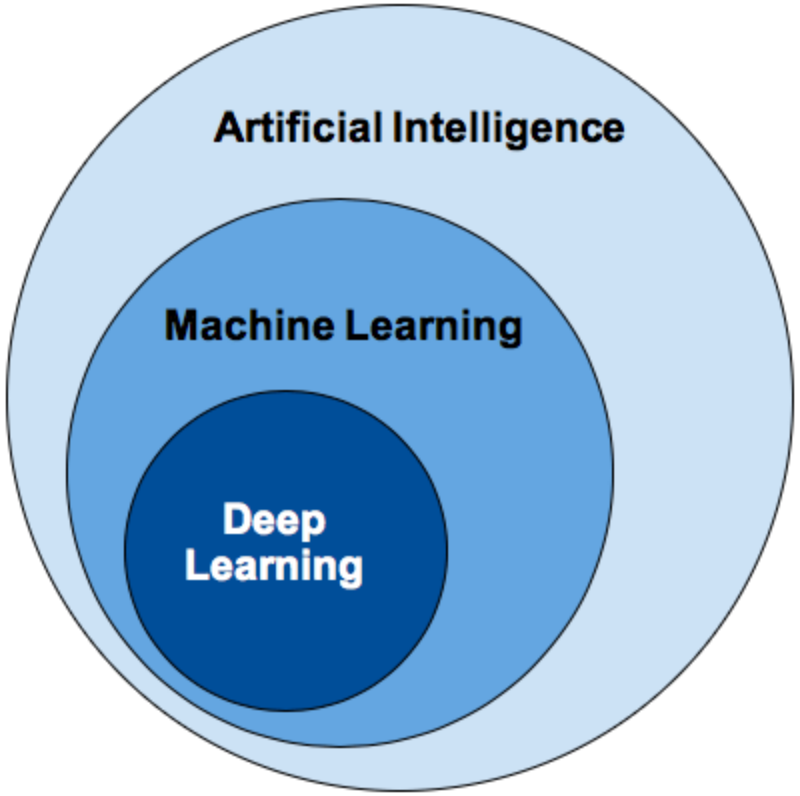
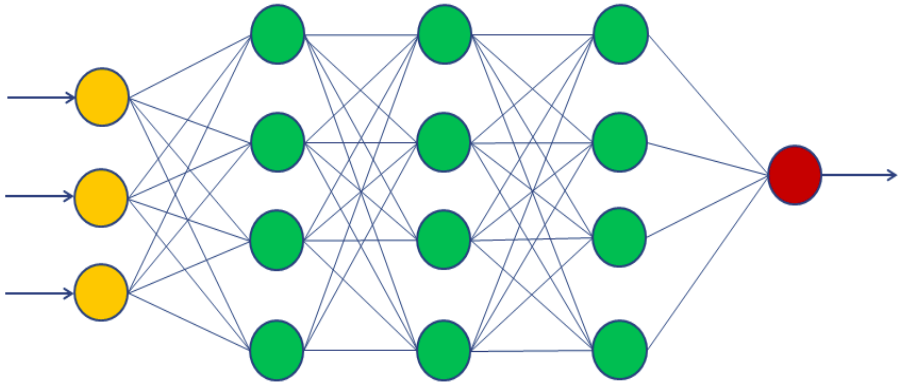
KU LEUVEN

Sensitiviteitsanalyse en onzekerheidskwantificatie van deep learning methodes

Joppe De Jonghe, Juha Carlon

Deep learning?





PANDA

“Spraakherkenning!”

