KU LEUVEN kulak

Groep Wetenschap & Technologie Kulak

Bachelor fysica, informatica, wiskunde
ingenieurswetenschappen

X0B53A – Probleemoplossen en ontwerpen, deel 1
X0A22D – Informaticawerktuigen

Support Vector Machines als middel 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

De gegevens

Stel dat we een gegevensset hebben, waarbij elk punt een aantal inputs heeft en daarenboven tot een bepaalde klasse of groep hoort. Deze inputs kunnen bijvoorbeeld verschillende eigenschappen zijn van een tumor. De tumor is ofwel kwaadaardig, ofwel goedaardig. De aard van de tumor zal dan de klasse zijn waartoe een datapunt behoort.

Predictie

We willen nu, gegeven een nieuwe dataset, graag voorspellen of een tumor al dan niet kwaadaardig is. Hiervoor kunnen we een machine learning techniek gebruiken die zo'n voorspelling doet voor de classificatie van een tumor met bepaalde eigenschappen. Hiervoor bestaan verschillende technieken, maar indien de hoeveelheid eigenschappen van de tumoren - wat dus meer algemeen overeenkomt met de inputs van de datapunten, zal blijken dat Support Vector Machines hier een zeer geschikte techniek voor blijken te zijn.

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 zullen we het principe achter Support Vector Machines, de toepassing waarvoor we kozen, uitleggen. Hierbij zullen we de verschillende classificatietechnieken vergelijken en beargumenteren waarom voor de tumor-dataset SVM een geschikte keuze blijkt te zijn.

Hoofdstuk 1

Machine Learning

TODO: wat is machine learning, ... ? Baseer je op de presentatie van Stijn en de andere documenten die we gekregen hebben.

Hoofdstuk 2

Supervised Learning

2.1 Supervised vs Unsupervised

Machine Learning wordt meestal gesplitst onder twee categoriën. *Supervised Learning* en *Unsupervised Learning*. Het grootste verschil tussen de twee, is dat *Supervised Learning* in tegenstelling tot *Unsupervised Learning* data met labels gebruikt. Bij *Supervised Learning* bestaat data uit parameters en labels. Men traint een *Supervised Learning* techniek op dergelijke data waardoor men vervolgens zal kunnen de labels voorspellen voor nieuwe gegeven waarden voor de parameters. Bij *Unsupervised Learning* zal de data enkel uit parameters bestaan. Een *Unsupervised Learning* techniek zal zelf patronen/structuren vinden in de data en vervolgens zelf labels uitvinden en geven aan de tuples van data.

2.2 Vormen van Supervised Learning

Supervised Learning en *Unsupervised Learning* kan nog verder gesplitst worden in categoriën; de twee voornaamste vormen van *Supervised Learning* zijn regressie en classificatie. Bij regressie bestaan de labels altijd uit getallen, waarbij de labels oneindige verschillende waarden kunnen aannemen. Dit staat in tegenstelling met classificatie waar de labels enkel een telbaar aantal waarden kunnen aannemen, dit kunnen getallen zijn wat quantitatieve classificatie genoemd wordt of ze kunnen ook woorden zijn wat qualitatieve classificatie genoemd wordt. Classificatie zal later in wat meer detail uitgelegd worden. De verschillende vormen van *Unsupervised Learning* ligt buiten de scope van dit rapport.

Hoofdstuk 3

Classificatie

3.1 Wat is classificatie

Classificatie is een machine learning techniek die binnen het gebied van supervised learning valt. Bij deze methode worden gegevens in een dataset ingedeeld in verschillende klassen, het kan worden uitgevoerd op zowel gestructureerde als ongestructureerde gegevens. Het proces begint met het voorspellen van de klasse van bepaalde datapunten.. De klassen worden vaak aangeduid als target, label of categorie. Het ultieme doel van classificatie is om te bepalen in welke categorie de nieuwe gegevens zullen vallen.

Om het concept een beetje beter te kunnen begrijpen bekijken we een eenvoudige voorstelling van de toepassing in verband met tumoren. Hierbij willen we aan de hand van classificatie onderzoeken of een tumor goedaardig of kwaadaardig is. Dit is een voorbeeld van een binaire classificatie aangezien er slechts twee klassen kunnen zijn, namelijk goedaardig of kwaadaardig.

