Praktijk 1

# Inleiding

Met de kennis die we hebben opgedaan bij het maken van het theorieverslag gaan we zelf een praktijk opdracht uitvoeren. We dagen onszelf tot het zelf maken van een computerprogramma dat de volgende vraag beantwoordt: *welke onderdelen zijn nodig voor een zelflerend computersysteem, in de vorm van een door ons ontworpen computerprogramma, dat in staat is afbeeldingen te classificeren binnen vastgestelde categorieën met een precisie van meer dan 80%?*

# Werkwijze

Voordat we kunnen beginnen aan het ontwerp van het systeem moeten we eerst duidelijk maken wat we precies willen bereiken en hoe we dat willen bereiken. We moesten een aantal vragen eerst beantwoorden:  
***Welke afbeeldingen gaan we proberen te classificeren?***We gaan proberen handgeschreven cijfers te classificeren. We gebruiken hiervoor de MNIST dataset. [1] Dit is een dataset van een groot aantal afbeeldingen van handgeschreven cijfers met het bijbehorende label. De afbeeldingen zijn elk 28 x 28 pixels groot. Deze dataset is erg geschikt om Machine Learning op toe te passen.  
**Welk algoritme gebruiken we voor het classificeren van de afbeeldingen?**Wij gebruiken een Neural Network (**ref**) om deze afbeeldingen te classificeren. Het is gebleken dat dit het beste algoritme is voor het classificeren van afbeeldingen. Er zijn 28 \* 28 = 784 input neuronen en 10 output neuronen. We kiezen ervoor om één *hidden layer* te maken.  
**Welke manier van leren gebruiken we voor het verbeteren voor het algoritme?**We gaan gebruik maken van Gradient Descent (**ref**). Dit lijkt ons de meest geschikte leer strategie voor dit probleem.

# Het netwerk

Het aantal input neuronen en het aantal output neuronen staat vast (respectievelijk 784 en 10). Het aantal hidden neuronen kunnen we zelf bepalen. Ook moeten nog een bias toevoegen (**ref)**. Dit is één neuron die altijd de waarde 1 heeft. Zoals uitgelegd in **(ref)** moeten we ook een activation function kiezen. Hiervoor gebruiken we een *sigmoid* functie (zie figuur x). De lagen zijn onderling volledig verbonden. Elke verbinding heeft zijn eigen weging.

Figuur 2 Schematische weergave Neural Network

# Gradient Descent met backpropergation

We hebben nu een werkend Neural Network. Nu is het een kwestie van het vinden van de juiste *weights*. Dit is makkelijker gezegd dan gedaan. We moeten nu Gradient Descent toepassen: *het bepalen van de afgeleide van de cost function relatief tot de te veranderen variable.*   
Het bepalen van de totale cost function is vrij simpel:

Wat hier eigenlijk staat is: het verschil tussen de juiste output en de gegeven output van neuron m

# Resultaat

# Conclusie

# Bronnen

[1]: http://yann.lecun.com/exdb/mnist/