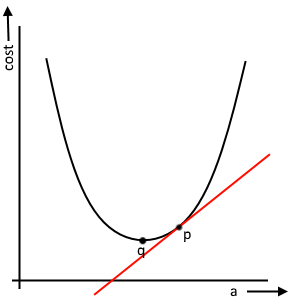
# Verbeteren

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 1 - 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]

# 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