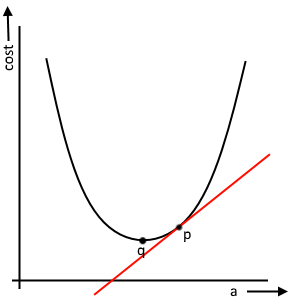
# Verbeteren

In de vorige deelvraag hebben we verschillende Machine Learning algoritmes behandeld. Hierbij hebben we nog niet besproken hoe een algoritme zichzelf kan verbeteren: Hoe bepalen we bij Linear Regression de waarden voor a en b in de formule y = ax + b? Hoe bepalen we de waarden voor x en b in de vector bij een Support Vector Machine? Hoe bepalen we de wegingen van de synapsen in een Artificial Neural Network? Er zijn verschillende manieren waarop al deze waarden bepaald kunnen worden: Evolutionary Improvement, Newton’s Method en Gradient Descent. Deze drie leerstrategieën zullen we in deze deelvraag behandelen.

# Gradient Descent

De laatste leerstrategie die we behandelen is **Gradient Descent.** Gradient Descent is een algoritme dat functies minimaliseert. Door het aanpassen van bepaalde parameters wordt geprobeerd de waarde van een bepaalde functie zo laag mogelijk te maken. De functie die we bij een zelflerend systeem proberen te minimaliseren is de **cost** **function**, ook wel **loss function** genoemd. Dit is een functie die bepaalt hoe goed het systeem op dat moment werkt. Er wordt bepaald hoeveel de huidige outputs afwijken van de gewenste outputs. Het is hierbij dus nodig dat je de gewenste outputs weet bij gegeven inputs. Er is dus bij gradient descent altijd sprake van supervised learning.   
  
Het algoritme

Figuur - De cost function

In figuur **\*\*\***  is de cost van een bepaalde situatie uitgezet tegen een variabele a. Dit kan bijvoorbeeld de a uit de formule bij Linear Regression zijn. Het is dus te zien dat de cost minimaal is in punt q. We willen dus dat a gelijk wordt aan de waarde van a in punt q. Nu is dit punt in deze grafiek erg makkelijk te vinden, maar zodra er gebruik wordt gemaakt van ingewikkeldere algoritmen, zoals een ANN, wordt dit punt moeilijker te bepalen.  
Op een gegeven moment in het trainingsproces is de a gelijk aan het punt p. Het Gradient Descent algoritme doet dan het volgende:

* De afgeleide op het huidige punt wordt bepaald (de rode lijn in figuur \*\*\*\*).
* De a wordt zodanig aangepast dat het meer in de richting komt van de q. (Dit wordt gedaan door de afgeleide bij de variabele op te tellen)

Wanneer Gradient Descent wordt toegepast zal een bepaalde variabele in een zelflerend systeem zo aangepast worden dat de cost als gevolg van die variabele het laagst wordt.

De wiskunde achter gradient descent  
Het Machine Learning algoritme produceert met een bepaalde input een bepaalde output, dit noemen we de **guess**. Omdat we weten wat de goede ouput is kunnen we de **error** bepalen voor die input. De goede output in de volgende formule is y.

De vorige formule geldt dus voor de individuele datapunten. De totale error, de som van alle individuele error waarden, ook wel cost of loss genoemd kan als volgt beschreven worden:

Zoals bekend uit de wiskunde is het mogelijk om hiervan de laagste waarde te bepalen door de afgeleide op nul te herleiden. Voor elk individueel datapunt is de afgeleide van de error:

Bij het differentiëren wordt gebruik gemaakt van de kettingregel.

