OPTIMALISATIETECHNIEKEN VOOR NEURALE NETWERKEN

bachelorproef

Beunckens Sam
Willio Tuur

UHasselt

Table of contents

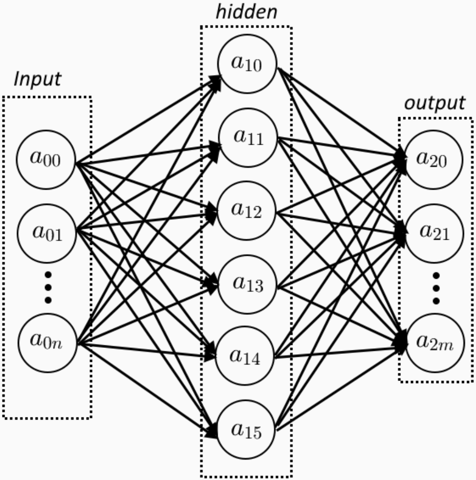
Inleiding

Vragen

- Hoe werken neurale netwerken, als deel van artificiële intelligentie?
- Hoe trainen we neurale netwerken?
- Wat is de invloed van hyperparameters op het trainingsproces?

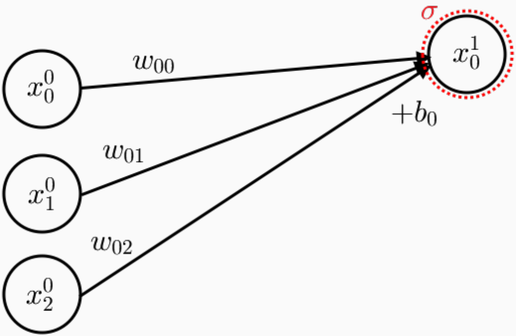
Opbouw van een neuraal netwerk

Architectuur van een neuraal netwerk



$$NN(x_0, x_1, ..., x_n) = (y_0, y_1, ..., y_m) \tag{1}$$

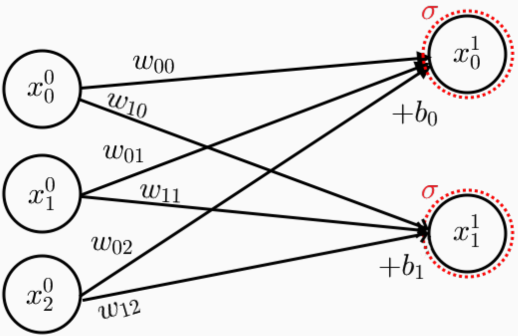
Architectuur van een neuraal netwerk



$$x_0^1 = \sigma\left(\begin{bmatrix} w_{00} & w_{01} & w_{02} \end{bmatrix}\begin{bmatrix} x_0^0 \\ x_1^0 \\ x_2^0 \end{bmatrix} + b_0\right) \tag{2}$$

$\mathbf{w}$ = weights | $\mathbf{b}$ = bias | $\boldsymbol{\sigma}$ = activatiefunctie

Architectuur van een neuraal netwerk



$$\begin{bmatrix} x_0^1 \\ x_1^1 \end{bmatrix} = \sigma\left(\begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \end{bmatrix} \begin{bmatrix} x_0^0 \\ x_1^0 \\ x_2^0 \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \right) \tag{3}$$

$\mathbf{w}$ = weights | $\mathbf{b}$ = bias | $\boldsymbol{\sigma}$ = activatiefunctie (elementsgewijs)

Loss van een netwerk

Manier om performantie te karakteriseren

Loss functie voor een configuratie parameters:

$$C(Y, NN(X)) = \sum_{i=1}^{n} (y_i - NN(x_i))^2 \tag{4}$$

$$Y = f(X)$$

Bijwerken van parameters

Corrigeren van parameters op basis van de loss

Gradiënt descent methode:

Zij C continu differentieerbaar op zijn domein, met startpunt $x^{(0)}$
Dan geldt:

$$x^{(k)} = x^{(k-1)} - \eta \nabla C(Y, NN_{x^{(k-1)}}) \tag{5}$$

Hierbij is η de learning rate

Bijwerken van parameters

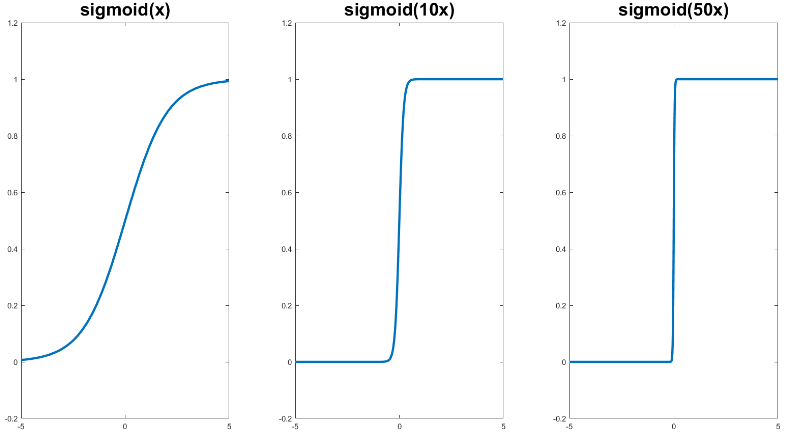
- Stochastische gradiënt descent

  1 willekeurig trainingsdatapunt i.p.v. volledige dataset bij een iteratie

  Efficiëntere daling met meer ruis.

Universele Approximatie Stelling

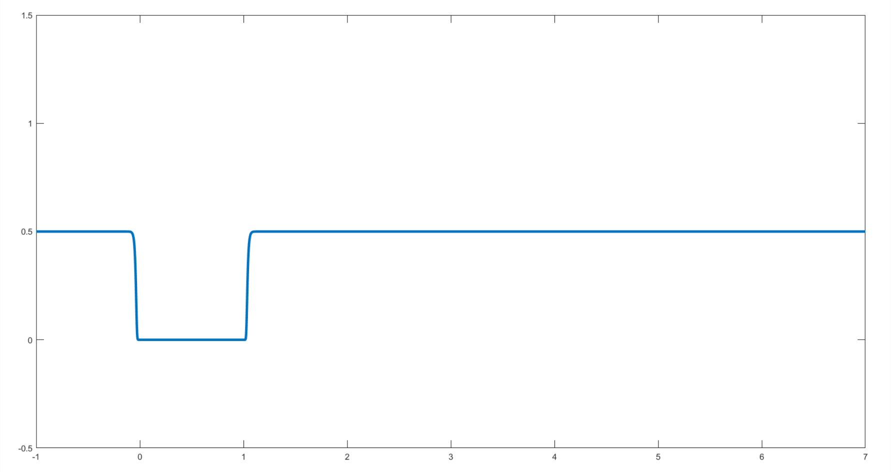
Sigmoidale Functies



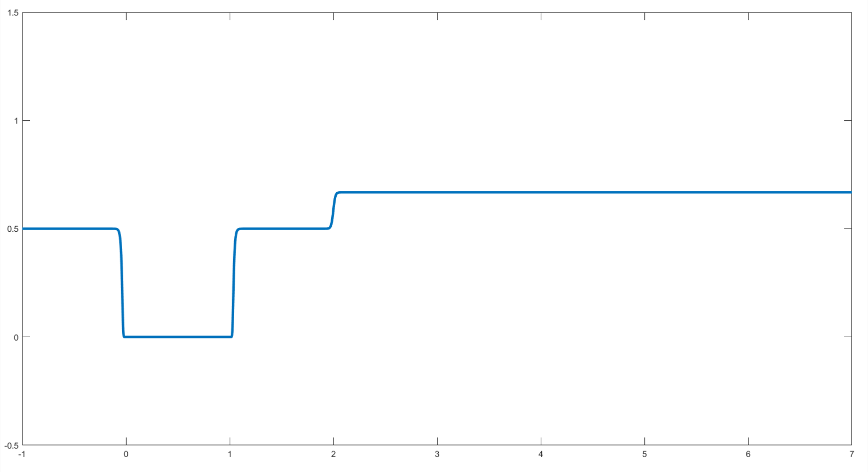
$$w \longrightarrow +\infty, \qquad \text{in } \sigma(wx)$$

$$\forall x < 0, \forall \epsilon_w > 0, \exists w_0 > 0, \forall w \geq w_0 : \ |\sigma(wx)| < \epsilon_w$$

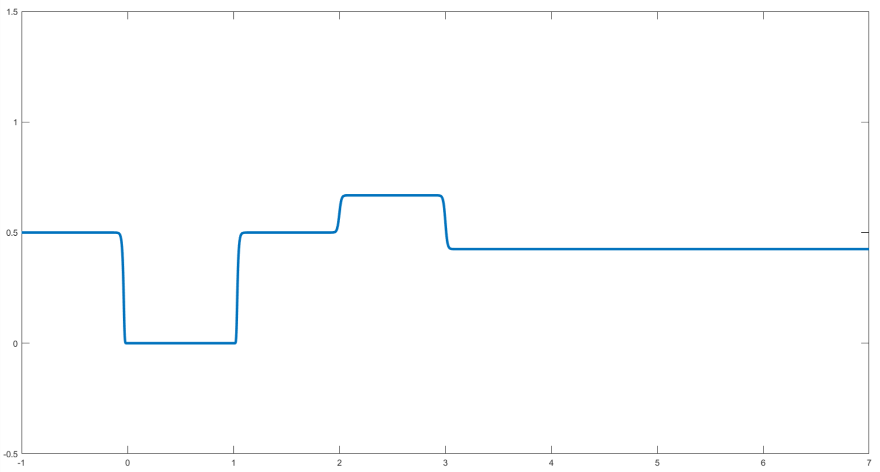
Sigmoidale Functies (2 nodes)



