2. Generieke neuronen

In dit hoofdstuk zullen we het perceptron en lineaire regressie algoritme veralgemeniseren tot het model van een neuron dat een activatiefunctie en een lossfunctie kent. Door de loss te minimaliseren middels stochastic gradient descent blijken we de modellen uit het vorige hoofdstuk te kunnen reproduceren. Met andere keuzes van deze functies kunnen we bijvoorbeeld ook logistische regressie uitvoeren. Dit leidt uiteindelijk tot een zeer algemeen model dat de bouwsteen van neurale netwerken zal worden.

2.1. Lossfunctie

In het voorgaande hoofdstuk hebben we gezien dat het perceptron algoritme in staat is om lineair separabele trainingsdata foutloos te classificeren. Het deed dat door, beginnend met een bias $b = 0$ en gewichten $w_i = 0$, voor de instances een klasselabel $\hat{y} = \operatorname{sgn}(b + \sum_i w_i \cdot x_i)$ te voorspellen, en aan de hand daarvan de parameters bij te werken volgens de regels $b \leftarrow b - (\hat{y} - y)$ en $w_i \leftarrow w_i - (\hat{y} - y)\, x_i$, net zolang totdat alle instances juist worden voorspeld. Een soortgelijk model kon worden geformuleerd voor lineaire regressie, waarbij dan de signum-functie werd vervangen door de identiteitsfunctie en een learning rate α als extra coëfficiënt werd toegevoegd aan de updateregel. In dit hoofdstuk gaan we dezelfde modellen wat rigoreuzer afleiden, met als bonus dat we ook een aantal andere modellen (zoals logistische regressie) kunnen begrijpen en optimaliseren, waardoor we niet langer beperkt blijven tot lineair separabele datasets.

We beginnen met terug te kijken naar hoe regressiemodellen traditioneel worden geoptimaliseerd. Hierbij wordt gebruik gemaakt van de *kleinste-kwadraten-methode*, oftewel *least squares*. Een regressiemodel dat aan een instance met een uitkomst y een voorspelling toekent gelijk aan $\hat{y}$ maakt hierbij een fout $e = \hat{y} - y$. Omdat fouten naar boven en naar beneden even ernstig zijn worden deze fouten gewoonlijk gekwadrateerd om het teken kwijt te raken. We definiëren hiertoe een kwadratische *lossfunctie* $\mathcal{L}(\hat{y}; y) = (\hat{y} - y)^2$ die aangeeft hoe ernstig een gemaakte fout is.

De lossfunctie levert een loss $l = \mathcal{L}(\hat{y}; y) = 0$ als de voorspelde uitkomst exact overeenkomt met de gewenste uitkomst, en heeft altijd een strict positieve waarde $l > 0$ als hierin een afwijking optreedt. Kortom, we kunnen nu zeggen dat een oplossing beter is naarmate l lager is. Hiermee is het probleem omgezet in een optimalisatievraagstuk waarin we op zoek moeten naar het *minimum* van een functie. De parameters die we kunnen variëren zijn de bias b en de gewichten w_i.

We zullen voor het gemak de bias weer even tijdelijk beschouwen als een van de gewichten (w_0) die altijd gecombineerd wordt met een constant attribuut ($x_0 = 1$), zodat we niet speciaal rekening hoeven te houden met de bias als we de gewichten w_i teza-

men optimaliseren. Alles wat hieronder over de gewichten w_i wordt geschreven geldt dus soortgelijk ook voor de bias b.

De vraag is nu: hoe kunnen we $\boldsymbol{w}$ zodanig kiezen dat de uitkomst l zo klein mogelijk is?

De berekening van de loss l geschiedt eigenlijk in drie stappen:

1. Eerst worden de gewichten gecombineerd met de waarden van de attributen om de pre-activatiewaarde te berekenen:

$$a = \sum_i w_i \cdot x_i$$

   Kortom, a hangt rechtstreeks af van w_i, en als je wil weten hoe a varieert als w_i verandert dan hoef je alleen de waarden van de attributen x_i te kennen.

2. Daarna wordt de pre-activatiewaarde in de activatiefunctie gestopt om de post-activatiewaarde te berekenen:

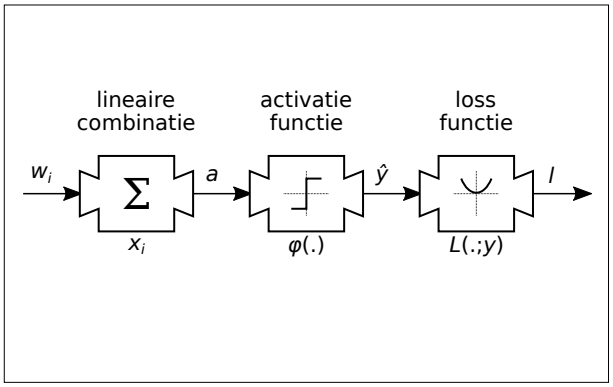
$$\hat{y} = \varphi(a)$$

   Kortom, $\hat{y}$ hangt rechtstreeks af van a, en als je wil weten hoe $\hat{y}$ varieert als je a verandert dan hoef je alleen de vorm van de activatiefunctie φ te kennen.

