Hoofdstuk 1

Support Vector Machines

1.1 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 klassen.

1.2 Het scheidingsprincipe

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 een $(p-1)$-dimensioneel objecten, genaamd *hypervlakken*, berekend worden om de dataset in twee te verdelen.

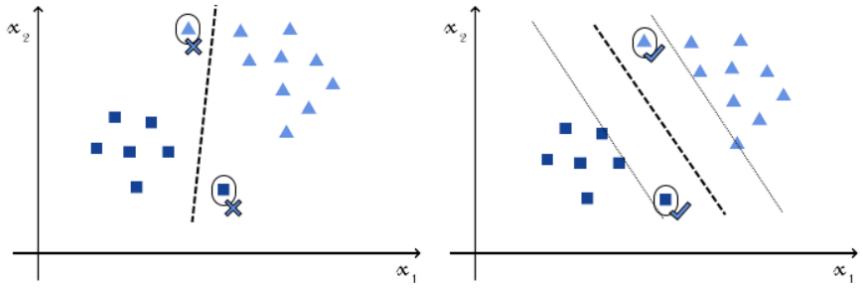
We construeren twee zo'n hypervlakken die de twee gegevensklassen scheiden, zodat de afstand daartussen zo groot mogelijk is. Het gebied dat wordt begrensd door deze twee hypervlakken wordt de *marge* genoemd, en het hypervlak ertussen noemen we de beslissingsgrens. We willen de marge zo groot mogelijk 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.

1.3 Waarom SVM en geen andere classificatietechniek?

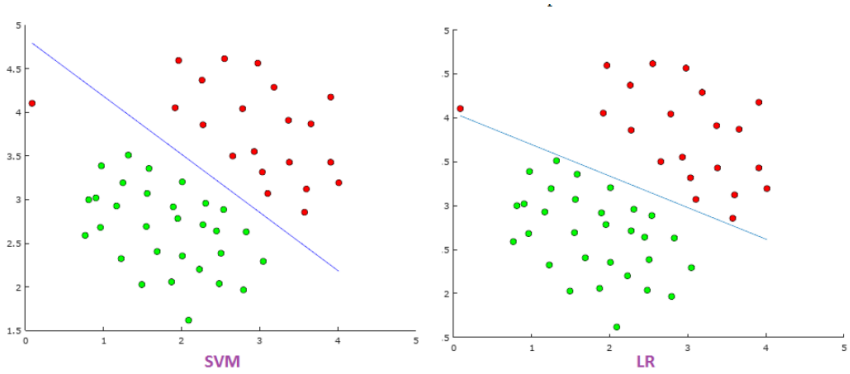
Waar Support Vector Machines in uitblinken, is het verwerken van datasets waarvan de datapunten elk zeer veel features hebben. SVM werkt namelijk uitermate goed in hoger dimensionale ruimtes, die we helaas dus niet visueel kunnen illustreren. Zo zouden we de tumoren met zeer hoge accuraatheid kunnen 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. Ook zal SVM in vergelijking met neurale netwerken bij het toevoegen van extra punten minder snel de mist in gaan zoals op figuur 1.1. Dit is waarom de hyperplane zo belangrijk is bij SVM.



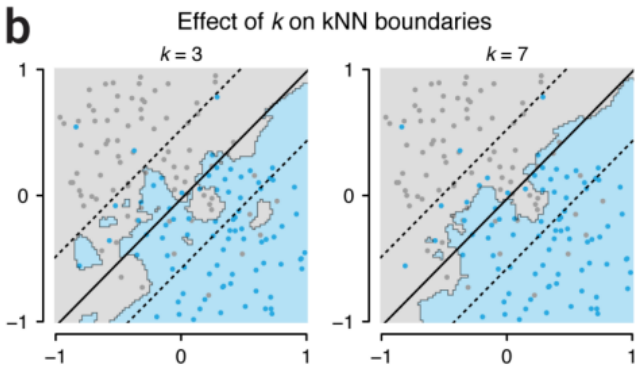
Figuur 1.1: Deze figuur uit bron [**?**] geeft een belangrijk verschil tussen neurale netwerken en SVM weer. Links wordt de werking van een neuraal netwerk geïllustreerd, rechts die van een SVM-model. Hierbij valt op het dat als je een nieuw punt zou classificeren, er bij neurale netwerken een veel grote kans was dat deze op het vorige model niet incorrect zou zijn, terwijl er bij SVM wel een veel grotere kans is dat het punt correct geclassificeerd zou worden. Deze figuur benadrukt dus ook het belang van de hyperplane bij SVM en zijn rol bij het voorkomen van *overfitting*.