3.2 Terminologie

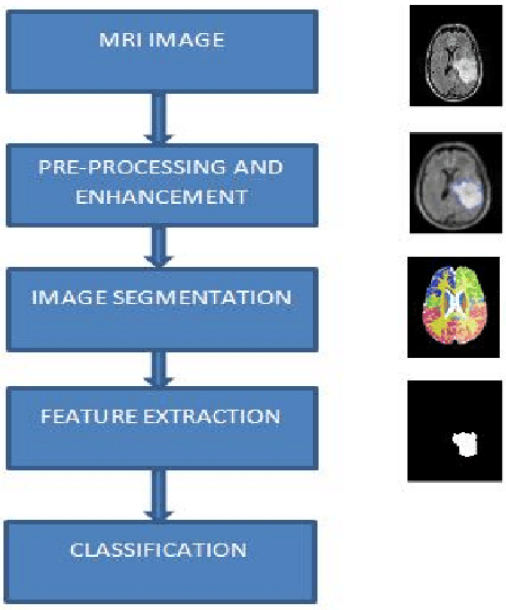
De **classifier** is een algoritme dat gebruikt wordt om de invoergegevens toe te wijzen aan een specifieke categorie. In ons geval heeft de classifier trainingsgegevens nodig om te begrijpen wat het verband is tussen de gegeven invoervariabelen en de klasse. Als de classifier nauwkeurig getraind is, kan deze worden gebruikt om na te gaan of de tumor goedaardig of kwaadaardig is.

Het **classificatiemodel** voorspelt de klasse of categorie van de gegevens aan de hand van de invoergegevens die tijdens de training zijn verstrekt.

Het **feature** of kenmerk is een meetbare eigenschap van iets dat je waarneemt. Bij tumor-classificatie kunnen bijvoorbeeld grootte, vorm, compositie en locatie van het beeld dienen als kenmerken.

Er bestaan ook classificaties met **meerdere klassen** of **meerdere labels**. Bij meerdere klassen wordt elk voorbeeld toegewezen aan één label, zoals bijvoorbeeld het identificeren van verschillende soorten dieren. Bij meerdere labels wordt elk voorbeeld toegewezen aan een reeks labels, zo kan een afbeelding meerdere objecten bevatten.

Overfitting treedt op wanneer een model te specifiek is voor de trainingsgegevens en slecht presteert op nieuwe, ongeziene gegevens. **Onderfitting** daarentegen treedt op wanneer het model te eenvoudig is en niet in staat is de complexiteit van de gegevens vast te leggen.



Figuur 3.1: Classificatie van een tumor. De eerste stap is de voorbewerking en verbetering van de MRI afbeelding. Fijnere details worden verbeterd en de ruis wordt uit het beeld gehaald. Bij image segmentation wordt het beeld opgedeeld in verschillende delen of segmenten om het beter te kunnen analyseren. Voor de features wordt rekening gehouden met bepaalde parameters, zoals grootte, vorm, compositie en locatie van het beeld. Volgens de resultaten die zijn verkregen uit de feature-extractie, wordt de classificatie van de tumor uitgevoerd.

3.3 Enkele toepassingen

- **Beeldherkenning:** Classificatie wordt vaak gebruikt in beeldherkenningstoepassingen, zoals het identificeren van objecten in foto's of video's.
- **Tekstclassificatie:** Hier wordt classificatie gebruikt om tekst te categoriseren, bijvoorbeeld het identificeren van spam-e-mails of het toewijzen van artikelen aan specifieke onderwerpen.
- **Medische diagnose:** Naast het eerder genoemde voorbeeld van tumorklassificatie, wordt classificatie ook toegepast op het diagnosticeren van verschillende medische aandoeningen op basis van verzamelde gegevens.

Hoofdstuk 4

Het model

4.1 Accuraatheid

Om de accuraatheid te bepalen van ons model splitsen we onze data in 3 delen namelijk: trainingsdata, validatiedata en testdata. Vaak wordt gesplitst in: 50% trainingsdata, 25% validatiedata, 25% testdata. Nadat het model is getraind met de trainingsdata, wordt het gevalideerd via de validatiedata en eventueel opnieuw getraind als de MSE te groot is. De testdata is voor de gebruiker om te zien of het model werkt voor nog nooit geziene data.

