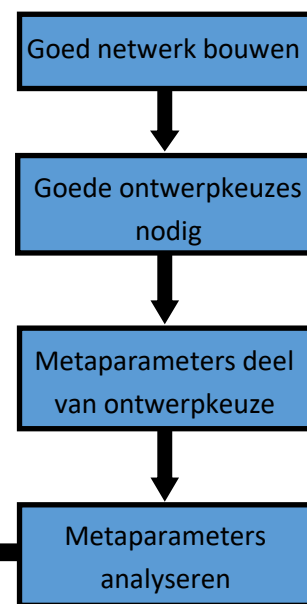


# Analyseren van de invloed van metaparameters op neurale netwerken

## Motivatie

Bij het bouwen van neurale netwerken:

- Lukrake keuzes van metaparameters kunnen leiden tot zeer trage training of onnauwkeurige resultaten.
- Analyse van metaparameters moet leiden tot betere ontwerpkeuzes.



## Schets van de werking

Neurale netwerken getraind met stochastic gradient descent.

$$Cost(p) = \underbrace{\frac{1}{2m} \sum_{i=1}^m \|y(x_i) - a(x_i)\|_2^2}_{\text{Netwerk fout}} + \underbrace{\frac{\lambda}{2m} \sum_{j=1}^n p_j^2}_{\text{Regularisatie}}$$

Verschillende waarden voor stap- en batch-grootte:

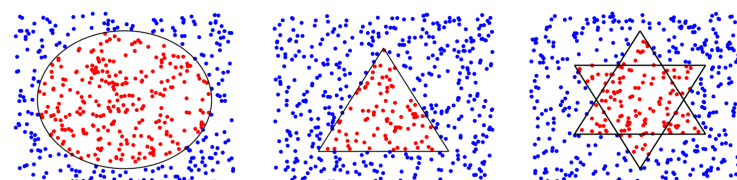
STAP-GROOTTE	BATCH-GROOTTE
0.05 0.15 0.25 0.35	1 10 100 500

Andere (meta)parameters constant houden.

Analyseren van metaparameters: *stap-grootte* en *batch-grootte*.

Invloed analyseren met:

- Eenvoudige netwerken
- Simpele vormen van data: cirkel, driehoek en ster

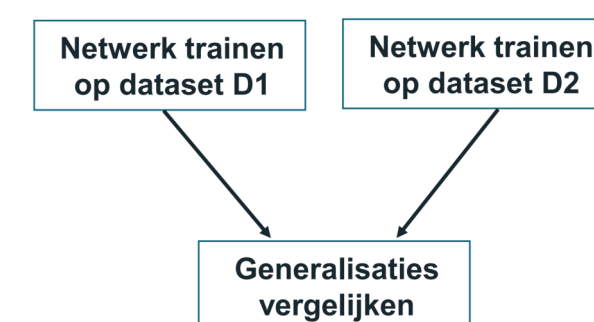


Performantie analyseren aan de hand van:

- Compact datarooster
- Percentage correcte classificaties

Sensitiviteit analyseren aan de hand van:

- Dataset D1
- Een datapunt van D1 perturberen
- Resulteert in dataset D2



→ *Sensitiviteit draagt bij aan performantie analyse*

## Referenties

- Deep Learning: An Introduction for Applied Mathematicians (2020)*
- Train faster generalize better: Stability of stochastic gradient descent (2015)*
- Hands-on Bayesian Neural Networks - a Tutorial for Deep Learning Users (2020)*