Linear Regression met gradient descent  
Om het principe van Gradient Descent beter te begrijpen gaan we nu doormiddel van Gradient Descent Linear Regression uitvoeren. **LEES EERST STUKJE OVER LINEAR REGRESSION**

De guess is hier dus de huidige uitkomst van

De waarde van bi, x­i en yi zijn hier constant. De x­i en yi zijn namelijk bekend uit de training data en bi veranderd wel, maar niet hierbij. De afgeleide van de error functie is dan:

De afgeleide van de cost function is dan:

Met de deze afgeleide is de helling van de cost function te bepalen. Hiermee dus te bepalen welke richting we de variabele a in moeten veranderen. Het aanpassen van de a bij Linear Regression gebeurd dus als volgt:  
a = a + ()  
  
Learning rate  
Een zelflerend systeem bereikt niet in een keer de gewenste output. Er wordt langzaam in de richting van de goede output gewerkt. De formule voor het aanpassen van de *a* waarde uit het vorige kopje is daarom iets anders. Er wordt een learning rate geïntroduceerd:

a = a + ( \* learning\_rate)

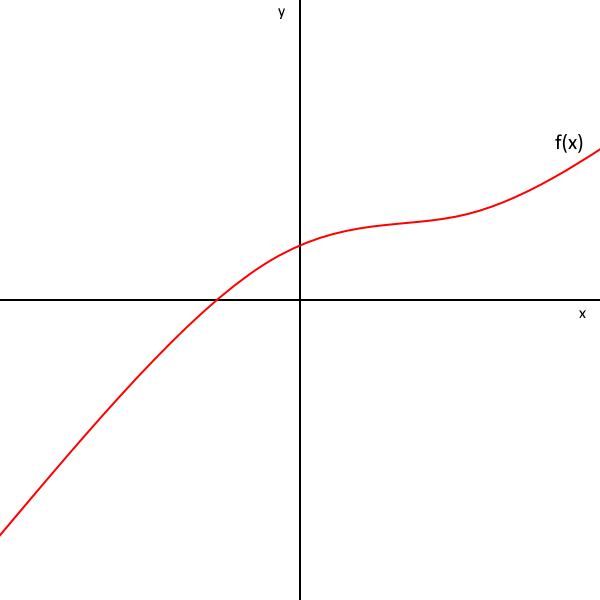
Het kiezen van een goede learning rate is heel belangrijk. Een te lage learning rate zorgt ervoor dat het heel lang duurt voordat de goede output bereikt wordt. Een te hoge learning rate zorgt ervoor dat de gewenste output voorbij wordt geschoten. De gewenste output wordt dan nooit bereikt omdat de variabele net te groot of te klein wordt gemaakt. [1,2]

# Newton’s method

Net zoals gradient decent is Newton’s method, vernoemd naar Isaac Newton, een manier om de laagste waarde van een bepaalde functie te bepalen. Hiervoor maakt gradient descent gebruik van het gegeven dat een extreme waarde van een grafiek een richtingscoëfficiënt van 0 heeft en de afgeleide op dat punt dus gelijk is aan 0. Newton’s method gebruik voor het bepalen van de laagste waarde de tweede afgeleide. Er wordt dan gekeken op welke punten deze lijn de x-as snijdt, dit zijn namelijk de toppen van de grafiek van de eerste afgeleide. Door gebruik te maken van Newton’s method, zal je een schatting krijgen van het snijpunt met de x-as, maar waarschijnlijk zal je dit punt niet exact kunnen vinden. De nauwkeurigheid van de schatting hangt af van de hoeveelheid waarmee je de stappen herhaalt.

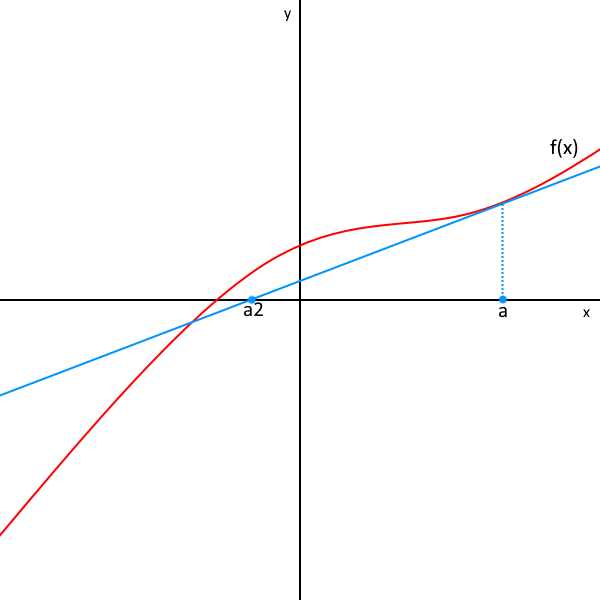
## Het proces

In figuur \*\*\*\* is een willekeurige grafiek getekend.