4.2 Bias

Bias kunnen we in woorden beschrijven als de gemiddelde fout op een voorspelde waarde. Bijvoorbeeld als de verwachte waarde 13 en de werkelijke waarde 65 zal de bias veel groter zijn dan wanneer de werkelijke waarde 14 is. Als we de bias wiskundig willen bereken doen we dit met de volgende formule

$$bias(\hat{\theta}) = E(\hat{\theta}) - \theta$$

waarbij θ gelijk is aan de variantie en$E(\hat{\theta})$ gelijk is aan de verwachte waarde van de schatter.

Stel dat de bias nul is dan zal $\theta = E(\hat{\theta})$. Dan noemen we dit een onvertekende schatter. Een lage bias is niet altijd beter. Een lage bias kan leiden tot overfitting waarbij het model te veel is aangepast aan de trainingsdata en er dus slechte voorspellingen worden gedaan voor de validatiedata en de testdata. De bias kan ook worden gebruikt in de formule voor de Mean-Squared Error (MSE) die bepaalt hoe algemeen sterk het model is. De term bias kan ook slaan op bepaalde stigma's die het model hanteert door niet representatieve testdata, volgens het principe garbage in, garbage out.

4.3 Variantie

Variantie in het kader van machine learning is de maat voor de gevoeligheid van een model. Het geeft inzicht over de flexibiliteit van het model, met name hoe goed het model zich kan aanpassen aan verschillende datasets. De variantie analyseert het verschil tussen de door het model voorspelde waarde en de werkelijke waarde van een variabele. Voor het berekenen van de variantie wordt het verschil tussen de werkelijke waarde en de voorspelde waarde genomen en dit

vervolgens gekwadrateerd. Neem een variabele x, dan is $f(x)$ de waarde horende bij de variabele x en $f(\hat{x})$ de voorspelde waarde voor x. Dan kan de variantie als volgt worden gedefinieerd:

$$Var(x) = (E(f(x) - f(\hat{x}))^2$$

Om een goed model te verkrijgen is het wenselijk om de variantie zo laag mogelijk te nemen. Een lage variantie betekent dat het model weinig afhankelijk is van veranderingen in de trainingsset en minder gevoelig is voor uitzonderlijke waarden binnen de set. Hierdoor zullen er accuratere voorspellingen gedaan worden bij andere datasets.

Bij een hoge variantie past het model zich nauwkeurig aan de trainingsset aan, maar zal het model de eventuele uitschieters ook als belangrijk beschouwen. Wanneer de trainingsset verandert, zal het model dus ook duidelijke aanpassingen vertonen. Bij het gebruik van een andere dataset zal er dan een duidelijker verschil zijn tussen de werkelijke en de voorspelde waarden. Een toename van de variantie zal leiden tot een verminderde nauwkeurigheid van het model.

4.4 Optimale metaparameters bepalen via cross-validation

4.4.1 Metaparameters

We hebben nu dus twee metaparameters: de variantie en de bias, die bepalen foe flexibel onze grafiek is en hoe dicht ze bij de punten wil aansluiten.

Met deze metaparameters kunnen de verwachte validatie MSE schrijven in functie van de variantie van de voorspelde waarde van x_0, de bias van de voorspelde waarde van x_0, en de variantie van de foutterm ϵ. In *An Introduction to Statistical Learning* [1] vinden we de formule

$$E(y_0 - \hat{f}(x_0))^2 = Var(\hat{f}(x_0) + [Bias(\hat{f}(x_0)]^2 + Var(\epsilon)$$

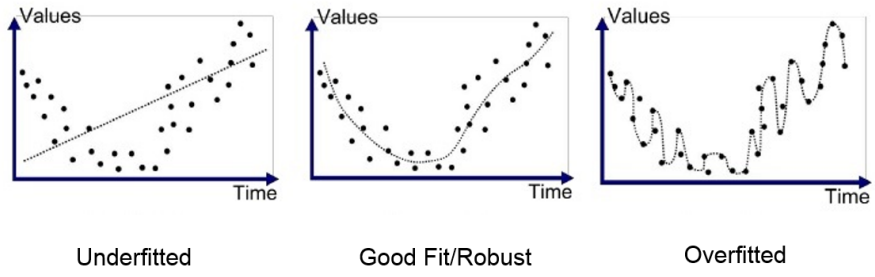
hiervoor terug.

4.4.2 Cross-validation

We proberen nu om de best passende metaparameters te kiezen zodat ons model een zo goed mogelijke voorspelling kan doen. Om deze parameters te kunnen bepalen gebruiken we cross-validation, hierbij delen we onze dataset op in een trainingsdata en een validatiedata. Onze trainingsdata wordt gebruikt om ons model op te stellen en de validatie data wordt hierna op ons model toegepast om de optimate metaparameters te vinden.