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. SVM is dus totaal niet gevoelig voor overfitting. Dit wordt nog eens extra benadrukt aan de hand van figuur 1.2.



Figuur 1.2: Deze figuur uit bron [**?**] toont de vergelijking tussen SVM en logistische regressie. Hierbij valt het duidelijk op dat wanneer er sprake is van een *outlier*, logistische regressie de mist in gaat. Dit toont dan ook aan dat Support Vector Machines veel beter bestendig zijn tegen overfitting.

Wanneer SVM in twee dimensies kan geplot worden, is deze ook zeer eenvoudig te begrijpen.



Figuur 1.3: Deze figuur uit bron [**?**] vergelijkt KNN-classificatie met SVM-classificatie. Bij de eerste van de twee technieken, ontstaan er twee 2 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 $k = 7$. M.b.v. *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 de grenzen van de hyperplanes van SVM. Hierbij valt het duidelijk op dat SVM veel makkelijker te interpreteren is in vergelijking met KNN, terwijl de zekerheid van SVM nog steeds gelijk is aan 85%. We boeten, door gebruik te maken van SVM, dus een heel klein beetje in op accuraatheid, maar dit is serieus in het voordeel van de interpreteerbaarheid van het model.

Het is namelijk zeer duidelijk of een punt nu tot de ene of andere klasse behoort volgens het model door naar de scheidingslijn te kijken. Deze interpretatie is moeilijker bij andere classificatietechnieken zoals KNN-classificatie. Hierbij wordt er gekeken naar de K-nearest-neighbours. Door naar deze dichtste buren te kijken kan deze plot veel moeilijker begrepen worden. Er kunnen namelijk hierbij allemaal aparte wolken ontstaan van punten die tot een bepaalde klasse behoren waardoor de figuur er chaotisch zou kunnen uitzien. Dit kan je ook zien op figuur 1.3.

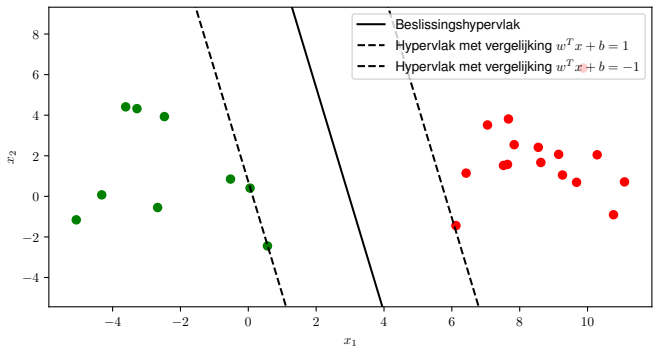
Tot slot is SVM zeer geheugenefficiënt: het model houdt enkel rekening met de dichtste punten tot de hyperplanes en zal dus, in tegenstelling tot bijvoorbeeld logistische regressie, niet met elk punt rekening moeten houden in de berekening van de beste beslissingsgrens.

1.4 Wiskundige berekening van het model

1.4.1 De hypervlakken

We stellen een voorschrift op voor de twee scheidingsrechten, die te zien zijn in figuur 1.4, 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.

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.



Figuur 1.4: Twee lineair scheidbare wolken van punten. De kleuren van de punten duiden aan tot welke klasse ze behoren.

1.4.2 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 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.

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

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

.