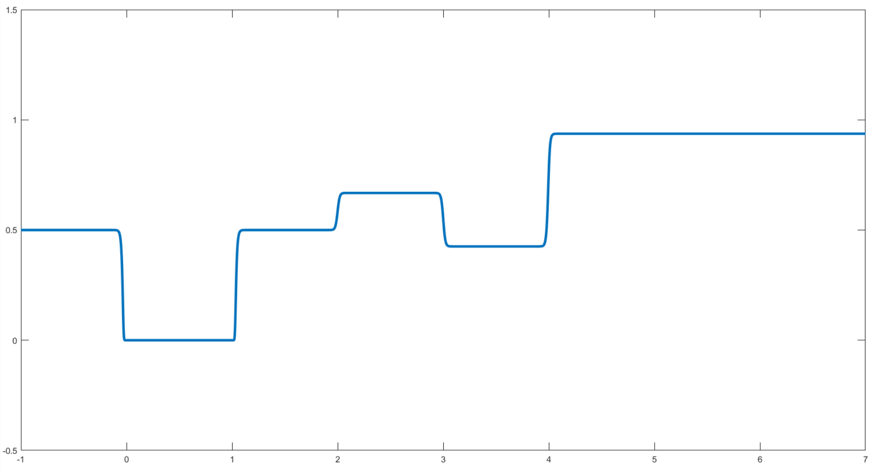
Sigmoidale Functies (3 nodes)



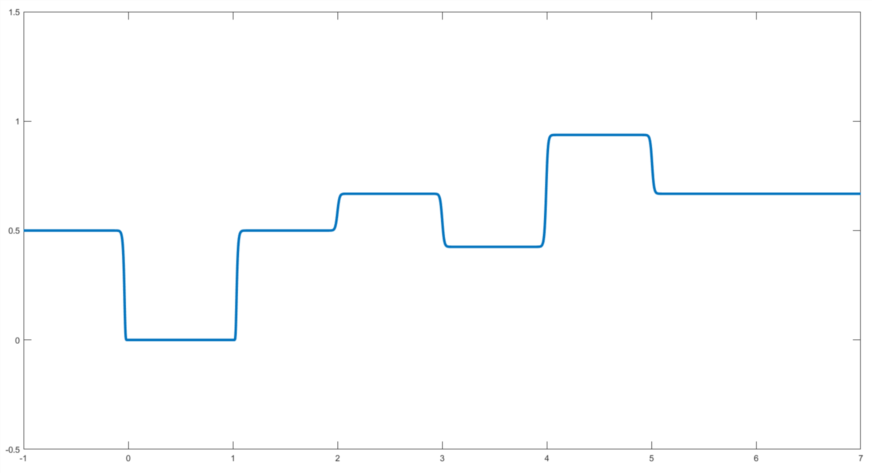
Sigmoidale Functies (4 nodes)



Sigmoidale Functies (5 nodes)

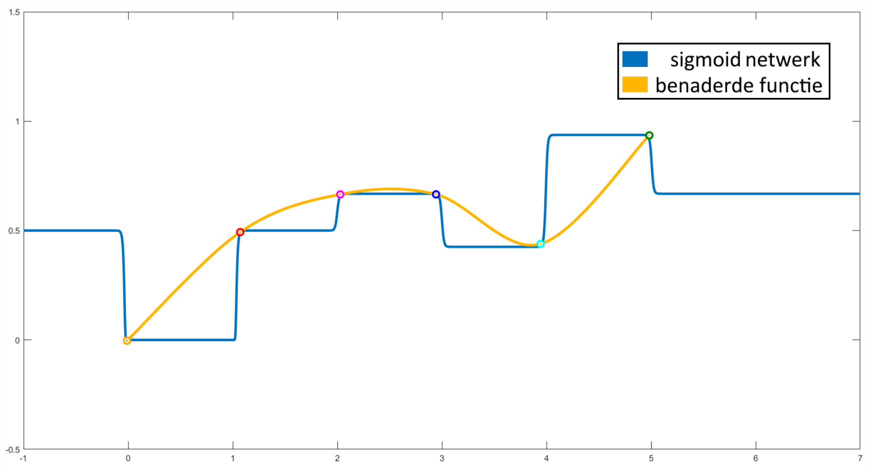


Sigmoidale Functies (6 nodes)



Opmerking: domein is vast normaal $\Rightarrow$ breedte van trappen worden kleiner, $\frac{b-a}{n}$ voor $[a, b]$

Sigmoidale Functies

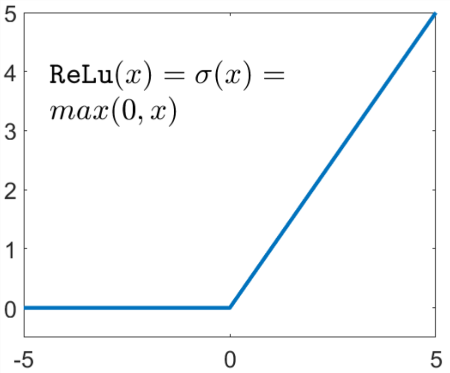


$$\epsilon = \frac{L}{n}(b-a) + \epsilon_w, \qquad L \text{ Lipschitz constante van } f$$