## Generalisatie performantie analyseren

### Stap-grootte (enkel cirkel en ster):

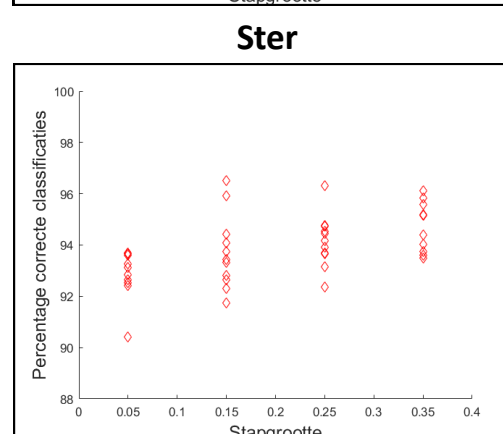
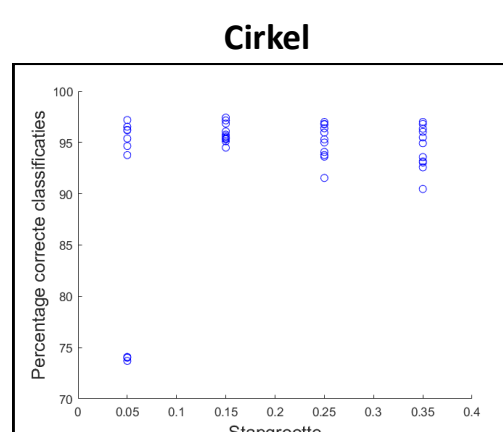
Resultaten uit de grafieken roepen bepaalde bevindingen/vragen op:

**Cirkel:**

- Stap van 0.05 minder betrouwbaar voor het halen van een hoog percentage correcte classificaties. → Mogelijks geen convergentie.
- Lijkt op een dalende trend bij hogere stap-grootte. → Gevoeliger om over minimale waarde van de kostfunctie te springen bij hogere stap-grootte?

**Ster:**

- Groot percentage correcte classificaties voor iedere stap-grootte maar een stijgende trend. → Mogelijks geen convergentie. Grotere stap-grootte mogelijk?



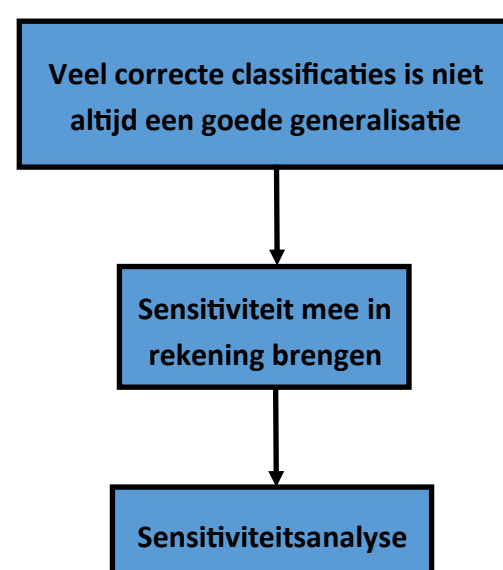
### Batch-grootte:

Analyse van batch-grootte is idem aan deze van stap-grootte:

- Uitvoeren van testen
- Bevindingen afleiden uit resultaten
- Inzicht verkrijgen over de invloed van deze meta-parameter op het aantal correcte classificaties.

### Generalisatie performantie:

Combinatie van percentage correcte classificaties en sensitiviteit van het resultaat.



## Generalisatie sensitiviteit analyseren

### Perturbatie van datapunt:

Niet alle perturbaties zijn even nuttig om generalisatie sensitiviteit te analyseren:

- Enkel punten gaan perturberen op een manier die een invloed kan uitoefenen op de generalisatie performantie.
- Classificatie van datapunten in de buurt van de rand van de data vorm omkeren.
- Creëren van één 'ruis' punt.

### Doel:

- Diepere analyse van netwerk + parameter combinaties die een hoog percentage correcte classificaties hebben.
- Indien het resultaat zeer sensitief is ten opzichte van de data kan dit op een slechte generalisatie duiden.