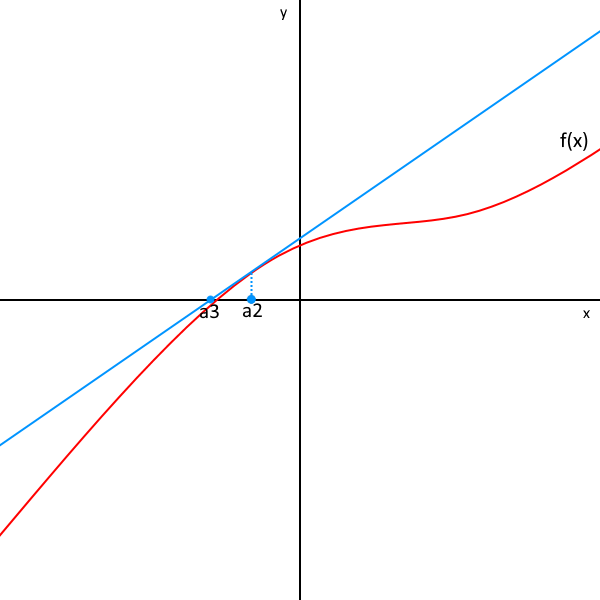
Figuur \*\*\*\*: de grafiek van een willekeurige functie f(x)

Bij het gebruik van Newton’s method, waarbij we dus zoeken naar een snijpunt met de x-as, wordt eerst een gok gedaan. Deze gok, op punt a, correspondeert met een waarde op de grafiek van . Aan dit punt wordt een raaklijn getekend.



Figuur \*\*\*\*

De raaklijn van op snijdt de x-as op een bepaald punt a2. Te zien is dat dit punt al aanzienlijk dichten bij het doel ligt dan de originele schatting. Ook a2 correspondeert met een waarde van en ook op dit punt kan weer een raaklijn getekend worden.



Figuur \*\*\*\*

Na slechts twee raaklijnen getekend te hebben, ligt het punt a3 al erg dicht bij het doel. Om een nauwkeurigere benadering van dit doel te bereiken kan je vaker een raaklijn tekenen en het nieuwe snijpunt bepalen. Hoe nauwkeurig je een benadering wil hebben verschilt per situatie.

## De wiskunde

Uiteraard zijn de waardes van de punten a2, a3 … an te berekenen, we willen immers de waarde van het nulpunt bepalen. Dit gebeurt als volgt:

We weten dat de afgeleide de helling van de grafiek aangeeft: . Het idee is dat je de helling berekent tussen twee punten die oneindig dicht bij elkaar liggen, hier aangegeven met d: . Voor het gemak noemen we deze punten en . Dit geeft voor de afgeleide:

Dit kan omgeschreven worden tot de formule voor een raaklijn:

Om het nulpunt te berekenen moet gelden . Omdat slechts een andere waarde voor aanduidde, kunnen we deze vervangen door het volgende: en .

Met deze formule kan de volgende waarde voor berekend worden.

## Het gebruik

Er zijn vele situaties te bedenken waarin je de nulpunten van een functie zou willen weten. In het gebied van machine learning wordt het gebruikt om te berekenen waar de cost functie minimaal is. Onder het kopje gradient descent staat al beschreven hoe we aan de cost functie komen en wat de afgeleide hier van is. De grafiek die afgebeeld staat zou de afgeleide van deze cost functie zijn. Dit betekent namelijk dat de tweede afgeleide van de cost functie wordt genomen wanneer je een raaklijn aan de grafiek berekent.  
Gradient descent kan goed worden gebruikt bij grafiek met slechts één minimum. Zodra dit niet het geval is, kan het makkelijk in een dal vast blijven hangen, denkend dat het de minimale waarde gevonden heeft, terwijl er misschien nog een lager punt te vinden is. Bij zulke gevallen kan Newton’s method ingezet worden, want al deze punten zullen wel op de x-as liggen en dus zullen ze allemaal te vinden zijn met Newton’s method.