Het is belangrijk dat de validatiedata niet gebruikt wordt om ons model te trainen, we proberen namelijk met onze validatiedata ervoor te zorgen dat ons model algemeen genoeg is en niet enkel geschikt is om voorspellingen te doen op onze trainingsdata.

Als een model te veel aangepast is aan de trainingsdata noemen we dit overfitting (rechts op figuur 4.1), het model is dan te flexibel en probeert het model te dicht bij de trainingsdata te liggen. Het is belangrijk dat ons model algemeen is omdat we het model juist willen gebruiken om voorspellingen te doen op nieuwe data en niet op diezelfde trainingsdata, daarvan weten we namelijk al tot welke klasse ze behoren. Als ons model niet dicht genoeg bij de trainingsdata ligt noemen we dit underfitting (links op figuur 4.1), het model is dan niet flexibel genoeg en er zal ook geen goeie voorspelling gemaakt worden op de validatiedata. We proberen dus de optimale metaparameters te bepalen (midden op figuur 4.1).



Figuur 4.1: Illustratie van een underfitting, goeie fit en overfitting. Het eerste model is duidelijk geen goed model omdat het niet aansluit bij de datapunten. het laatste model zal bij deze dataset een zeer goed model zijn, maar ook enkel voor dit model. Het is niet algemeen genoeg om op nieuwe data toe te passen. We proberen dus het middelste model te vinden die goed bij de punten aansluit maar niet te flexibel is.[2]

Hoofdstuk 5

Support Vector Machines

5.1 Inleiding

Support Vector Machines is een classificatietechniek binnen het domein van supervised learning. We zullen, gegeven een dataset met grootte n voorspellen tot welke klasse een bepaald datapunt behoort. Zo'n datapunt heeft p verschillende features $(x_1, x_2, ..., x_p)$ en één y-waarde, zijnde de bijhorende klasse. We noemen SVM ook wel een *binaire classificeerder*, aangezien er maar twee mogelijke y-waardes zijn en elk punt maar tot een van twee mogelijke klassen kan behoren.

Uit wat later zal blijken, is het heel moeilijk om datapunten met meer dan 2 features te scheiden. Vanaf hogere dimensies is het ook heel moeilijk of zelfs onmogelijke om de werking van het model te visualiseren. Daarom beperken we ons in dit eindverslag tot een dataset waarbij elk punt slechts 2 features heeft. We zullen de dataset dan opsplitsen in 2 klasses.

5.2 Het scheidingsprincipe

Het SVM model zal trachten onze datapunten te scheiden in twee klasses. 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 een $(p-1)$-dimensioneel objecten, genaamd *hypervlakken*, berekend worden om de dataset in twee te 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, en het hypervlak ertussen noemen we de beslissingsgrens. We willen de marge zo groot mogelijk maken.

In het geval dat elk datapunt 3 features heeft, zal zo'n hypervlak een 2-dimensionaal vlak zijn. In ons geval heeft elk datapunt 2 features, dus zal de dataset kunnen gescheiden worden door een rechte, dat strikt genomen een 1-dimensioneel hypervlak is.

5.3 Waarom SVM en geen andere classificatietechniek?

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, i.h.b. als het aantal features groter is dan het aantal datapunten.

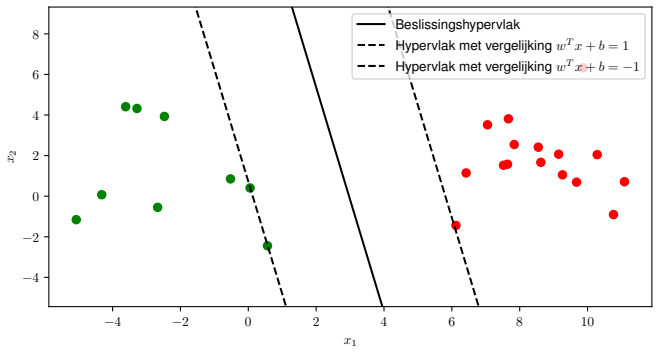
Indien er heel veel variabiliteit is in de dataset, zal SVM ook een *edge* hebben tegenover andere classificatietechnieken: juist door het feit dat de marge tussen de scheidingshyperplanes zo groot mogelijk wordt gehouden, zullen *outliers* of andere vormen van ruis in de dataset niet in acht worden genomen.