Neurale netwerken

Input vector

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Neuronen

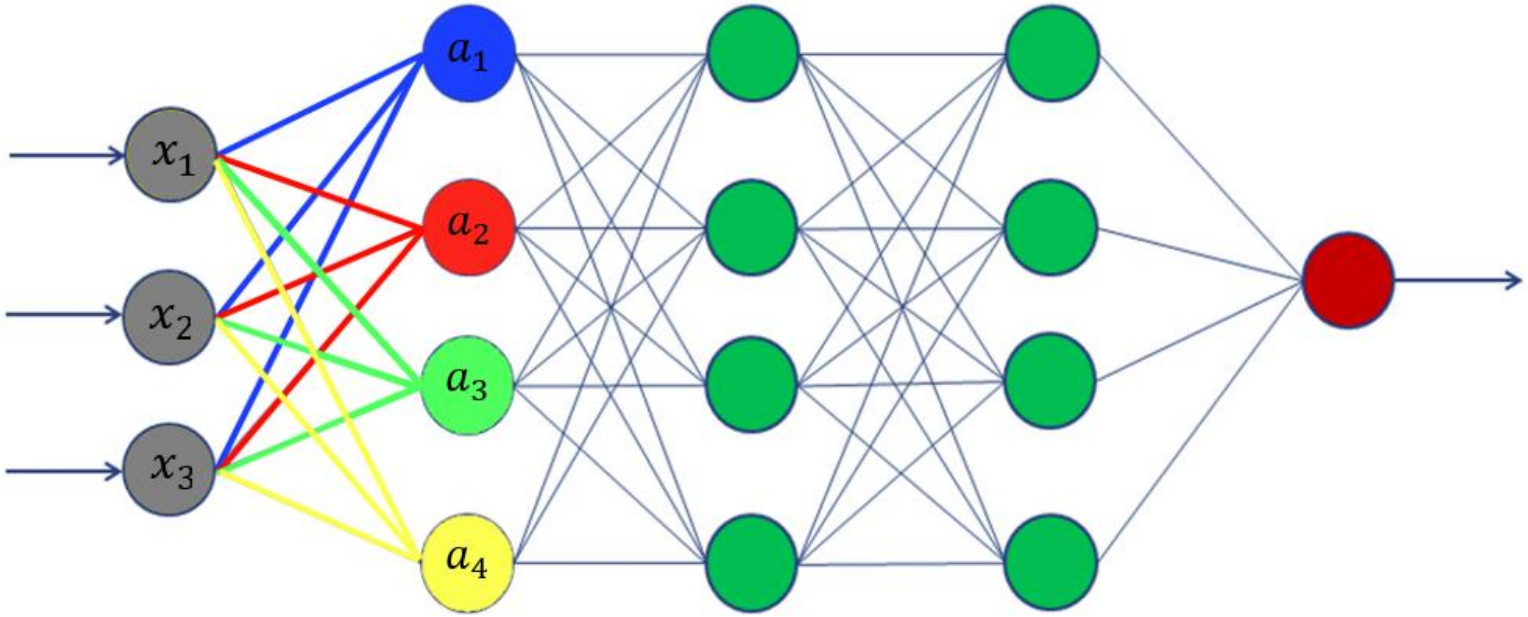
PANDA

Lineaire transformatie gevolgd door niet-lineaire transformatie

1. Gewichtsmatrix
2. Bias vector

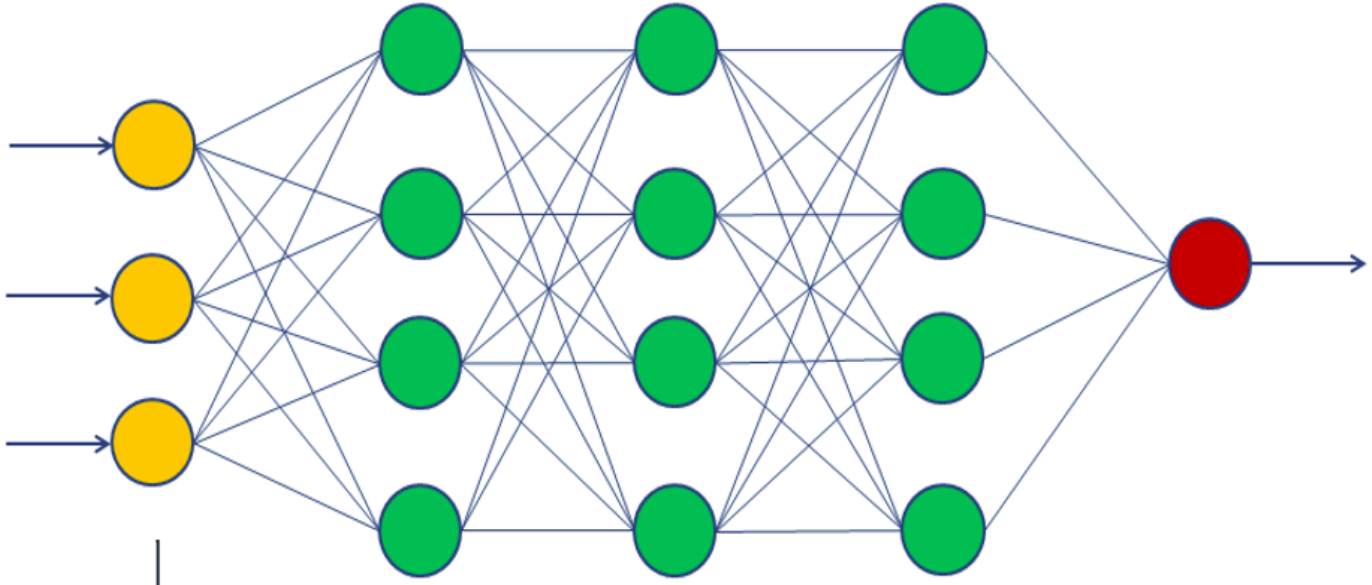
Sigmoid functie: $f(x) = \frac{1}{1+e^{-x}}$

- Input vector: x
- Gewichtsmatrix: W
- Bias vector: b
- Niet-lineaire transf. : f(x)
- Output netwerk: F(x)