# Evolutionary improvement

Het leerproces van een systeem zou de evolutie van het systeem genoemd kunnen worden, het leert zichzelf beter te functioneren in een omgeving. Net zoals evolutie in de biologie, gaat evolutionary improvement in generaties van systemen. Deze manier van leren maakt gebruik van het doorgeven van informatie tussen deze generaties om het algemene niveau van presteren te verhogen.

## DNA

Wanneer evolutionary improvement wordt toegepast, is er altijd sprake van een bepaald DNA. In dit DNA staan een aantal waardes. Deze waardes kunnen worden doorgegeven aan de volgende generatie.

## Een populatie

Wanneer de informatie van een enkel individu telkens wordt doorgegeven aan een volgende generatie die ook bestaat uit slecht één individu, zal de verbetering van een systeem niet zo groot of zelfs afwezig zijn. Het systeem weet niet of het DNA dat doorgegeven wordt goed of slecht presteert, want er is maar één individu per generatie. Om deze reden bestaat een generatie meestal uit meerdere individuen. Ze zullen niet allemaal even goed presteren en dus zal er onderscheid gemaakt kunnen worden tussen goed en slecht.  
Hoe de overgave van DNA wordt geregeld kan op veel verschillende manieren en is afhankelijk van het soort programma en de voorkeur van de programmeur. Je zou bijvoorbeeld de individuen uit een populatie kunnen rangschikken op volgorde van prestatie (hoe prestatie wordt gemeten is natuurlijk ook geheel afhankelijk van het soort programma) en een bepaald percentage van het slechtst presterende deel laten afvallen1. Vervolgens vul je dit deel weer op met individuen met een willekeurig DNA, de rest van de populatie blijft gelijk. Het idee is dat je door telkens het slechte DNA weg te filteren, uiteindelijk een populatie krijgt die gemiddeld steeds beter presteert.  
Een andere manier voor evolutie is stellen dat na een bepaalde tijd elk individu een ‘kind’ krijgt. Dit zou goed kunnen werken in een simulatie waarin individuen een rivaliserend verband met elkaar hebben, bijvoorbeeld doordat ze dezelfde voeding nodig hebben. De individuen die langer overleven zullen meer kinderen krijgen en hun DNA dus vaker doorgeven, terwijl de individuen met slechte eigenschappen snel doodgaan. Natuurlijk kan je er ook voor kiezen het DNA van meerdere individuen te combineren voor een volgende generatie.

## Mutaties

In de biologie kan in het DNA een mutatie plaatsvinden. Een mutatie is een willekeurige verandering zonder echte reden. Nu kunnen deze mutaties nadelig zijn door bijvoorbeeld ziektes te veroorzaken, maar voor evolutie zijn ze erg nuttig. Zonder mutaties zou het DNA altijd gebonden blijven aan wat er al bestaat omdat het telkens wordt doorgegeven. Zo zou er niets nieuws kunnen ontstaan en zou het systeem misschien vast komen te zitten. Als je bijvoorbeeld een programma hebt waarin een systeem leert een hindernis baan over te gaan, maar geen enkel individu heeft in zijn DNA de informatie om te springen staan, dan kan het programma nooit over een horde heen komen. Mutaties dienen ervoor zulke problemen te voorkomen. Je voegt een bepaalde mutatiefactor toe, een kleine kans die ervoor zorgt dat het programma soms een willekeurige verandering aanbrengt waardoor nieuwe mogelijkheden voor de individuen kunnen ontstaan.

# Bronnen

###### [1] <https://www.coursera.org/learn/machine-learning/lecture/kCvQc/gradient-descent-for-linear-regression> Auteur: Andrew Ng Bezocht op: 17-7-2017 [2] https://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/ Auteur: [Matt Nedrich](https://spin.atomicobject.com/author/nedrich/)

Bezocht op: 17-7-2017