### Testen:

$n$  willekeurige verschillende perturbaties uitvoeren → steekproef:

- Gemiddelde absolute verschil berekenen tussen het percentage correcte classificaties met ( $= v_i$ ) en zonder perturbatie ( $= u_i$ ):

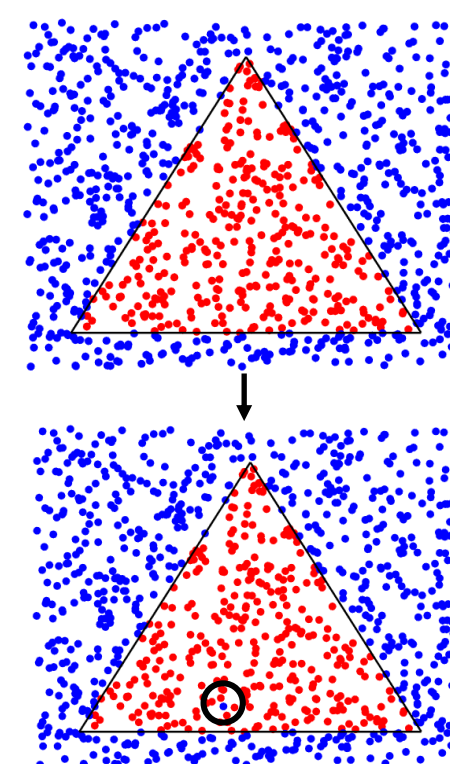
$$\bar{X} = \frac{\sum_{i=1}^n |u_i - v_i|}{n}$$

- Gebruikmaken van t-test om te bepalen of er een significant verschil is van nul:

$$T = \frac{\bar{X} - \mu_0}{S/\sqrt{n}} \text{ met } S = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (|u_i - v_i| - \bar{X})^2}$$

Nulhypothese  $H_0: \mu_0 = 0$

Alternatieve hypothese  $H_1: \mu_0 > 0$



## BAYESIAANS

### Motivatie

In veel toepassingen moet een neuraal netwerk beslissingen nemen met een bepaalde zekerheid:

- Zelfrijdende auto's
- Autonome reactoren
- Robots

→ Bayesiaanse algoritmes

→ Bevatten ook metaparameters

→ Analyse van de invloed van metaparameters op onzekerheid mogelijk

### Geanalyseerd algoritme

#### Metropolis-Hastings:

```

Draw  $\mathbf{x}_0 \sim \text{Initial}$ 
while  $n = 0$  to  $N$  do
  Draw  $\mathbf{x}' \sim Q(\mathbf{x}|\mathbf{x}_n)$ 
   $p = \min\left(1, \frac{Q(\mathbf{x}'|\mathbf{x}_n) f(\mathbf{x}')}{Q(\mathbf{x}_n|\mathbf{x}') f(\mathbf{x}_n)}\right)$ 
  Draw  $k \sim \text{Bernoulli}(p)$ 
  if  $k$  then
     $\mathbf{x}_{n+1} = \mathbf{x}'$ 
     $n = n + 1$ 
  end if
end while
  
```

Met:  $Q(\mathbf{x}'|\mathbf{x}_n) = \mathcal{N}(\mathbf{x}_n, \sigma^2)$   
 →  $Q$  symmetrisch verdeeld  
 →  $p = \min\left(1, \frac{f(\mathbf{x}')}{f(\mathbf{x}_n)}\right)$

### Onzekerheid visualiseren

Trekken van  $n$  verschillende netwerken uit het resultaat van Metropolis-Hastings algoritme:

- Data door ieder netwerk sturen
- Twee mogelijkheden (A of B)

Voor ieder datapunt:

→  $n_1$  netwerken die punt als A classificeren

→  $n_2$  netwerken die punt als B classificeren

$$\frac{|n_1 - n_2|}{n} \approx 0 \text{ Zeer onzeker}$$

$$\frac{|n_1 - n_2|}{n} \approx 1 \text{ Zeer zeker}$$

#### Visualisatie data cirkel:

