PSY9511: Seminar 7

Deep learning for computer vision tasks

Esten H. Leonardsen

07.11.24

UNIVERSITY OF OSLO

Outline

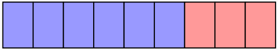
1. Exercise 4
2. Deep learning
    - Motivation
    - (Deep) neural networks
    - Training procedure
3. Convolutional neural networks for computer vision

Weekly exercises

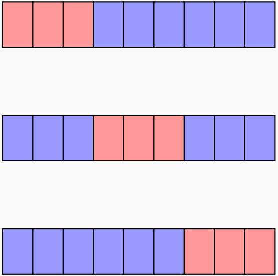
- The weekly exercises are **mandatory**
- The deadlines are **strict**

Validation procedures

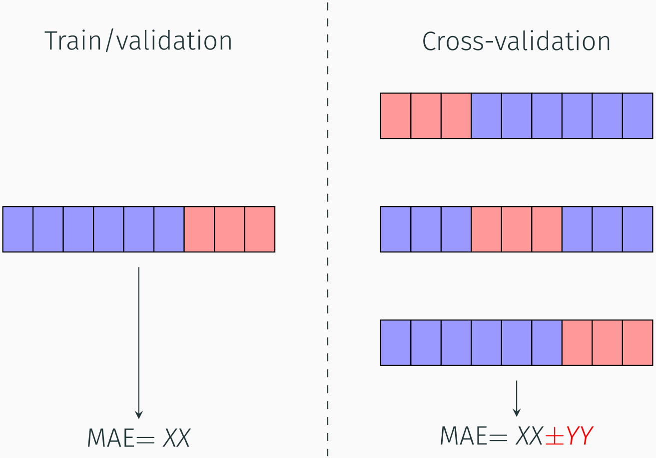
Train/validation



Cross-validation



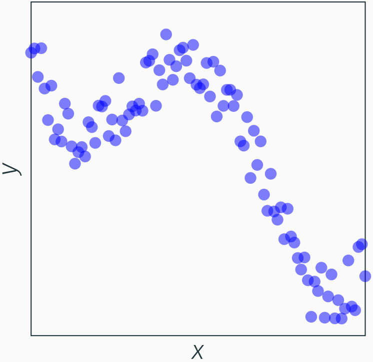
Validation procedures



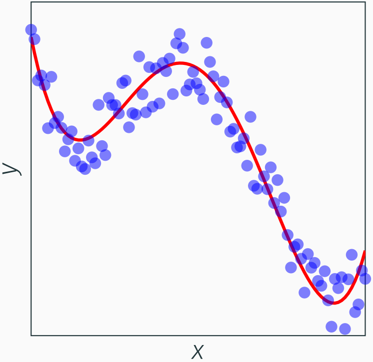
Deep learning

UNIVERSITY
OF OSLO

Deep learning: Motivation

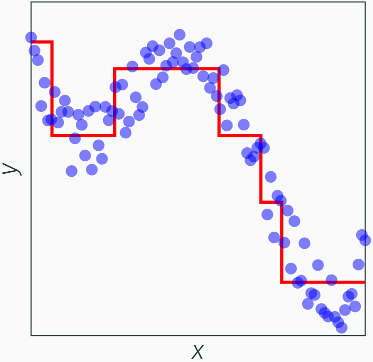


Deep learning: Motivation



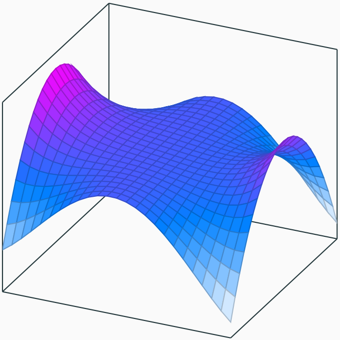
$$\hat{y} = s(x)$$

Deep learning: Motivation

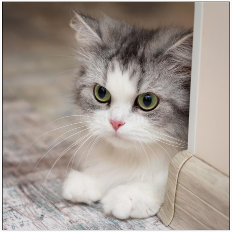


$$\hat{y} = \begin{cases} 4 & \cdots \\ 3 & 0.2 \leq x < 0.6 \\ 1.5 & \cdots \end{cases}$$

