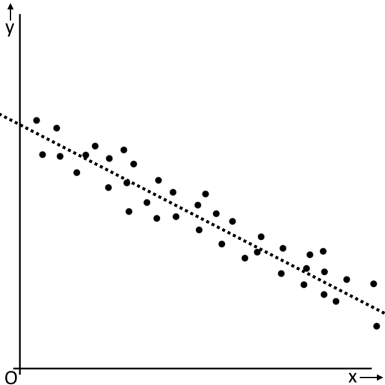
# Deelvraag Inleiding

In de vorige deelvraag hebben we behandeld wat een zelflerend systeem is. In deze deelvraag gaan wij dieper in op de verschillende soorten zelflerende algoritmes. We zullen in deze deelvraag naar **drie** verschillende algoritmes kijken: *Linear Regression*, *Support vector machines* en *Artificial Neural Networks*. Verder zullen we naar een belangrijk deel van een zelflerend systeem, namelijk hoe het zichzelf verbeterd. Ook hiervoor kijken we naar **drie** verschilende manieren van het verbeteren van een zelflerend systeem: *Evolutionary, Newton’s Method* en *Gradient Descent.* [1]

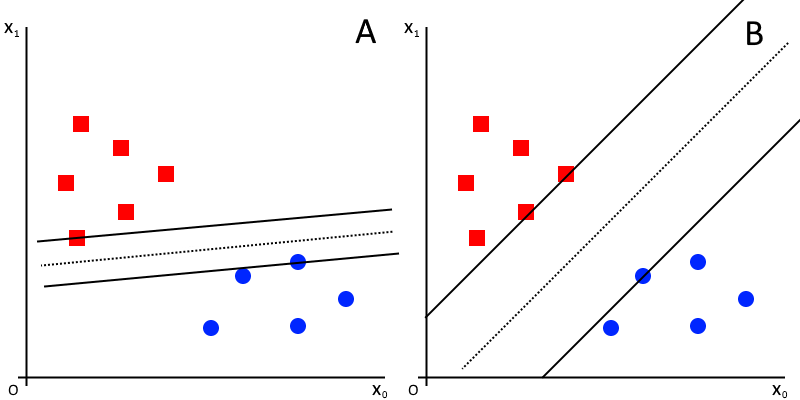
# Linear Regression

Het eerste *machine learning* algoritme dat we gaan behandelen is linear regression. Dit kan gebruikt worden voor het voorspellen van een y-waarde bij gegeven x-waarde(n). Om linear regression te kunnen gebruiken is het belangrijk dat er wel een linear verband bestaat tussen de x-waarde(n) en de y-waarde. In figuur \*\*\*\* is een linear verband te zien.   
Dit lineare verband is te beschrijven met de formule:  
y = ax + b

Figuur

# Support vector machine

Een support vector machine (SVM) is een machine learning algoritme ontwikkeld door vladimir vapnik. Het algoritme kan gebruikt worden voor het classificeren van data. Het algoritme heeft training data nodig en is dus een vorm van supervised learning. [2]



Figuur 2 Support vector machine

Een support vector machine werkt als volgt: Het trekt een lijn, een **vector**, tussen de twee groepen. Deze vector is zo dat *de afstand tussen de vector en het dichtstbijzijnde datapunten zo groot mogelijk is* [3]Deze dichtstbijzijnde datapunten worden de **support vectoren** genoemd In figuur \*\*\* is twee keer dezelfde dataset weergegeven. In de linker afbeelding is te zien dat de vector de twee groepen scheidt maar de afstand tussen het dichtstbijzijnde datapunt kleiner is dan bij de rechter afbeelding, deze afstand wordt de marge genoemd. De in de rechter afbeelding is de marge het grootst, dus dit is de betere vector. Het gebied tussen de twee support vectoren wordt het **hyperplane** genoemd.

## Het algoritme

Het doel van het algoritme is van een nieuw datapunt bepalen of het tot groep A (de rode vierkantjes) of groep B (de blauwe cirkels) behoort. Als een nieuw datapunt behoort tot groep A dan willen we dat de output van het algoritme negatief is en als het nieuwe datapunt behoort tot groep B willen we dat de output positief is. Hoe positiever of negatiever de output is hoe zekerder het is dat dit punt daadwerkelijk tot die groep behoort. Als de output 0 is, dan bevindt het punt zich precies tussen de twee groepen, het ligt dan op de stippellijn van figuur \*\*\*\*. Verder is het zo dat de output tussen -1 en 1 ligt als het binnen de twee support vectoren ligt. In dit gebied is het niet helemaal zeker tot welke groep het punt behoort.