3. Tenslotte wordt de post-activatiewaarde in de lossfunctie gestopt om de loss te berekenen:

$$l = \mathcal{L}(\hat{y}; y)$$

   Kortom, l hangt rechtstreeks af van $\hat{y}$, en als je wil weten hoe l varieert als je $\hat{y}$ verandert dan hoef je alleen de vorm van de lossfunctie $\mathcal{L}$ te kennen (voor een gegeven gewenste uitkomst y voor het betreffende instance).



We kunnen dit proces visualiseren als een reeks van drie transformaties, zoals in de figuur hierboven. Links gaan de gewichten erin, die worden omgezet in de pre-activatiewaarde middels de attributen x_i, die worden omgezet in de post-activatiewaarde middels de activatiefunctie φ, die worden omgezet in de loss middels de lossfunctie $\mathcal{L}$. Dit machientje wordt ook wel een *neuron* genoemd, naar analogie met hersencellen.

Om de loss te minimaliseren dienen we te weten hoe l reageert als we de waarden van alle w_i één voor één veranderen. Alle gewichten zijn als het ware instelknoppen van het neuron waaraan we kunnen draaien. Hoe moeten we nu aan die knoppen draaien om de uitvoer (dat wil zeggen: de loss) omlaag te krijgen? Om dat te bepalen is het voldoende om te weten hoe de uitvoer van elk van de drie genoemde stappen afhangt van hun invoer. Preciezer geformuleerd, als we één van de gewichten verhogen van de huidige waarde w_i naar een nieuwe waarde $w_i + \Delta w_i$, dan zal de pre-activatiewaarde meeveranderen van a naar een zekere $a + \Delta a$, wat op zijn beurt leidt tot een verandering van de post-activatiewaarde van $\hat{y}$ naar $\hat{y} + \Delta\hat{y}$, hetgeen tenslotte leidt tot een verandering van de loss van l naar $l + \Delta l$. We gebruiken de notatie Δ om veranderingen aan te geven; dit kan gaan om een toename als $\Delta w_i > 0$ of ook om een afname als $\Delta w_i < 0$. We willen nu weten, als we de gewichten veranderen met een hoeveelheid Δw_i, hoe groot zijn dan achtereenvolgens de bijbehorende Δa, $\Delta\hat{y}$, en Δl? Het doel is daarbij om de loss l te verlagen, dus te bereiken dat $\Delta l < 0$.

Opgave 20. *

Leg uit waarom zelfs de meest optimale oplossing die bij lineaire regressie gevonden kan worden slechts zelden een loss $l = 0$ oplevert.

Opgave 21. ***

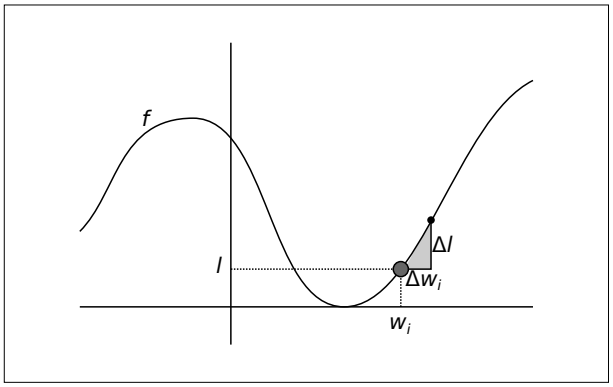
Welke van de onderstaande functies zou je in principe ook als lossfunctie kunnen gebruiken? Bedenk wat vereiste eigenschappen zijn voor een lossfunctie en leg uit wat er mis is met sommige van deze functies.

- $\mathcal{L}(\hat{y}; y) = \hat{y} - y$
- $\mathcal{L}(\hat{y}; y) = (\hat{y} - y)^2$
- $\mathcal{L}(\hat{y}; y) = |\hat{y} - y|$
- $\mathcal{L}(\hat{y}; y) = \hat{y}^2 - y^2$
- $\mathcal{L}(\hat{y}; y) = (\hat{y} + y)^2$

2.2. Gradient descent

Er zijn diverse methoden om minima van een functie te bepalen, maar een algemeen bruikbare methode is *gradient descent*. Deze methode kan ook toegepast worden op de afhankelijkheid in de loss van de gewichten. Gradient descent werkt door te beginnen met een willekeurige startwaarde voor w_i, en dan te kijken naar de helling van de functie die beschrijft hoe de loss afhangt van de gewichten. Als de helling ongelijk aan nul is dan kan de loss worden verkleind door een stap te zetten in de richting van de neergaande helling.