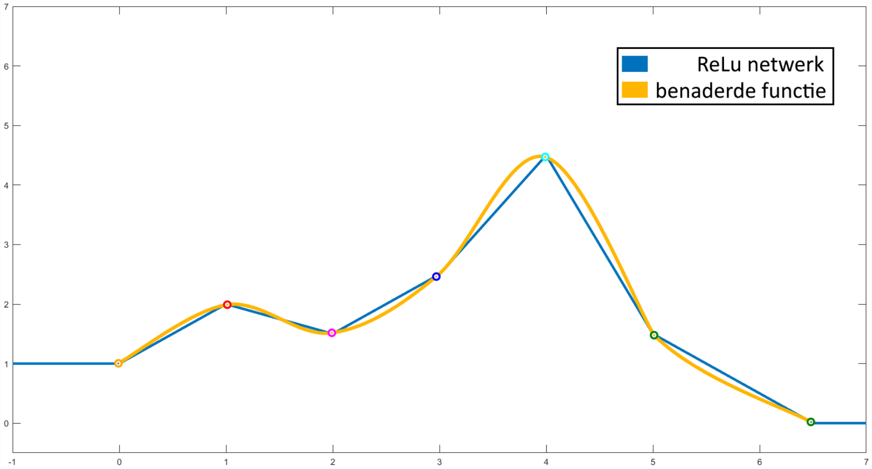
[9] M. Nielsen. Neural networks and deep learning.

Algemene functie (ReLu)

- Stuksgewijs C^2
- $\exists M \in \mathbb{R}: \ x \geq M \vee x \leq M: \sigma(x) = 0$
- $supp(\sigma) \neq \varnothing$
- $f|_{supp(f)}$ injectief

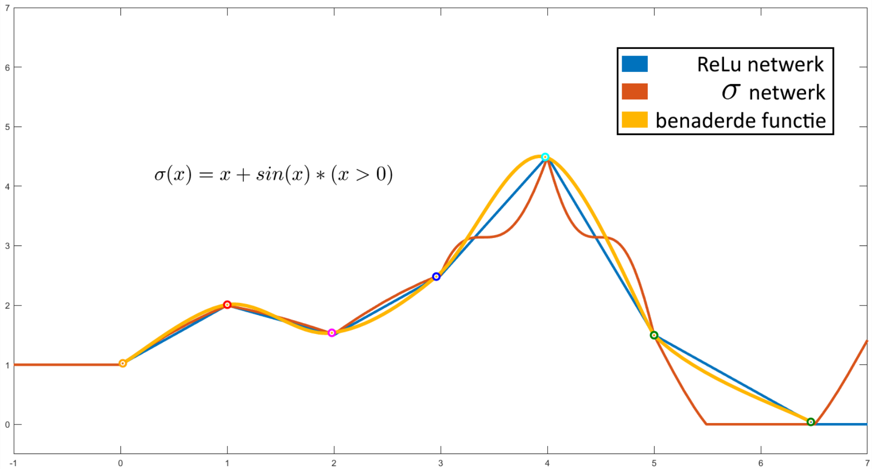


Algemene functie (ReLu) (6 nodes)



$$\epsilon = \frac{|f''(\xi)|}{2n^2}(b-a)^2, \qquad \xi \in [a, b]$$

Algemene functie (ReLu) (6 nodes)



$$\epsilon = \frac{|f''(\xi_0)| + |NN''(\xi_1)|}{2n^2}(b-a)^2, \qquad \xi_{0,1} \in [a, b]$$

Meerdere lagen

Stappenplan:

- $f := \sigma^{-1}(a \cdot f + b)$, zodat $f([a, b])$ in $supp(\sigma)$
  $\Rightarrow \sum_{k=1}^{m} w_k \cdot NN_k(x) = f$

- $w_k = \frac{f^{(k)}(c)}{k!}$, een c in $[a, b]$
- $NN_k(x) \longrightarrow (x - c)^k$
  $\Rightarrow$ Taylor polynoom, orde = aantal nodes,
  $f_m = \sum_{k=1}^{m} \frac{f^{(k)}(c)}{k!} \cdot (x - c)^k$

Meerdere lagen

Met inductie, voor 2 lagen:

$$\begin{aligned} |NN_2(x) - f(x)| &= |NN_2(x) - f_m(x) + f_m(x) - f(x)| \\ &\leq |f_m(x) - f(x)| + |NN_2(x) - f_m(x)| \\ &\leq err1 + |\epsilon_1 \sum_{k=0}^{m} \frac{f^{(k)}(c)}{k!}| \\ &\leq err1 + \epsilon_1(|f(c+1)| + err2) \end{aligned}$$

Met err Taylor error $= \frac{|f^{(k+1)}(\zeta_0)|}{(k+1)!}|x - c|^{k+1} \qquad \zeta_0 \in [x, c] \vee [c, x]$
ALLE nodes naar $+\infty$, ook $\epsilon_1 \to 0$, fout naar nul
Inductiestap $\epsilon_1 = |NN_2(x) - f(x)|$,...