$$\begin{bmatrix} w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \\ w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}$$

- Input vector: x
- Gewichtsmatrix: W
- Bias vector: b
- Niet-lineaire transf. : f(x)
- Output netwerk: F(x)



$$x$$

$$f(W^1 x + b^1)$$

$$f(W^2 f(W^1 x + b^1) + b^2)$$

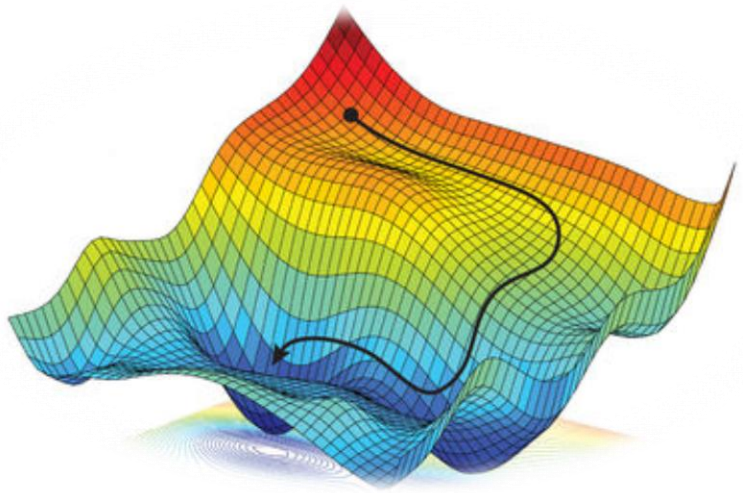
$$F(x) = f(W^3 f(W^2 f(W^1 x + b^1) + b^2) + b^3)$$

Hoe kan het netwerk leren?