In het voorbeeld hieronder is een willekeurig verloop van de loss l geschetst met slechts één argument w_i. Noemen we deze functie even f, dan is dus $l = f(w_i)$; in de vorige paragraaf hebben we gezien dat deze functie f eigenlijk uit drie opeenvolgende transformaties bestaat die afhangen van x_i, φ en $\mathcal{L}$.



De helling van de functie f ter plekke van het aangegeven grijze punt kun je bepalen door een klein stapje opzij te zetten naar de zwarte stip. Laat je w_i toenemen tot $w_i+\Delta w_i$ dan correspondeert dat met een toename van l tot een zekere nieuwe waarde $l + \Delta l$. De helling van de functie ter plekke van dit punt komt overeen met de steilheid van het getekende grijze driehoekje als je zorgt dat de grootte van het stapje voldoende klein is. Deze steilheid kun je karakteriseren met de verhouding $\frac{\Delta l}{\Delta w_i}$, ook wel de *afgeleide* genoemd. Als de functie f *lineair* is en een rechte lijn beschrijft ken je deze helling vermoedelijk als de *richtingscoëfficient*.

Omdat enerzijds voor het grijze punt geldt dat $l = f(w_i)$ en anderzijds voor de zwarte stip geldt dat $l + \Delta l = f(w_i + \Delta w_i)$, kun je schrijven $\Delta l = f(w_i + \Delta w_i) - l = f(w_i + \Delta w_i) - f(w_i)$. Hieruit volgt voor de afgeleide:

$$\frac{\Delta l}{\Delta w_i} = \frac{f\left(w_i + \Delta w_i\right) - f\left(w_i\right)}{\Delta w_i}$$

Hetzelfde kan bereikt worden door een stapje in de ene richting én een stapje in de andere richting te maken en die twee punten met elkaar te vergelijken. Dat leidt tot dit resultaat:

$$\frac{\Delta l}{\Delta w_i} = \frac{f\left(w_i + \Delta w_i\right) - f\left(w_i - \Delta w_i\right)}{2\Delta w_i}$$

Als de stapjes maar voldoende klein zijn en f een gladde functie is dan is de uitkomst van beide uitdrukkingen gelijk.

Als de stapjes *infinitesimaal* klein gemaakt worden, wat wil zeggen willekeurig klein maar nog net niet nul, dan spreken we ook wel van een *differentiaal* $\mathrm{d}l$ of $\mathrm{d}w_i$; de notatie met de griekse letter Δ wordt dan vervangen door een d, en we vinden een afgeleide $\frac{\mathrm{d}l}{\mathrm{d}w_i}$. Als daarenboven de functie f afhangt van meerdere argumenten $w_0, w_1, w_2, \ldots, w_d$ en we alleen maar kijken naar de invloed van een verandering in één van de argumenten tegelijkertijd, dan spreken we van een *partiële* differentiaal ∂l of ∂w_i. We zullen vanaf nu deze laatste notatie gebruiken omdat die het meest algemeen is en ook in boeken en artikelen op het gebied van machine learning veel voorkomt. De helling van de lossfunctie wordt daarmee genoteerd als $\frac{\partial l}{\partial w_i}$. In gedachten mag je houden dat het bij al deze verschillende notaties steeds over de verhouding tussen kleine corresponderende verschillen gaat, zoals afgebeeld in de figuur hierboven.

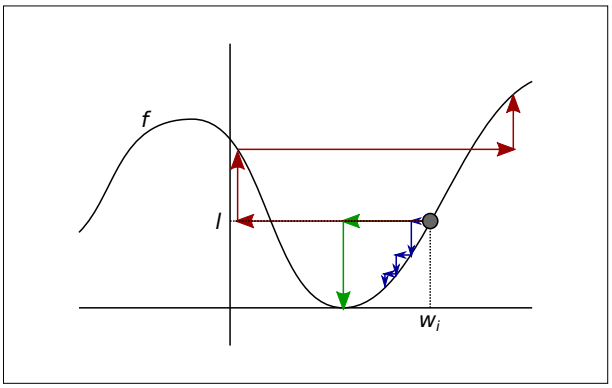
Keren we terug naar gradient descent, dan is het idee dat als de functie toeneemt, en dus de helling van de functie positief is, dat we dan de waarde van l kunnen laten afnemen door de waarde van w_i te verkleinen. Omgekeerd, als de functie afneemt, en de helling negatief is, dan moet w_i worden vergroot om in de richting van een minimum in l te bewegen. Hieruit kunnen we de updateregel $w_i \leftarrow w_i - \frac{\partial l}{\partial w_i}$ afleiden. Ga voor jezelf na dat de gewichten dan inderdaad worden verkleind als de helling $\frac{\partial l}{\partial w_i}$ positief is en worden vergroot als de helling negatief is.