Meerdere lagen

Methode 1 laag, met ϵ_{m_1}
Fout voor $\mathcal{L}$ lagen:

$$||NN_{\mathcal{L}} - f||_{L^2} \leq$$

$$\sqrt{b-a} \cdot \left(\sum_{i=2,\, \mathcal{L}>1}^{\mathcal{L}} (R^0_{m_i} \cdot \prod_{k=i+1,\, i<\mathcal{L}}^{\mathcal{L}} (C + R^1_{m_k})) + \epsilon_{m_1} \prod_{i=2,\, \mathcal{L}>1}^{\mathcal{L}} (C + R^1_{m_i}) \right)$$

$$R^0_m := \frac{|f^{(m+1)}(\zeta)|}{(m+1)!}|x-c|^{m+1} \qquad \text{voor } \zeta \in [x,c] \vee [c,x]$$

$$R^1_m := \frac{|f^{(m+1)}(\zeta)|}{(m+1)!} \qquad \text{voor } \zeta \in [c, c+1]$$

$$C := |f(c+1)|$$

Optimalisatie van hyperparameters

Hyperparameters

Parameters die te kiezen zijn voor het trainingsprogramma:

- Topologie van het netwerk
- Learning rate
- Activatie functies
- .....

Θ is de hyperparameterparameterruimte:
Zoek $\theta = \underset{\hat{\theta} \in \Theta}{\operatorname{argmin}} \, \mathrm{f}(\hat{\theta})$

methoden:

Grid search:

- Verdeel Θ op in een rooster
- Voer een (gedeeltelijke) training uit in elk punt

Random search:

- Selecteer een willekeurig punt uit Θ
- Voer een (gedeeltelijke) training uit
- Herhaal tot stopcriteria voldaan is

Methode: Bayesiaanse optimalisatie

Methode om 'black box' functies te optimaliseren

Vergt geen kennis over onderliggende structuur functie

Werkt iteratief:
1) Stel een model op met aantal beginpunten
2) Bepaal welk punt men vervolgens opneemt in het model
3) Werk het model bij

Het model opstellen

Men zal een model opstellen m.b.v. **Gaussian Process regressie**
Beschouw: $\mathcal{D} = \{(\mathbf{x}, f(\mathbf{x})|\mathbf{x} \in \mathbb{R}^n, f(\mathbf{x}) \in \mathbb{R}^m\}$

Distributie over functies

Elke eindige verzameling van punten is Multivariaat normaal verdeelt

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{6}$$

Met:

$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ is de mean functie
$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x} - m(\mathbf{x})))(f(\mathbf{x}') - m(\mathbf{x}'))]$ is de covariantiefunctie

Het model opstellen

$X^\star$ regressie inputwaarden z.d. $\mathbf{f}^\star = f(X^\star)$

$$\mathbf{f}^\star \sim \mathcal{N}(0, K(X^\star, X^\star)) \tag{7}$$

A priori distributie (zonder gebruik te maken van $\mathcal{D}$)
Men zoekt nu : $\mathbf{f}^\star \mid X^\star, X, f(X)$

Het model opstellen

Het resultaat: $\mathbf{f}^\star \mid X^\star, X, \mathbf{f} \sim \mathcal{N}\left(m(\mathbf{x}), \sigma(\mathbf{x})^2\right)$ met:

$$m(\mathbf{x}) = \left(K(X^\star, X)K(X, X)\right)^{-1}\mathbf{f}$$
$$\sigma(\mathbf{x})^2 = \left(K(X^\star, X^\star) - K(X^\star, X)K(X, X)\right)^{-1}K(X, X^\star)$$

acquisitie functie

Model is opgesteld, Hoe kiest men het volgende datapunt?

Acquisitie functies: Eenvoudig het maximum te vinden

Maxima $\implies$ interessante input waarde

Verschillende types hebben verschillende doelen
Exploitation versus exploration

Probability of improvement:

Wat is de kans op nog extremere waarden?

Beschouw $I(\mathbf{x}) = \max(f(\mathbf{x}) - f^\star(\mathbf{x}), 0)$ met:

$P(I(\mathbf{x}) > 0) = 1 - \Phi\left(\frac{f(\mathbf{x})^\star - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right)$

