste stap in het proces is om met ons model een voorspelling te maken voor de testdata.



Figuur 5.2: Ons SVM model toegepast op de testdata. We konden een accuraatheid van $89,51\%$ bewerkstelligen.

Op die manier krijgen we een goed beeld van de effectieve accuraatheid van het model indien het data te zien krijgt waarop het niet getraind is. Zoals te zien is in figuur 5.2, was ons model in staat om voorspellingen te doen met een accuraatheid van maar liefst $89,51\%$ op onze testdata.

Besluit

Uit ons onderzoek kunnen we concluderen dan Support Vector Machines een classificatietechniek is die heel wat voordelen heeft dankzij de unieke eigenschappen van het model. De punten worden namelijk gescheiden met behulp van *maximum margin* hypervlakken. Dit wil zeggen dat de marge tussen de hypervlakken zo groot mogelijk gemaakt wordt, wat belangrijk is voor het minimaliseren van foute voorspellingen. Daarnaast viel het ons op dat wanneer er een *outlier* aan onze data wordt toegevoegd, er weinig invloed is op het definitieve model. SVM is dus zeer bestendig tegen *overfitting* en kan daarom ook gebruikt worden voor kleinere datasets.

Tijdens het trainen van het model wordt de *hinge loss* berekend als foutterm, waarbij de focus ligt op het minimaliseren hiervan. Zo wordt een nauwkeurige classificatie van punten buiten de hypervlakken verzekerd. Omdat we deze fout willen minimaliseren, maar toch een bruikbaar model willen overhouden, ontstaat er de kostfunctie J. Deze functie maakt een balans tussen de foutterm en de breedte van marge tussen de hypervlakken. Hierbij wordt een regularisatieparameter ingevoerd. Deze wordt vervolgens geoptimaliseerd om een zo hoog mogelijke accuraatheid te verkrijgen en toch nog een leesbaar model over te houden.

Na ons model in het algemeen berekend te hebben, pasten we deze toe op een concrete dataset. Met deze dataset wilden we bepalen op basis van enkele *features* of een tumor goedaardig of kwaadaardig is. Uit deze *features* kozen we er twee die werden toegepast op ons SVM-model. Hierbij verkregen we een accuraatheid van $89,52\%$, waaruit we konden besluiten dat SVM de beste manier is om een tumor te classificeren als goedaardig of kwaadaardig.

Bibliografie

[1] William Nick Street, William H. Wolberg, and Olvi L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Electronic imaging*, 1993.

[2] Mangasarian Olvi Street Nick Wolberg, William and W. Street. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1995. DOI: https://doi.org/10.24432/C5DW2B.

[3] Gabriele De Luca. Svm vs neural network, Nov 2022.

[4] Klevis Ramo. Spam emails with support vector machines, Dec 2017.

[5] Danilo Bzdok, Martin Krzywinski, and Naomi Altman. Machine learning: supervised methods. *Nature methods*, 15(1):5, 2018.

[6] Wikipedia contributors. Support vector machine — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Support_vector_machine&oldid=1183475870`, 2023. [Online; accessed 24-November-2023].

[7] Tarlan Ahadli. Linear svm classifier: Step-by-step theoretical explanation with python implementation. https://medium.com/, Gepubliceerd op 24 aug, 2021. Bezocht op 20 november 2023.

KU Leuven Kulak
Groep Wetenschap & Technologie
Etienne Sabbelaan 53
8500 Kortrijk