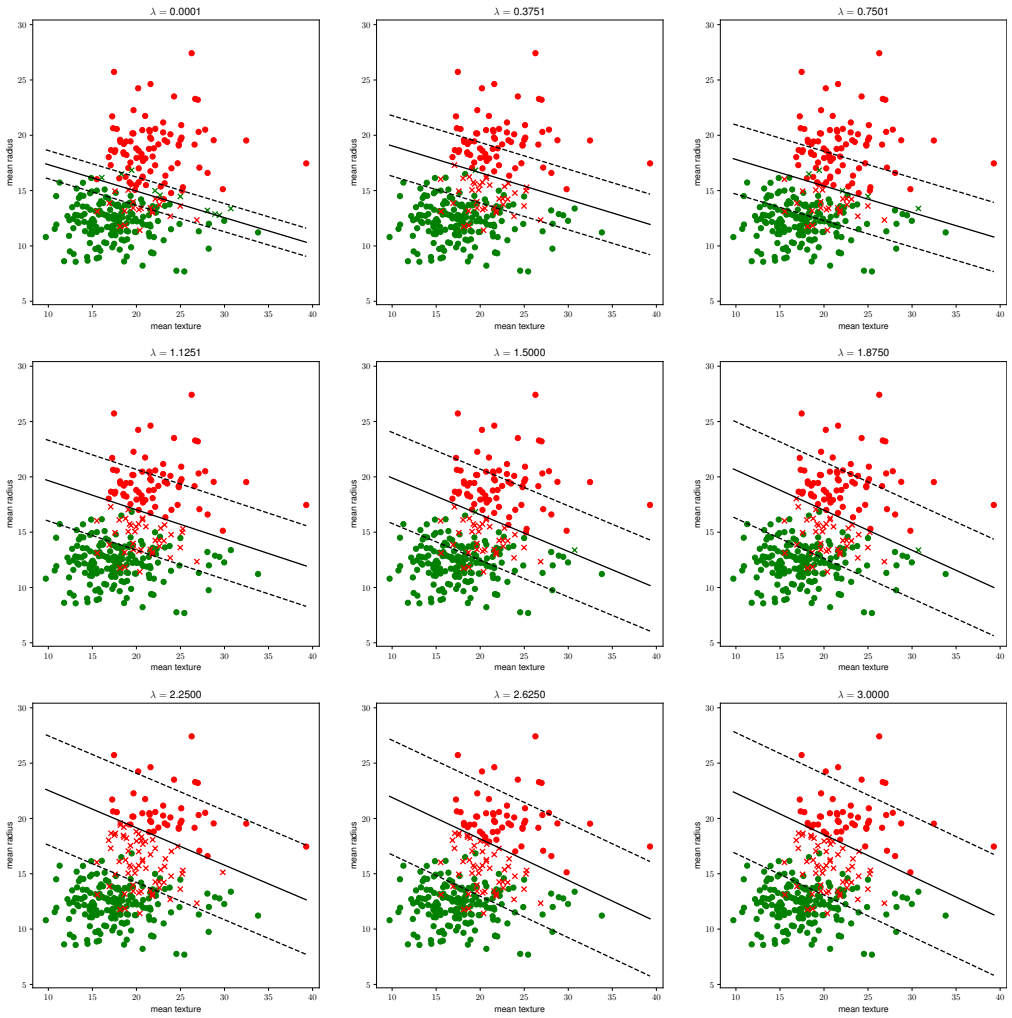
Hierbij is $\frac{1}{n}\sum_{i=1}^{n}\max\left[0, 1-y_i(\vec{w}\cdot\vec{x}_i-b)\right]$ de gemiddelde *hinge loss*, dus de gemiddelde foutterm die werd toegekend over alle n datapunten in de dataset. De tweede term $\lambda\cdot||\vec{w}||^2$ regelt het evenwicht tussen enerzijds het minimaliseren van 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 $\|\vec{w}\|$.

1.4.3 Regularisatieparameter

In het geval van SVM noemen we de metaparameter (zie deel **??**) 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 1.5.

Voor het vinden van de optimale waarde λ_{opt} voor de regularisatieparameter, verwijzen we graag terug naar deel **??**, waar via *cross-validation* de beste waarde voor de metaparameter wordt bepaald. Een illustratie van de invloed van λ op de accuraatheid van het model is te zien in figuur **??**.



Figuur 1.5: 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.