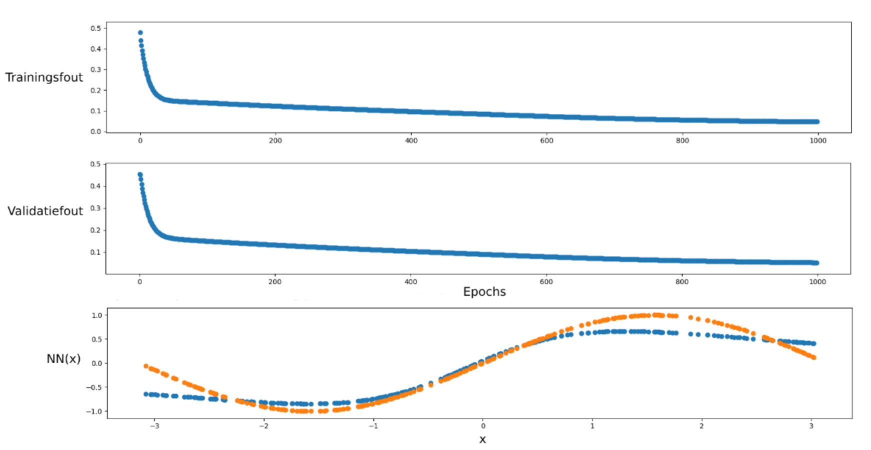
Expected improvement:

Welke verbetering verwachten we?

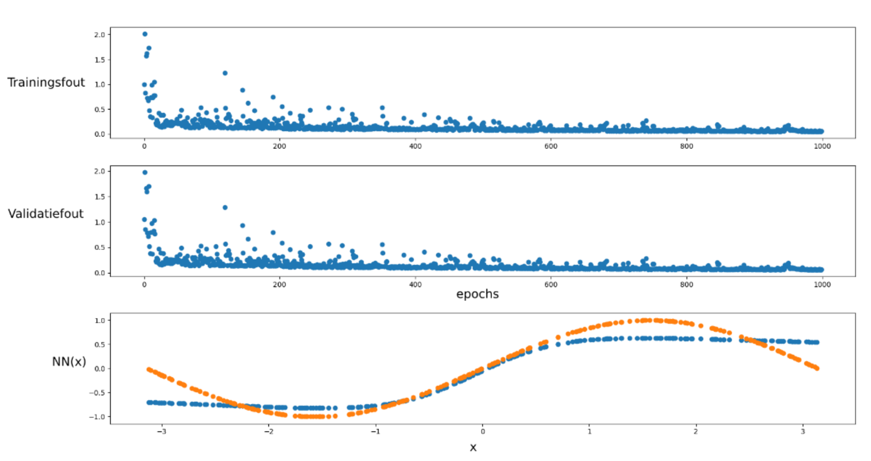
$$\mathbb{E}\left[I(x)\right] = (\mu(\mathbf{x}) - f(\mathbf{x}^\star))\Phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^\star)}{\sigma(\mathbf{x})}\right) + \sigma(\mathbf{x})\varphi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^\star)}{\sigma}\right)$$

Rekenresultaten

Gradient Descent

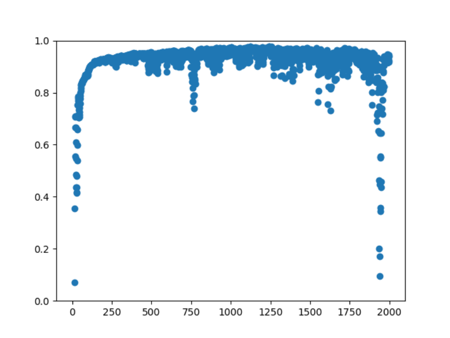


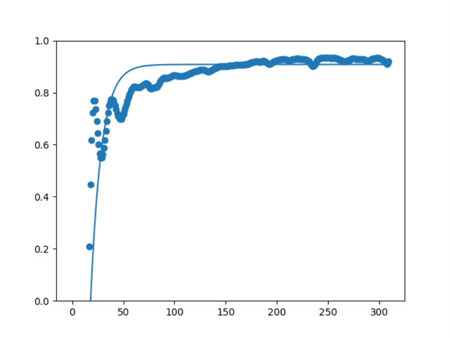
Stochastische Gradient Descent



Performance grafiek

R^2 over aantal iteraties





$$\frac{dP(t)}{dt} = k \cdot (M - P(t))$$

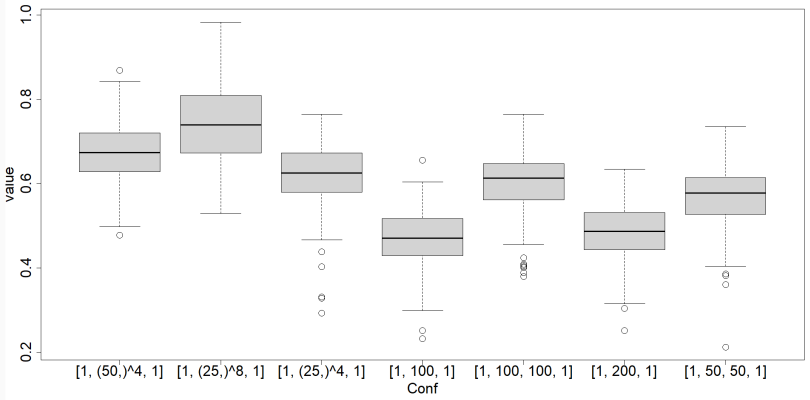
Topologie van neuraal netwerk

$f(x) = sin(x) + cos(5x)$,
200 netwerken, 250 epochs, 32 mini-batch grootte, 100 trainings- 100 validatiepunten