Hoe groot de stap in het ideale geval moet zijn hangt af van de sterkte van de kromming van de functie. Deze kennen we niet. Daarom gebruiken we wederom de learning rate α als coëfficiënt om de stapgrootte mee in te kunnen stellen. We krijgen uiteindelijk:

$$w_i \leftarrow w_i - \alpha \cdot \frac{\partial l}{\partial w_i}$$

Soms wordt hiervoor ook wel de vectornotatie $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \cdot \boldsymbol{\nabla}_{\boldsymbol{w}} l$ gebruikt, waarbij dan de *gradiënt* een vector is gelijk aan $\boldsymbol{\nabla}_{\boldsymbol{w}} l = \left[\frac{\partial l}{\partial w_0}, \frac{\partial l}{\partial w_1}, \frac{\partial l}{\partial w_2}, \ldots, \frac{\partial l}{\partial w_d}\right]$. Hier komt de naam van de methode vandaan.

In de figuur hieronder is te zien hoe het bijwerken van de gewichten kan verlopen. Beginnen we weer met een willekeurige waarde van w_i, gemarkeerd met het grijze punt. De functie loopt hier naar boven, oftewel de loss neemt toe met het gewicht. Om een minimum op te sporen in het geval van een dergelijke positieve helling moeten we de waarde van het gewicht verkleinen. Met de gekleurde pijlen is aangegeven hoe dit in zijn werk kan gaan.



- Voor de groene pijlen is de learning rate α toevallig optimaal gekozen. De waarde van w_i wordt in één keer verlaagd tot in het "dal" van de functie. Hier loopt de functie horizontaal en is de helling gelijk aan nul, dus hierna hoeft het gewicht niet meer bijgewerkt te worden. Dit blijkt ook uit de gradient descent regel: als de afgeleide $\frac{\partial l}{\partial w_i}$ gelijk is aan nul worden de gewichten niet meer veranderd. Overigens is het niet gegarandeerd dat je nu ook in het *globale* minimum terecht bent gekomen; het zou ook een sub-optimaal *lokaal* minimum kunnen zijn. Maar in de praktijk van machine learning blijkt dit meestal niet tot al te grote problemen te leiden.

- Als je de waarde van de learning rate α te klein neemt, dan neem je te kleine stapjes. Dit is aangegeven in blauw. Weliswaar nader je in de richting van het minimum, maar de stap is niet groot genoeg om dit helemaal te bereiken. Pas je daarna nogmaals gradient descent toe, dan zul je dus nog een verder stapje zetten in de goede richting, enzovoorts. Op deze manier zul je uiteindelijk wel uiterst nabij het minimum uitkomen, maar je bereikt het nooit helemaal exact, en je hebt er ook veel meer stappen voor nodig dan strict noodzakelijk.

- In rood is geïllustreerd wat er kan gebeuren als je de learning rate α te groot neemt, met een gigantische stapgrootte tot gevolg. Dit kan ertoe leiden dat je over het minimum heen schiet en op een ander deel van de functie terechtkomt. In de figuur beland je op een deel met een steile negatieve helling, waardoor de volgende stap de andere kant opgaat en nog groter is. Je merkt dat je in dit geval het risico loopt alleen maar verder van het minimum vandaan te belanden. De methode convergeert dan niet. Dit is natuurlijk onwenselijk.

Helaas is in de praktijk de ideale waarde van de learning rate α niet goed te voorspellen, aangezien dat van de vorm van de te optimaliseren functie afhangt. Er bestaan heuristieken om deze geschikt te kiezen of gaandeweg aan te passen, maar in afwachting daarvan is een veilige waarde zoals $\alpha = 0.01$ of kleiner aan te raden.

Bij het minimaliseren van de loss wordt door alle trainingsinstance heen gelopen. Elke volgende stap die je zet wordt berekend op grond van telkens een andere instance. Eigenlijk optimaliseren we met elke instance een net iets andere functie: de ene iteratiestap trainen we het model richting het beter herkennen van een eerste instance, en in een volgende stap trainen we het model richting het beter herkennen van een tweede instance. Door de trainingsinstances telkens vrij willekeurig af te wisselen hopen we terecht te komen in een punt dat voor alle instances tezamen een optimaal compromis is. We spreken vanwege deze willekeurige afwisseling ook wel van *stochastic* gradient descent.

Opgave 22. *
Beschrijf in je eigen woorden het verschil tussen de notaties $\frac{\Delta y}{\Delta x}$, $\frac{\mathrm{d}y}{\mathrm{d}x}$, $\frac{\partial y}{\partial x}$, en $\boldsymbol{\nabla}_{\boldsymbol{x}} y$.