- Verwachte output vergelijken met netwerkoutput
- Verschil geeft kostfunctie
- Deze minimaliseren door gewichten en biases aan te passen
- Simpel algoritme: gradient descent

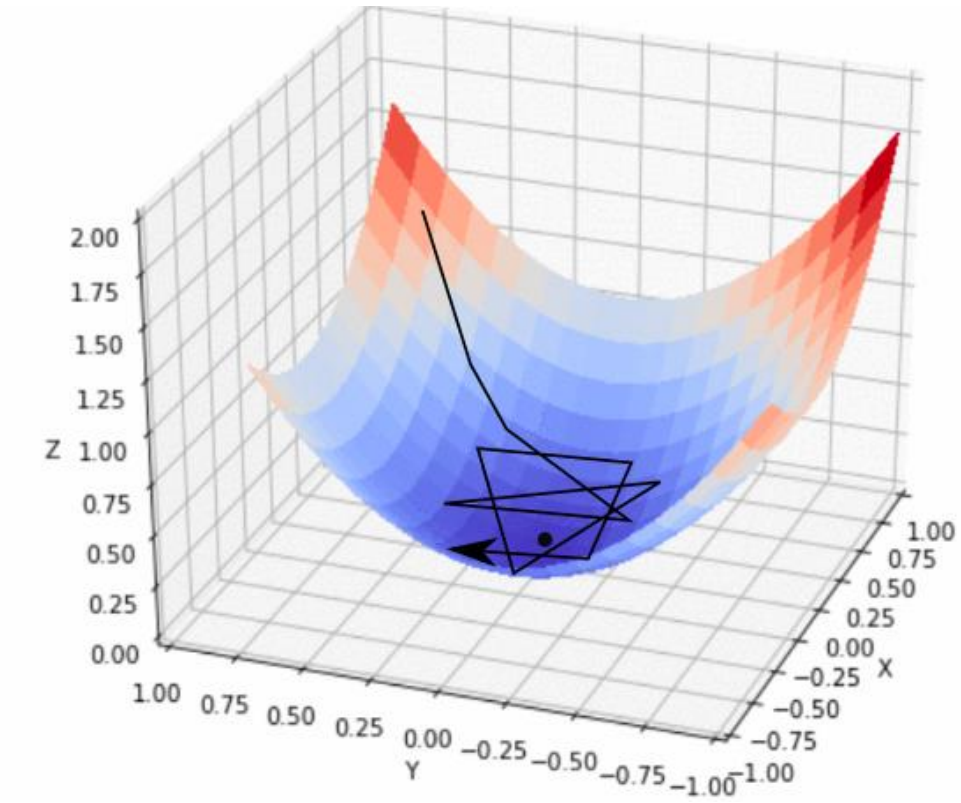
Gradient descent:

- $Cost(p)$
- $p \rightarrow p - \alpha \nabla Cost(p)$



Gradient descent

- $p \rightarrow p - \alpha \nabla Cost(p)$
  1. Stap-grootte: α
  2. Gradiënt van kostfunctie: $\nabla Cost(p)$



- $-\nabla Cost(p)$ is de richting waarin de functie het sterkst daalt
- Stap te groot: mogelijk om over minimum te springen
- Stap te klein: algoritme te traag

Gradient descent

- Mogelijke kostfunctie: $\text{Cost} = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2}\|y(x^{\{i\}}) - a^{[L]}(x^{\{i\}})\|_2^2$

- Datapunten: $x^{\{i\}}$
- Gewenst resultaat: $y(x^{\{i\}})$
- Resultaat model: $a^{[L]}(x^{\{i\}})$

- Deel-functies: $C_{x^{\{i\}}} = \frac{1}{2}\|y(x^{\{i\}}) - a^{[L]}(x^{\{i\}})\|_2^2$

- Berekening gradiënt: $\nabla\text{Cost}(p) = \frac{1}{N}\sum_{i=1}^{N} \nabla C_{x^{\{i\}}}(p)$