We kunnen de drie vectoren als volgt definiëren:  
De linker support vector   
De middelste vector   
De rechter support vector   
Een support vector machine probeert het volgende:

* Alle datapunten moeten buiten de twee support vectoren liggen
* De afstand tussen de support vectoren moet zo groot mogelijk zijn

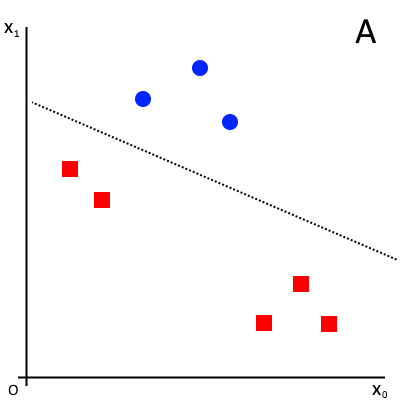
Vanuit de vectoren zijn de volgende twee formules af te leiden:

De formule voor of een datapunt buiten de twee support vectoren ligt:   
De formule voor de afstand tussen de twee support vectoren:   
  
Een vector die voldoet aan de volgende eisen wordt gekozen:

1. Voor alle datapunten moet gelden:
2. moet zo klein mogelijk zijn, ofwel zo groot mogelijk

## Kernel Methods

In veel gevallen zal de dataset niet zo mooi geordend zijn als in figuur \*\*\*. Het is dan niet mogelijk om een rechte lijn te trekken die de twee groepen scheidt. Een support vector machine zou in dit geval dus niet werken. Om toch een support vector machine te kunnen gebruiken is er iets genaamd de Kernel Trick.

In figuur \*\*\*\* is een één dimensionale dataset te zien, dit wil zeggen, er is maar één variabele. Met een support vector machine is het nu niet mogelijk om een lijn te trekken die de twee groepen scheidt. Daarom wordt er een extra variabele bij gemaakt, bijvoorbeeld X1 = (X0)2. Nu is het wel mogelijk een lijn te trekken door de dataset die de twee groepen opdeelt:

# Verbeteren

## Gradient Descent

Een andere manier om het verband tussen een x-waarde en y-waarde te berekenen is door gebruik te maken van **gradient descent.** We proberen weer een lijn ( y = ax + b) te vinden die het best het verband tussen de y- en de x-waarde beschrijft. Er wordt op het begin gekozen voor een willekeurige waarde voor a en b. Deze willekeurige lijn wordt gebruikt om “gok” te krijgen voor elk van de x-waarden. Deze gok zal bij een willekeurige lijn dus erg afwijken van de echte y-waarde. De afwijking noemen we de **error**. Deze error berekenen we als volgt:

Het doel van het algoritme is om zo klein mogelijk te maken, ofwel de totale *error* zo klein mogelijk te maken.

## Newton’s Method

## Evolutionary

# Bronnen

[1] https://www.analyticsvidhya.com/blog/2015/08/common-machine-learning-algorithms/  
Geraadpleegd op: 3-6-2017  
Laatst gewijzigd op: August 10, 2015  
Auteur: Sunil Ray

[2] https://books.google.nl/books?hl=nl&lr=&id=HUnqnrpYt4IC&oi=fnd&pg=PP7&dq=support+vector+machines&ots=g8lIEB0rSi&sig=FTLWxhxAwcf95E1xLoWZ8WYFZ4k#v=onepage&q=support%20vector%20machines&f=false  
Titel: Support Vector Machines

Door Ingo Steinwart,Andreas Christmann

[3] <http://www.saedsayad.com/support_vector_machine.htm>  
Geraadpleegd op: 4-6-2017  
Auteur:

[4] https://www.svm-tutorial.com/2015/06/svm-understanding-math-part-3/  
Geraadpleegd op: 4-6-2017  
Auteur: Alexandre KOWALCZYK