Opgave 23. *
Als je k-means clustering uitvoert is het verstandig om het algoritme vele malen te herhalen waarbij je elke keer de cluster centroïden willekeurig initialiseert. Leg uit waarom dit nodig is, en gebruik in je antwoord de termen "lokaal minimum" en "globaal minimum". Hint: zoek de betekenis van de parameter NSTART op in de implementatie van k-means in R, of die van N_INIT in die van python's module *scikit-learn*.

Opgave 24. *
Geef een synoniem voor het woord "stochastisch"; zoek deze term indien nodig op in een woordenboek.

Opgave 25. **
Laat zien dat de helling $\frac{\partial l}{\partial w_i}$ benaderd kan worden door $\frac{\Delta l}{\Delta w_i} = \frac{f(w_i+\Delta w_i)-f(w_i-\Delta w_i)}{2\Delta w_i}$.

Opgave 26. **

Bereken met je rekenmachine (of computer) de afgeleide van de functie $y = \ln(x)$ ter plekke van het punt $x = 2$. Gebruik een numerieke benadering waarbij je gebruik maakt van een stapgrootte $\Delta x = 0.01$ om Δy te bepalen. Maakt het uit of je $\Delta y = f(x + \Delta x) - f(x)$ gebruikt of $\Delta y = \frac{f(x+\Delta x)-f(x-\Delta x)}{2}$? Het exacte antwoord dat je kan vinden door analytisch te differentiëren luidt $\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{1}{2}$; komen je numerieke resultaten hiermee overeen?

2.3. Optimalisatie

Laten we de voorgaande resultaten combineren om ons in staat te stellen de gewichten w_i te variëren op een dusdanige manier dat l geminimaliseerd wordt. We gaan uit van de gradient descent updateregel $w_i \leftarrow w_i - \alpha \cdot \frac{\partial l}{\partial w_i}$. Het probleem is dat we geen eenvoudige manier hebben om de helling van de lossfunctie in termen van de gewichten w_i te bepalen. We moeten immers drie verschillende transformaties achter elkaar toepassen om van de gewichten w_i uiteindelijk tot de loss l te komen. Echter, met een simpel trucje kunnen we de ene afgeleide in deze formule omzetten in drie afzonderlijke afgeleiden die overeenkomen met elk van de drie transformaties:

$$w_i \leftarrow w_i - \alpha \cdot \frac{\partial a}{\partial w_i} \cdot \frac{\partial \hat{y}}{\partial a} \cdot \frac{\partial l}{\partial \hat{y}}$$

Ga voor jezelf na dat dit precies hetzelfde is als de eerdere updateregel omdat de partiële differentialen $\partial\hat{y}$ en ∂a boven en onder de deelstreep telkens tegen elkaar wegvallen. Dit wordt ook wel de *kettingregel* genoemd.

We bekijken de drie partiële afgeleiden nu elk afzonderlijk.

1. De uitdrukking $\frac{\partial a}{\partial w_i}$ geeft aan hoe groot de verandering in de pre-activatiewaarde a is als je het gewicht w_i een klein beetje verandert. Dit zegt iets over de eerste van de drie transformaties $a = \sum_i w_i \cdot x_i$. Het is duidelijk dat als we w_i bijvoorbeeld met 1 laten toenemen, dat dan de uitkomst a met een hoeveelheid x_i toeneemt. Immers, w_i wordt vermenigvuldigd met x_i. Kortom, voor $\Delta w_i = 1$ is de bijbehorende $\Delta a = x_i$, zodat $\frac{\Delta a}{\Delta w_i} = \frac{x_i}{1} = x_i$. Hieruit kunnen we concluderen:

$$\frac{\partial a}{\partial w_i} = x_i$$

2. De uitdrukking $\frac{\partial \hat{y}}{\partial a}$ geeft aan hoe groot de verandering in de post-activatiewaarde $\hat{y}$ is als je de pre-activatiewaarde a een klein beetje verandert. Dit zegt iets over de tweede van de drie transformaties $\hat{y} = \varphi(a)$. Deze afgeleide zegt iets over de helling van de activatiefunctie. Voor veel standaardfuncties kun je de helling bepalen door analytisch te *differentiëren*; deze uitkomsten zijn ook op te zoeken. In het algemeen kun je deze afgeleide ook numeriek bepalen, bijvoorbeeld middels de eerder genoemde benadering:

$$\frac{\partial \hat{y}}{\partial a} \approx \frac{\varphi(a + \Delta a) - \varphi(a - \Delta a)}{2\Delta a}$$

voor een voldoende kleine waarde van Δa.