Gradient descent

- Groot neuraal netwerk → veel parameters → $\nabla C_{x\{i\}}(p)$ hoge dimensie
- Diepe neurale netwerken vereisen veel datapunten (= grote N)

$$\nabla \text{Cost}(p) = \frac{1}{N} \sum_{i=1}^{N} \nabla C_{x\{i\}}(p)$$

- Oplossing: beperk N in elke iteratie

Stochastic gradient descent

- Telkens 1 punt nemen:

$$p \to p - \eta \nabla C_{x\{i\}}(p)$$

- Telkens m << N punten nemen:
  - Minibatch

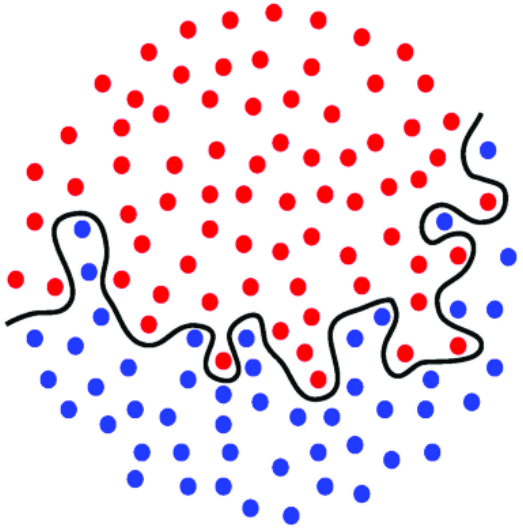
$$p \to p - \eta \frac{1}{m} \sum_{i=1}^{m} \nabla C_{x\{k_i\}}(p)$$

- Met/zonder vervanging

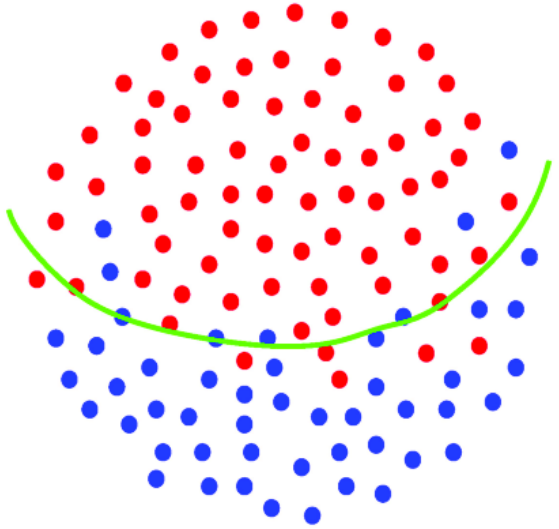
Generaliseren

- Generaliseren = niet eerder geziene datapunten correct labelen
- Doel van het netwerk: goed generaliseren
- Kleine fout op trainingsset ≠ goede generalisatie

Voorbeeld:



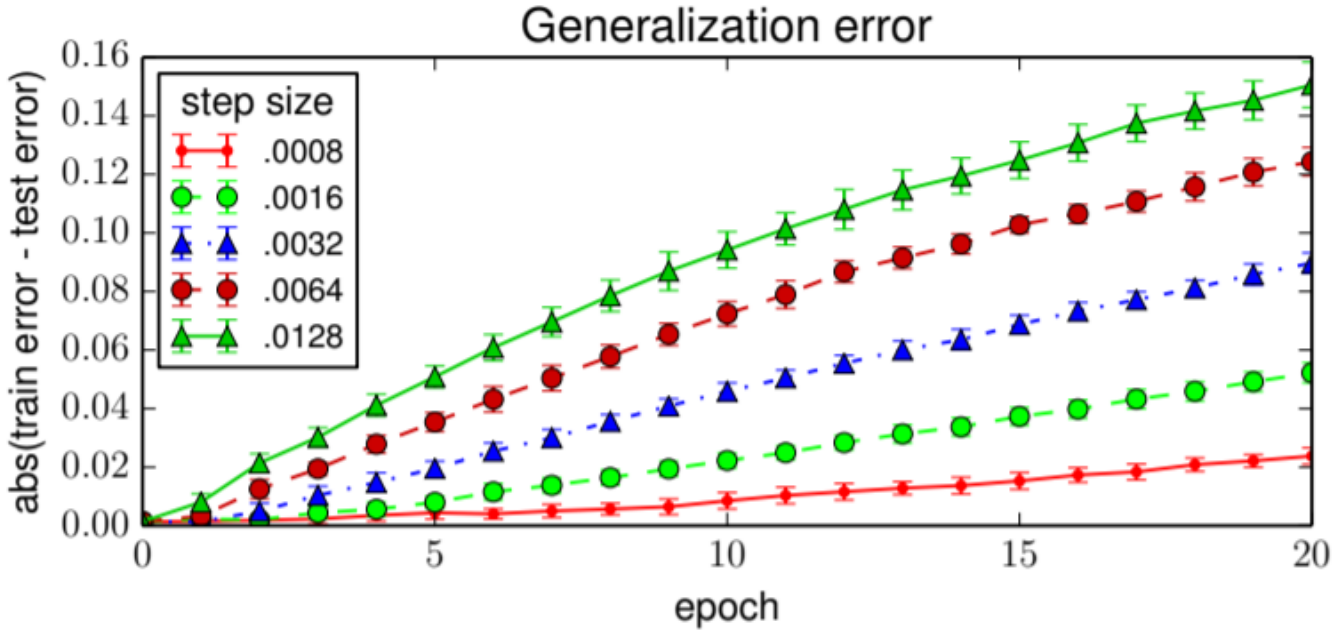
Overfitting



Betere generalisatie

Veel of weinig iteraties?

- Overfitting
- Goede generalisatie
- Training met minder iteraties beter
    1. Snelle convergentie
    2. Indirecte voordelen