Deep learning: Motivation

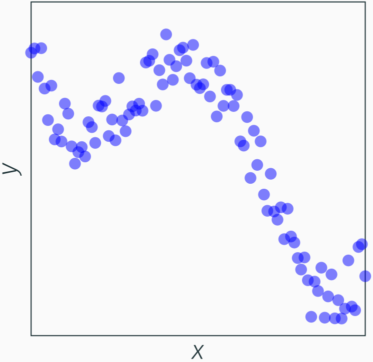


Deep learning: Motivation

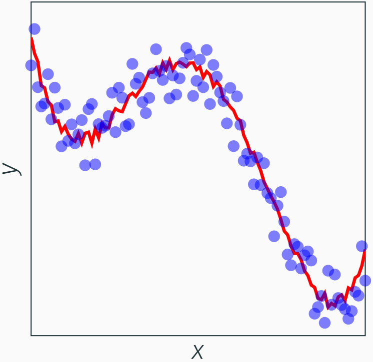


The cat wagged its tail

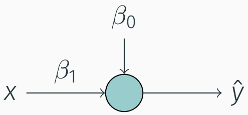
Deep learning: Motivation



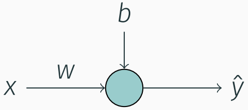
Deep learning: Motivation



Deep learning: Artificial neural networks

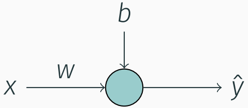


$$\hat{y} = \beta_0 + \beta_1 x$$

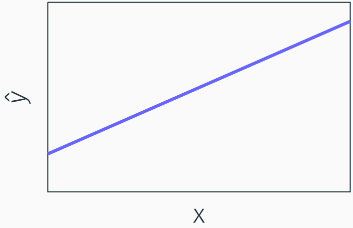


$$\hat{y} = wx + b$$

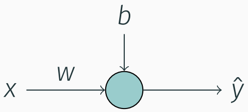
Deep learning: Artificial neural networks



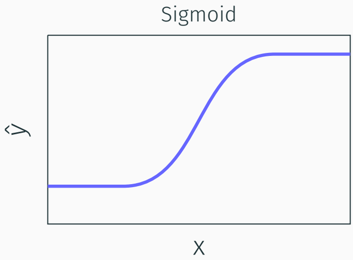
$$\hat{y} = wx + b$$



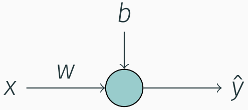
Deep learning: Artificial neural networks



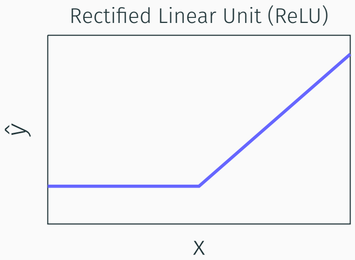
$$\hat{y} = \frac{e^x}{1 + e^x}$$



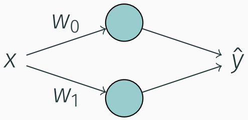
Deep learning: Artificial neural networks



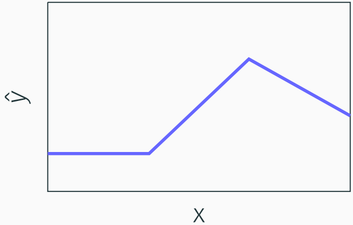
$$\hat{y} = max(0, wx + b)$$



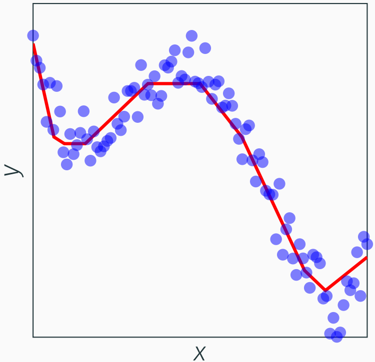
Deep learning: Artificial neural networks



$$\hat{y} = max(0, w_0 x + b_0) + max(0, w_1 x + b_1)$$



Deep learning: Artificial neural networks

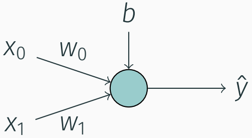


Piecewise linear function

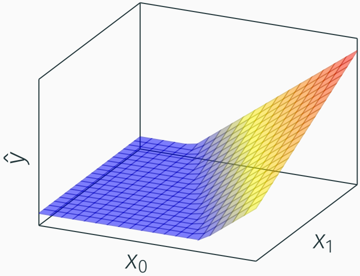
Universal approximation theorem:

”Any relationship that can be described with a polynomial function can be approximated by a neural network with a single hidden layer.”