3. De uitdrukking $\frac{\partial l}{\partial \hat{y}}$ tenslotte geeft aan hoe groot de verandering in de loss l is als je de post-activatiewaarde $\hat{y}$ een klein beetje verandert. Dit zegt iets over de derde van de drie transformaties $l = \mathcal{L}(\hat{y}; y)$. Deze afgeleide zegt iets over de helling van de lossfunctie. Ook hier kun je echter weer een numerieke benadering gebruiken:

$$\frac{\partial l}{\partial \hat{y}} \approx \frac{\mathcal{L}(\hat{y} + \Delta\hat{y}; y) - \mathcal{L}(\hat{y} - \Delta\hat{y}; y)}{2\Delta\hat{y}}$$

voor een voldoende kleine waarde van $\Delta\hat{y}$.

Samengevat, voor elk model dat kan worden geschreven in termen van een lineaire combinatie, een activatiefunctie, en een lossfunctie hebben we hierboven de updateregel $w_i \leftarrow w_i - \alpha \cdot \frac{\partial a}{\partial w_i} \frac{\partial \hat{y}}{\partial a} \frac{\partial l}{\partial \hat{y}}$ afgeleid. Weliswaar komen hier een drietal afgeleiden in voor, maar die kunnen allemaal uitgerekend worden met de formules hierboven (hetzij analytisch, hetzij numeriek).

De modellen die we in het vorige hoofdstuk bekeken hebben, te weten het perceptron en het lineaire regressiemodel, zijn beide voorbeelden van modellen die zo uitgewerkt kunnen worden. Daarnaast zijn er echter nog vele andere modellen mogelijk, waaronder logistische regressie.

Opgave 27. *
Bepaal analytisch of numeriek de afgeleide $\frac{\partial \hat{y}}{\partial a}$ voor de identiteitsfunctie $\varphi(a) = a$.

Opgave 28. *
Omschrijf in woorden de vorm van de afgeleide $\frac{\partial \hat{y}}{\partial a}$ wanneer je de signum-functie als activatiefunctie neemt, dat wil zeggen $\varphi(a) = \operatorname{sgn}(a)$. Deze afgeleide is nauw gerelateerd aan de zogenaamde *Dirac-deltafunctie*.

Opgave 29. *
Leg uit dat de afgeleide van de pre-activatiewaarde a naar de bias b gelijk is aan $\frac{\partial a}{\partial b} = 1$. Je kunt dit bijvoorbeeld doen aan de hand van de gegeven formule $\frac{\partial a}{\partial w_i} = x_i$.

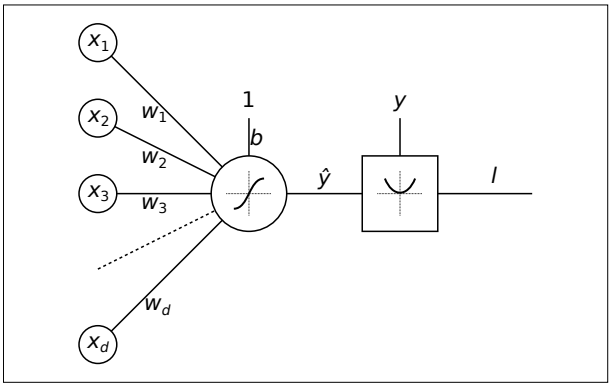
Opgave 30. **
Laat zien dat voor de kwadratische lossfunctie $\mathcal{L}(\hat{y}; y) = (\hat{y} - y)^2$ de partiële afgeleide gelijk is aan $\frac{\partial \mathcal{L}}{\partial \hat{y}} = 2(\hat{y} - y)$. Gebruik hiervoor de numerieke formule $\frac{\partial \mathcal{L}}{\partial \hat{y}} \approx \frac{\mathcal{L}(\hat{y}+\Delta\hat{y};y) - \mathcal{L}(\hat{y}-\Delta\hat{y};y)}{2\Delta\hat{y}}$ en vereenvoudig het resultaat.

Opgave 31. **
Motiveer dat voor de absolute lossfunctie $\mathcal{L}(\hat{y}; y) = |\hat{y} - y|$ de partiële afgeleide geschreven kan worden als $\frac{\partial \mathcal{L}}{\partial \hat{y}} = \operatorname{sgn}(\hat{y} - y)$.

2.4. Het generieke model

Het totale model van een neuron, inclusief lossfunctie, is hieronder schematisch getoond in een diagram. Er is nu een extra component bijgekomen, gesymboliseerd met een

vierkant dat een kwadratische functie bevat, die de voorspelde waarde $\hat{y}$ binnenkrijgt en in combinatie met het ware klasselabel y omzet in de loss l. Stochastic gradient descent wordt nu toegepast op dit diagram in zijn geheel: alle beschikbare vrije parameters (dat wil zeggen de bias b en gewichten $w_1, w_2, \ldots, w_d$) worden in kleine stapjes bijgesteld om het optimalisatiecriterium l aan de rechterkant zo laag mogelijk te krijgen.