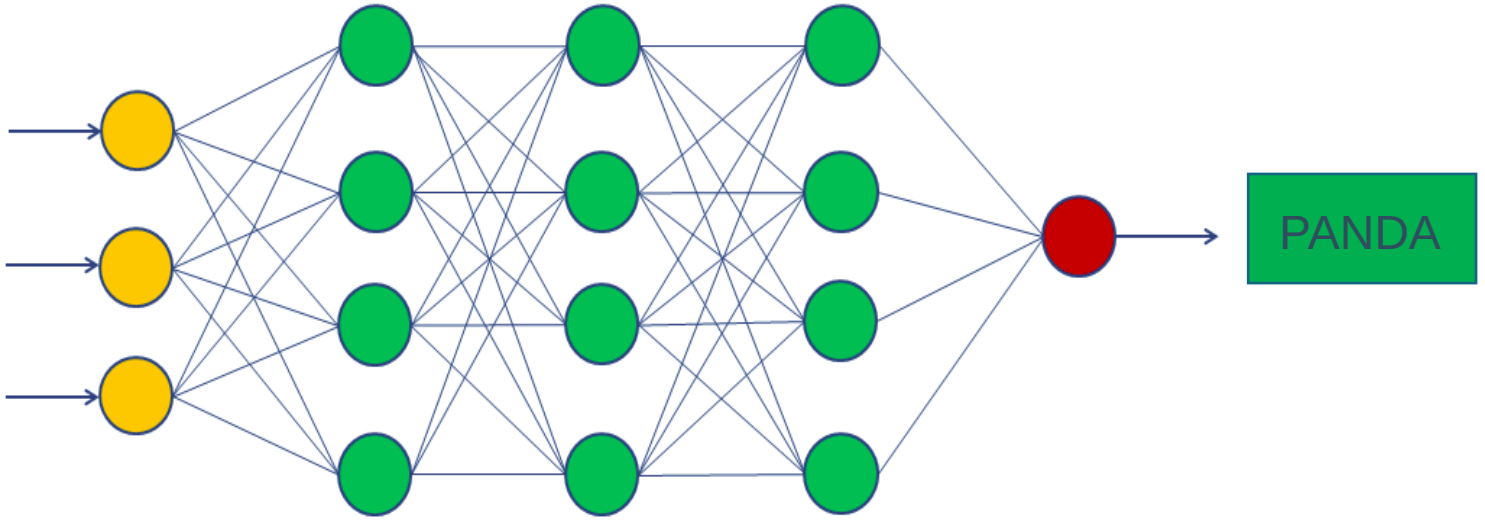
Hangt samen met stabiliteit van SGD

- 2 datasets D1 en D2: Verschillen in 1 punt
- parameters p_1 en p_2: $\|p_1 - p_2\|$

- Stabiliteitsgrenzen bepalen mogelijk
- Afhankelijk van eigenschappen kostfunctie
- Betere stabiliteit -> betere generalisatie error

Het netwerk gebruiken

- Kostfunctie geoptimaliseerd
- Controleren op test data
- Gebruik voor classificatie

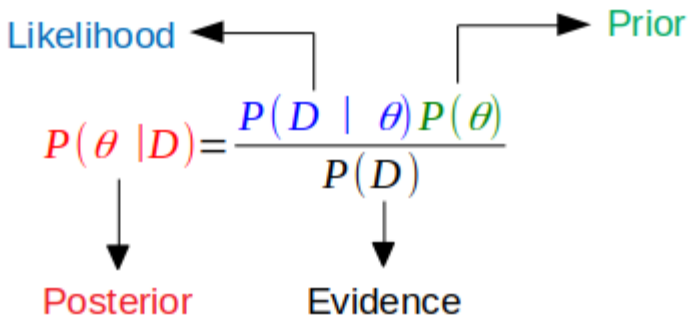


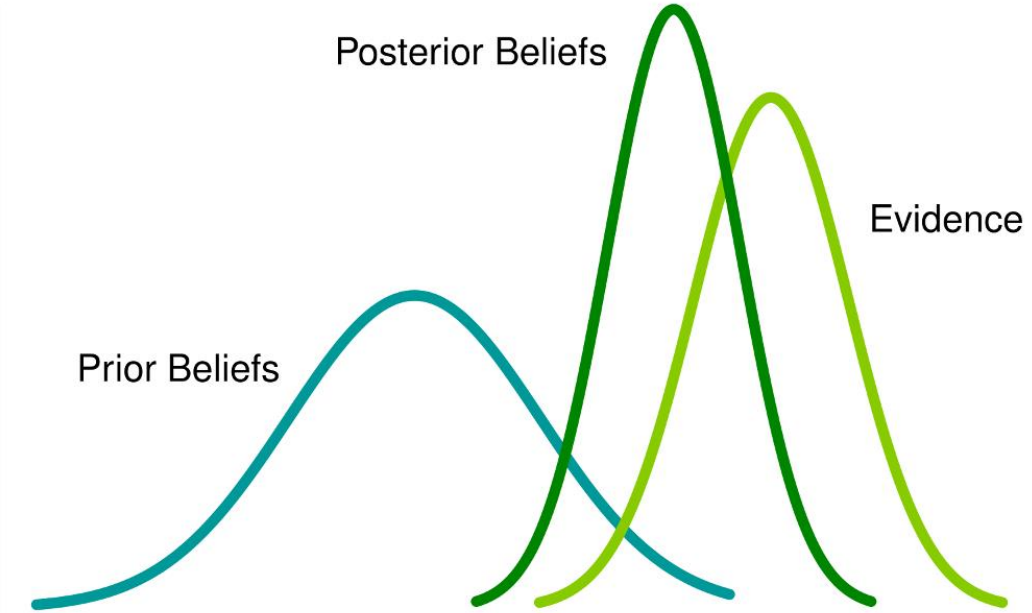
Hoe zeker is het netwerk?