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

5.4 Wiskundige berekening van het model

5.4.1 De hypervlakken

We stellen een voorschrift op voor de twee scheidingsrechten, die te zien zijn in figuur 5.1, waartussen zich eem marge bevindt. Als $\vec{w}$ de normaalvector is op de hypervlakken, kunnen we de vergelijkingen van deze rechten schrijven als $w^T x - b = 1$ en $w^T x - b = -1$. Hierbij is b dan de *intercept* van de rechte die de beslissingsgrens beschrijft. 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.



Figuur 5.1: Twee lineair scheidbare wolken van punten. De kleuren van de punten duiden aan tot welke klasse ze behoren.

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 m.a.w. dus $\|\vec{w}\|$ trachten te minimaliseren.

5.4.2 Trainen van het model

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 nu de *hinge loss* in. Dit 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 belissingslijn.

   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.

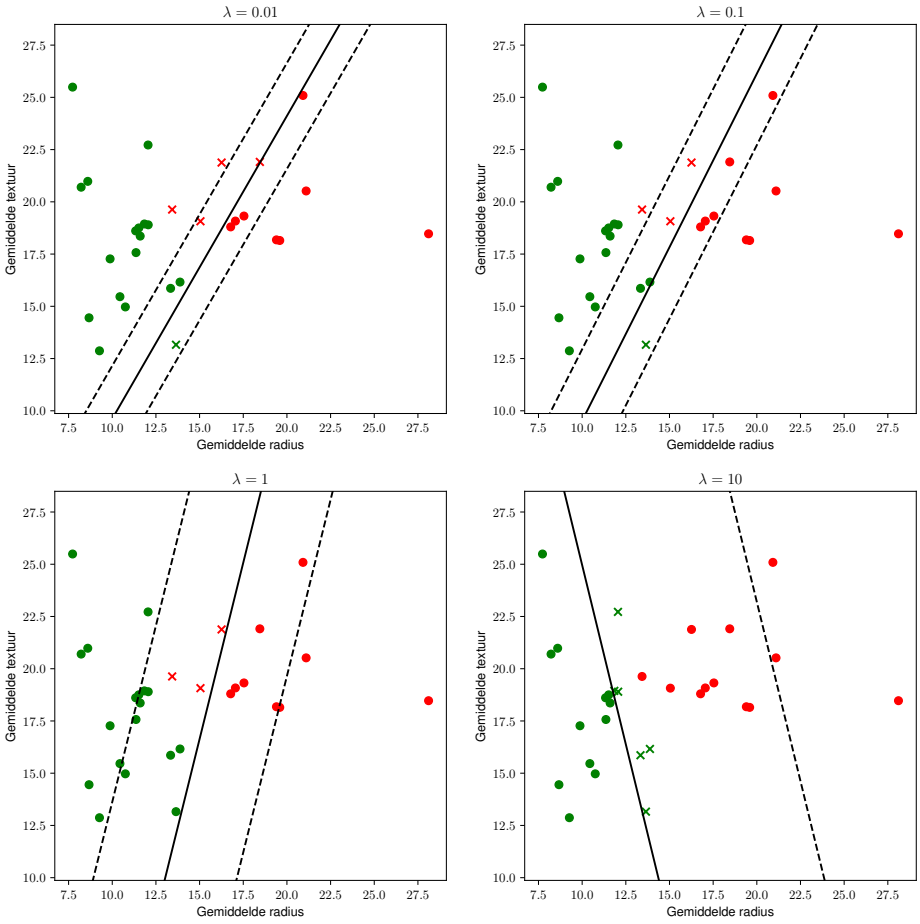
Om het SVM-model te trainen, moeten we nu dus met twee zaken rekening houden. Enerzijds willen we $\|\vec{w}\|$ minimaliseren om een zo groot mogelijke marge te bekomen anderzijds willen we de *hinge loss* of strafterm zo klein mogelijk houden. We voeren een kostfunctie J in en zoeken het minimum $\min_{w,b}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

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 5.2.



Figuur 5.2: De invloed van de metaparameter λop het SVM-model. Een grote λ resulteert in een grote marge, een kleine λ resulteert in een kleine marge.

Besluit

Bibliografie

[1] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. An introduction to statistical learning: With applications in python. *(No Title)*, 2023.

[2] Anup Bhande. What is underfitting and overfitting in machine learning and how to deal with it., Mar 2018.

[3] 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].

[4] 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