Zoals we eerder al zagen wordt lineaire regressie gekarakteriseerd door een activatiefunctie die gelijk is aan de identiteitsfunctie $\varphi(a) = a$. Deze functie doet eigenlijk niets: de post-activatiewaarde is simpelweg gelijk aan de pre-activatiewaarde. Verder wordt bij regressie gebruik gemaakt van de eerder genoemde kwadratische lossfunctie $\mathcal{L}(\hat{y}; y) = (\hat{y} - y)^2$.

De afgeleide van de kwadratische lossfunctie is $\frac{\partial l}{\partial \hat{y}} = 2(\hat{y} - y)$; de afgeleide van de identiteitsfunctie is $\frac{\partial \hat{y}}{\partial a} = 1$; en de afgeleide van de pre-activatiewaarde naar de gewichten is immer $\frac{\partial a}{\partial w_i} = x_i$. Combineren we dit met de stochastic gradient descent updateregel $w_i \leftarrow w_i - \alpha \cdot \frac{\partial a}{\partial w_i} \frac{\partial \hat{y}}{\partial a} \frac{\partial l}{\partial \hat{y}}$ zoals hierboven uiteengezet, dan vinden we:

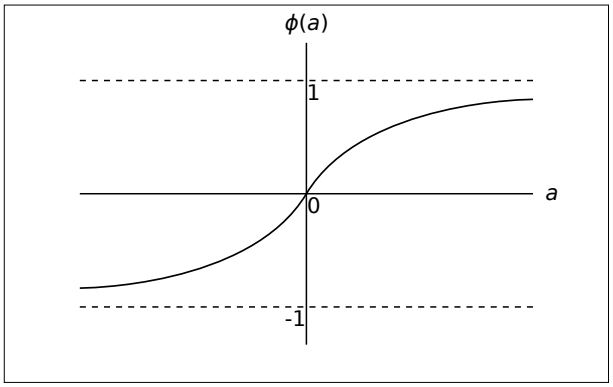
$$\begin{cases} b \leftarrow b - 2\alpha(\hat{y} - y) \\ w_i \leftarrow w_i - 2\alpha(\hat{y} - y)\, x_i \end{cases}$$

Dit is bijna exact de updateregel voor lineaire regressie die we in het vorige hoofdstuk al hadden gevonden! Het enige verschil is de factor 2; soms wordt daarom ook wel een loss-functie $\mathcal{L}(\hat{y}; y) = \frac{1}{2}(\hat{y} - y)^2$ genomen met een extra factor $\frac{1}{2}$, maar door een twee keer zo kleine learning rate α te kiezen kun je het effect van de factor 2 ook volkomen opheffen. Deze updateregel komt dus op hetzelfde neer. Het nieuwe inzicht is dat onze updateregel de lossfunctie minimaliseert over alle modelparameters middels stochastic gradient descent.

In het vorige hoofdstuk hebben we gezien dat het perceptron gebruikt kan worden voor classificatieproblemen. Het perceptron had als kenmerk dat de signum-functie als activatiefunctie wordt gebruikt. Helaas is deze functie niet zo geschikt voor onze huidige aanpak omdat diens helling zich niet netjes gedraagt. Rond $a = 0$ maakt $\operatorname{sgn}(a)$ een sprong. De functie is hier *discontinu* en daardoor niet differentieerbaar: de helling is eventjes oneindig sterk. Ook voor positieve en negatieve a gaat het echter mis. Elders heeft de signum-functie immers overal een helling gelijk aan nul, zodat $\frac{\partial \hat{y}}{\partial a} = 0$. Dit zou

betekenen dat updateregel zegt dat je de gewichten moet bijwerken met een waarde nul. Hierdoor verandert er niets aan de parameters, en wordt het model dus helemaal niet geoptimaliseerd.

Met de resultaten hierboven is het niet zo heel moeilijk om deze problemen te omzeilen. Door de signum-functie af te vlakken tot een gladde S-vormige *sigmoïde* functie kan zowel het probleem van de oneindige helling rondom nul als dat van de helling nul elders worden opgelost. We verkrijgen dan een vorm van *logistische* regressie. De activatiefunctie ziet er dan ongeveer uit zoals hieronder.



Er zijn diverse activatiefuncties $\varphi(a)$ te verzinnen die deze vorm hebben. Een aantal bekende voorbeelden zijn:

- de *tangens-hyperbolicus-functie* $\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$;
- de *softsign-functie* $\text{softsign}(a) = \frac{a}{1+|a|}$;
- de *inverse square root unit* functie $\text{isru}(a) = \frac{a}{\sqrt{1+a^2}}$.

Voor uitkomsten met een bereik tussen 0 en 1 kan de *logistische functie* $\sigma(a) = \frac{1}{1+e^{-a}}$ worden gekozen.