- [1,200,1]
- [1, 100,1]
- [1,100,100,1]
- [1,50,50,1]
- [1,25,25,25,25,1]
- [1,50,50,50,50,1]
- [1,$(25)^8$,1]

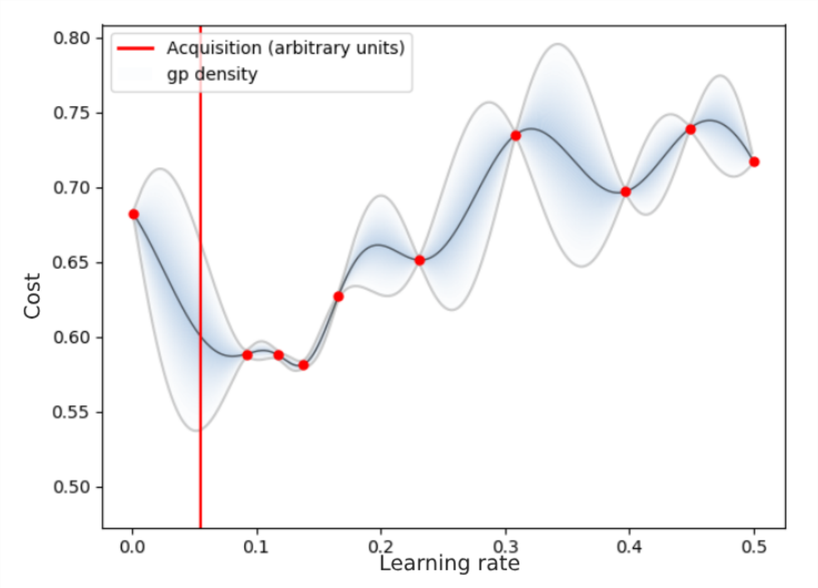
paarsgewijs t-testen met Bonferroni correctie

Topologie van neuraal netwerk

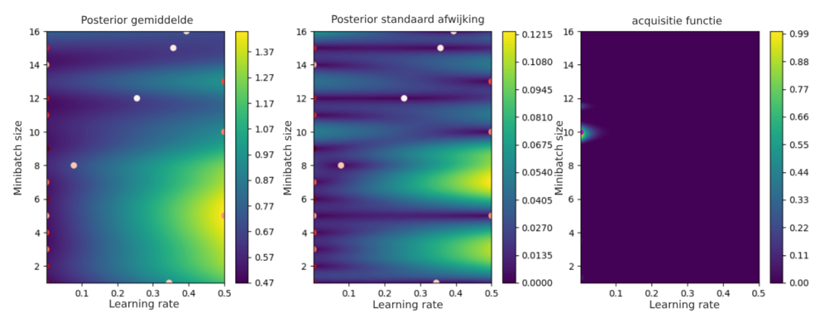


- `[1, 200, 1]` met `[1, 100, 1]` ($p = 0.43$)
- `[1, 25, 25, 25 25, 1]` met `[1, 100, 100, 1]` ($p = 0.19$)

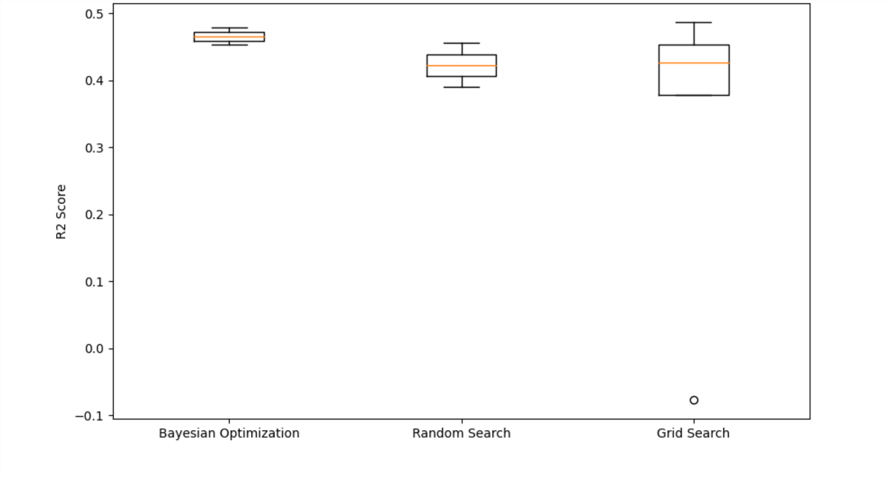
Optimalisatie van hyperparameters



Optimalisatie van hyperparameters



Vergelijking methoden



Vervolgonderzoek

References i

R. A. Adams and C. Essex.
Calculus a complete Course, 10th Edition.
Pearson, 2022.

A. Y. C. akmak.
Universal Approximation Theorem.
Istanbul Technical University, 2022.

J. Bergstra, R. Bardenet, B. Kégl, and Y. Bengio.
Algorithms for hyper-parameter optimization.
12 2011.

V. M. C. Eric Brochu and N. de Freitas.
A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning.
2020.

T. Janssens.
Hyperparameter tuning for Artificial Neural Networks applied to inverse mapping parameter updating.
Eindhoven University of Technology, 2022.