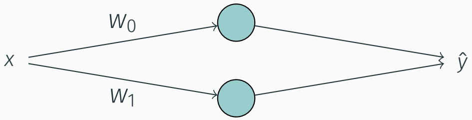
Deep learning: Artificial neural networks



$$\hat{y} = max(0, w_0x_0 + w_1x_1 + b)$$

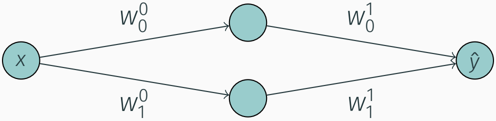


Deep learning: Artificial neural networks



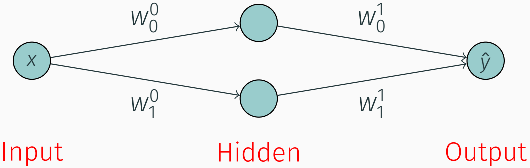
$$\hat{y} = max(0, w_0 x + b_0) + max(0, w_1 x + b_1)$$

Deep learning: Artificial neural networks



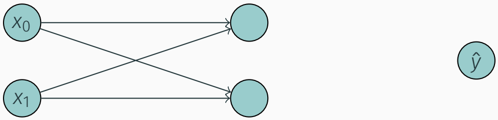
$$\hat{y} = max(0, w_0^1 * max(0, w_0^0 * x + b_0^0) + w_1^1 * max(0, w_1^0 * x + b_1^0) + b_0^1)$$

Deep learning: Artificial neural networks



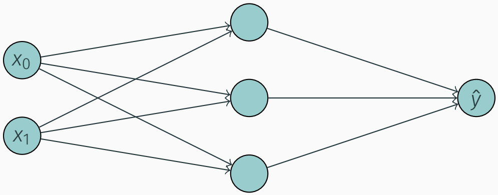
$$\hat{y} = max(0, w_0^1 * max(0, w_0^0 * x + b_0^0) + w_1^1 * max(0, w_1^0 * x + b_1^0) + b_0^1)$$

Deep learning: Artificial neural networks



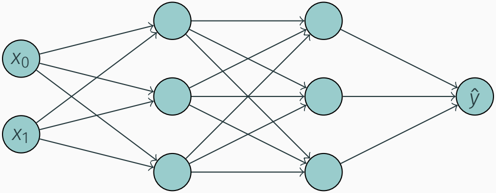
$$\hat{y} = max(0, w^1_{0,0} * max(0, w^0_{0,0} * x_0 + w^0_{1,0} * x_1 + b_{0,0}) + w^1_{1,0} * max(0, w^0_{0,1} * x_0 + w^0_{1,1} * x_1 + b_{0,1}) + b_1)$$

Deep learning: Artificial neural networks



$$\begin{aligned}\hat{y} = max(0, & w^1_{0,0} * max(0, w^0_{0,0} * x_0 + w^0_{1,0} * x_1 + b_{0,0}) + \\ & w^1_{1,0} * max(0, w^0_{0,1} * x_0 + w^0_{1,1} * x_1 + b_{0,1}) + \\ & w^1_{2,0} * max(0, w^0_{0,2} * x_0 + w^0_{1,2} * x_1 + b_{0,2}) + \\ & b_1)\end{aligned}$$

Deep learning: Artificial neural networks



$$
\begin{aligned}
\hat{y} = max(0, & w^2_{0,0} * max(0, w^1_{0,0} * max(0, w^0_{0,0} * x_0 + w^0_{1,0} * x_1 + b_{0,0}) + \\
& \qquad w^1_{1,0} * max(0, w^0_{0,1} * x_0 + w^0_{1,1} * x_1 + b_{0,1}) + \\
& \qquad w^1_{2,0} * max(0, w^0_{0,2} * x_0 + w^0_{1,2} * x_1 + b_{0,2}) + \\
& \qquad b_{1,0}) + \\
& w^2_{1,0} * max(0, w^1_{0,1} * max(0, w^0_{0,0} * x_0 + w^0_{1,0} * x_1 + b_{0,0}) + \\
& \qquad w^1_{1,1} * max(0, w^0_{0,1} * x_0 + w^0_{1,1} * x_1 + b_{0,1}) + \\
& \qquad w^1_{2,1} * max(0, w^0_{0,2} * x_0 + w^0_{1,2} * x_1 + b_{0,2}) + \\
& \qquad b_{1,1}) + \\
& w^2_{2,0} * max(0, w^1_{0,2} * max(0, w^0_{0,0} * x_0 + w^0_{1,0} * x_1 + b_{0,0}) + \\
& \qquad w^1_{1,2} * max(0, w^0_{0,1} * x_0 + w^0_{1,1} * x_1 + b_{0,1}) + \\
& \qquad w^1_{2,2} * max(0, w^0_{0,2} * x_0 + w^0_{1,2} * x_1 + b_{0,2}) + \\
& \qquad b_{1,2}) + \\
& b_2)
\end{aligned}
$$

Deep learning: Artificial neural networks

Artificial neural networks: Combines artificial neurons, simple computational units that compute a non-linear function of their inputs, in a computational graph

- Can approximate arbitrarily complex polynomial functions (given enough neurons)
- Neurons are organized in layers, and we expand a model in width (e.g. more neurons per layer) or depth (e.g. more layers)