De tangens-hyperbolicus heeft de prettige eigenschap dat diens afgeleide eenvoudig kan worden geformuleerd in termen van zichzelf. Er geldt namelijk dat $\frac{\mathrm{d}\tanh(a)}{\mathrm{d}a} = 1 - \tanh^2(a)$. Voor een model dat een activatiefunctie heeft gelijk aan $\hat{y} = \tanh(a)$ geldt dan dat $\frac{\partial \hat{y}}{\partial a} = 1 - \hat{y}^2$. De updateregel $w_i \leftarrow w_i - \alpha \cdot \frac{\partial a}{\partial w_i} \frac{\partial \hat{y}}{\partial a} \frac{\partial l}{\partial \hat{y}}$ voor stochastic gradient descent leidt dan uiteindelijk tot:

$$\begin{cases} b \leftarrow b - \alpha (\hat{y} - y)\left(1 - \hat{y}^2\right) \\ w_i \leftarrow w_i - \alpha (\hat{y} - y)\left(1 - \hat{y}^2\right) x_i \end{cases}$$

Alle factoren in deze formules zijn bekend of worden reeds berekend tijdens het doen van een voorspelling.

Vergeleken met lineaire regressie is er een extra factor bijgekomen gelijk aan $\left(1 - \hat{y}^2\right)$. Deze gedraagt zich als een wegingsfactor. Voor instances waarvoor $\hat{y} = 0$, dat wil zeggen instances die op de geschatte scheidingslijn tussen de klassen liggen, is deze wegingsfactor gelijk aan 1. Deze instances worden dus als het ware voor 100% meegewogen bij het

bepalen of de scheidingslijn moet worden verschoven. Voor instances waarvoor $\hat{y} = \pm 1$, dat wil zeggen instances die ver van scheidingslijn vandaan liggen en daarom met grote zekerheid als de ene of de andere klasse worden gelabeld, nadert de wegingsfactor naar 0. Deze instances worden dus niet of nauwelijks meegewogen bij het aanpassen van de modelparameters. Het idee hierachter is dat instances die ver van de scheidingslijn af liggen na een kleine aanpassing toch nog immer ver aan dezelfde kant van de lijn blijven liggen, dus voor hen heeft het niet zoveel zin om de lijn te verschuiven. Voor instances die vlak bij de scheidingslijn liggen is het echter wel degelijk zeer relevant of die lijn enigszins verschoven wordt, dus deze worden het zwaarst meegeteld bij het aanpassen van de modelparameters. Dit gedrag is anders dan bij lineaire regressie, waar alle instances altijd even zwaar meewegen.

Opgave 32. *

Het perceptron en logistische regressie zijn beide modellen om classificatie uit te voeren. Beschrijf een aantal verschillen tussen deze twee methoden.

Opgave 33. *

Leg uit dat een helling exact gelijk aan nul voor de activatiefunctie, dat wil zeggen $\frac{\partial \hat{y}}{\partial a} = 0$, ertoe leidt dat de updateregel de gewichten niet langer bijwerkt. Is dit een probleem?

Opgave 34. **

Voor een *even* functie geldt dat $f(-x) = f(x)$, terwijl voor een *oneven* functie $f(-x) = -f(x)$. Toon aan dat de activatiefuncties $\tanh(a)$, $\text{softsign}(a)$, en $\text{isru}(a)$ alledrie oneven functies zijn.

Opgave 35. **

Laat zien dat de logistische functie $\sigma(a) = \frac{1}{1+e^{-a}}$ gerelateerd is aan de tanh-functie volgens $\tanh(a) = 2 \cdot \sigma(2a) - 1$, of omgekeerd $\sigma(a) = \frac{1+\tanh(a/2)}{2}$. Schets de beide functies samen in één grafiek.

Opgave 36. ***

De afgeleiden van de softsign- en isru-functies kunnen eveneens geschreven worden in termen van zichzelf, namelijk $\frac{\mathrm{d}\text{softsign}(a)}{\mathrm{d}a} = (1 - |\text{softsign}(a)|)^2$ en $\frac{\mathrm{d}\text{isru}(a)}{\mathrm{d}a} = \left(1 - \text{isru}^2(a)\right)^{\frac{3}{2}}$. Leid hiermee de updateregels af voor logistische regressiemodellen waarin deze functies worden gebruikt als activatiefunctie, samen met een kwadratische lossfunctie. Leg uit dat deze modellen verschillen van lineaire regressie door een hoge weging van instances die dichtbij de geschatte scheidingslijn tussen de twee klassen liggen en een lage weging van instances die hier ver vandaan liggen.

Opgave 37. ***

Zoek zelf informatie op over de sigmoïde Gudermannfunctie $\text{gd}(a)$ en de formule voor diens afgeleide. Kun je de afgeleide van de Gudermannfunctie schrijven in termen van de Gudermannfunctie zelf (vergelijkbaar met hoe dat in de vorige opgave voor de softsign en isru functies werd gedaan)?