D. P. Kingma and J. L. Ba.
ADAM: A method for stochastic optimization.
University of Amsterdam, OpenAI and University of Toronto, 2017.

A. Kratsios.
The Universal Approximation Property.
McMaster University, 2020.

Y. Liang and Z. Shi.
Lecture 4 Approximation II.
University of Wisconsin–Madison, 2022.

References iii

M. Nielsen.
Neural networks and deep learning.
http://neuralnetworksanddeeplearning.com/, 2019.

I. Panageas and G. Piliouras.
Gradient Descent Only Converges to Minimizers: Non-Isolated Critical Points and Invariant Regions.
Georgia Institute of Technology and Singapore University of Technology Design, 2016.

E. Parzen.
On Estimation of a Probability Density Function and Mode.
The Annals of Mathematical Statistics, 33(3):1065 – 1076, 1962.

C. Rasmussen, O. Bousquet, U. Luxburg, and G. Rätsch.
Gaussian processes in machine learning.
Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures, 63-71 (2004), 3176, 09 2004.

M. Schmidt.
CPSC 540: Machine Learning, Convergence of Gradient Descent.
University of British Columbia, 2017.

S. Shalev-Shwartz and S. Ben-David.
Understanding Machine Learning.
Cambridge University Press, The Hebrew University, Jerusalem and University of Waterloo, Canada, 2019.

References v

G. STRANG.
Linear Algebra and Learning from Data.
Wellesley-Cambridge Press, Massachusetts Institute of Technology, 2019.

Vragen?

Sigmoidale Functies

- $\lim_{x \to -\infty} \sigma(x) = 0$ en $\lim_{x \to +\infty} \sigma(x) = c$ voor een $c \in \mathbb{R}$, sigmoid $c = 1$
- σ is continu op $\mathbb{R}$
- $\sigma'(x) \geq 0 \quad \forall x \in \mathbb{R}$
- $\lim_{x \to \infty} x\sigma'(x) = 0$

Meerdere lagen

gegeven:

- methode benaderen met 1 laag
- $f \in C^\infty$
- $\exists a, b \in \mathbb{R}: \; Img(a \cdot f([a, b]) + b) \subseteq supp(\sigma)$
- $\sigma^{-1} : \sigma(supp(\sigma)) \rightarrow supp(\sigma)$ goed gedefinieerd
- node in laag i is neuraal netwerk met $i - 1$ lagen
- neuraal netwerk $\mathcal{L}$ lagen $\equiv$ activatiefunctie van gewogen som neurale netwerken $\mathcal{L} - 1$ lagen

alternatief op Gaussian processes

Tree structured Parzen Estimator

- Geen veronderstelling van onderliggende distributie
- Maakt gebruik van parzen Window estimation

I.P.V. $f|X$ rechtstreeks te beschouwen: $x|f$

Tree structured Parzen Estimator

verdeel $\mathcal{D}$ in 2 verzamelingen:
$f(\mathbf{x}) < \mathbf{y}^\star$ en $f(\mathbf{x}) \geq \mathbf{y}^\star$
$\mathbf{y}^\star$ is hier het γde percentiel, te kiezen
stel $p(\mathbf{x}|\mathbf{y} = f(\mathbf{x}))$:

$$p(\mathbf{x}|\mathbf{y}) = \begin{cases} l(\mathbf{x}) \text{ als } f(\mathbf{x}) < \mathbf{y}^\star \\ g(\mathbf{x}) \text{ als } f(\mathbf{x}) \geq \mathbf{y}^\star \end{cases} \tag{8}$$

z.d.:$p(\mathbf{x}) = \gamma \cdot l(\mathbf{x}) + (1 - \gamma) \cdot g(\mathbf{x})$

Tree structured Parzen Estimator

M.b.v. het theorema van bayes:

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{x}|f(\mathbf{x})) \cdot p(f(\mathbf{x}))}{p(\mathbf{x})} \tag{9}$$

Men beschouwt opnieuw: $\mathbb{E}(I(x))$
Deze uitdrukking zal leiden tot:

$$\mathbb{E}(I(x)) = \frac{\int_{y^\star}^{\infty} y \cdot p(y) dy - y^\star \cdot (1-\gamma)}{\gamma \cdot \frac{l(x)}{g(x)} + (1-\gamma)}$$

Tree structured Parzen Estimator

$\mathbb{E}(l(x)) \propto (\gamma \cdot \frac{l(x)}{g(x)} + (1-\gamma))^{-1}$
Men zoekt dus eigenlijk:

> punten met een hoge kans om onder $g(x)$ te liggen
>
> punten met een lage kans om onder $l(x)$ te liggen