- Moeilijk vast te stellen
- Geen maat voor onzekerheid
- Motivatie om onzekerheid te kwantificeren

Bayesian deep learning

- Stochastisch neuraal netwerk
- Bayesiaanse inferentie



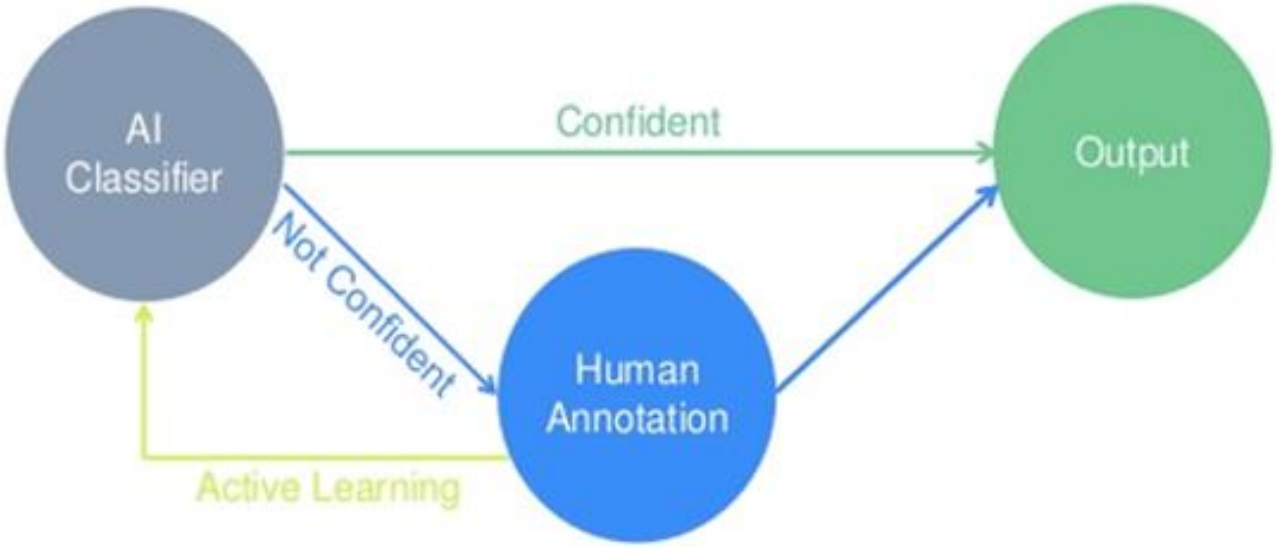


Gradient descent:

- $\mathrm{Cost}(\theta) = -\log\left(p(D|\theta) * p(\theta)\right)$
- $\theta \rightarrow \theta - \alpha\nabla\mathrm{Cost}(\theta)$, α = stap-grootte

Applicaties van Bayesian deep learning

- Onzekerheidskwantificatie
- Zelfrijdende auto’s
- Active learning



KU LEUVEN

Vragen?

Referenties

- C. Higham and D. Higham (2019). Deep Learning: An Introduction for Applied Mathematicians. SIAM Review Vol. 61, No. 4, pp. 860–891

https://epubs.siam.org/doi/pdf/10.1137/18M1165748

- L. Valentin Jopsin, W. Buntine, F. Boussaid, H. Laga, M. Bennamoun (2020), Hands-on Bayesian Neural Networks - a Tutorial for Deep Learning Users

https://arxiv.org/pdf/2007.06823.pdf

- M. Hardt, B. Recht, and Y. Singer, Train faster, generalize better: Stability of stochastic gradient descent, in Proceedings of the 33rd International Conference on Machine Learning, 2016, pp. 1225-1234

http://proceedings.mlr.press/v48/hardt16.pdf