PSY9511: Seminar 7

Deep learning for computer vision tasks

Esten H. Leonardsen

07.11.24

UNIVERSITY
OF OSLO

Outline

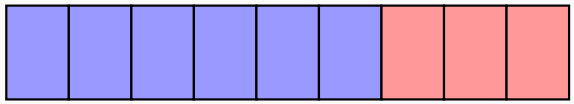
1. Exercise 4
2. Deep learning
    - Motivation
    - (Deep) neural networks
    - Training procedure
3. Convolutional neural networks for computer vision

Weekly exercises

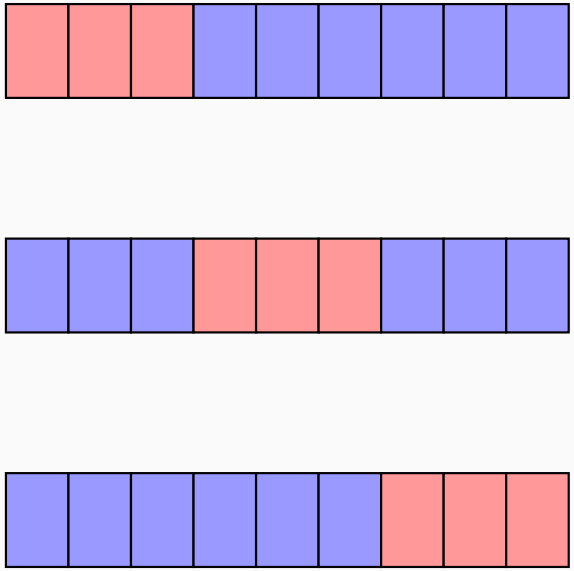
- The weekly exercises are **mandatory**
- The deadlines are **strict**

Validation procedures

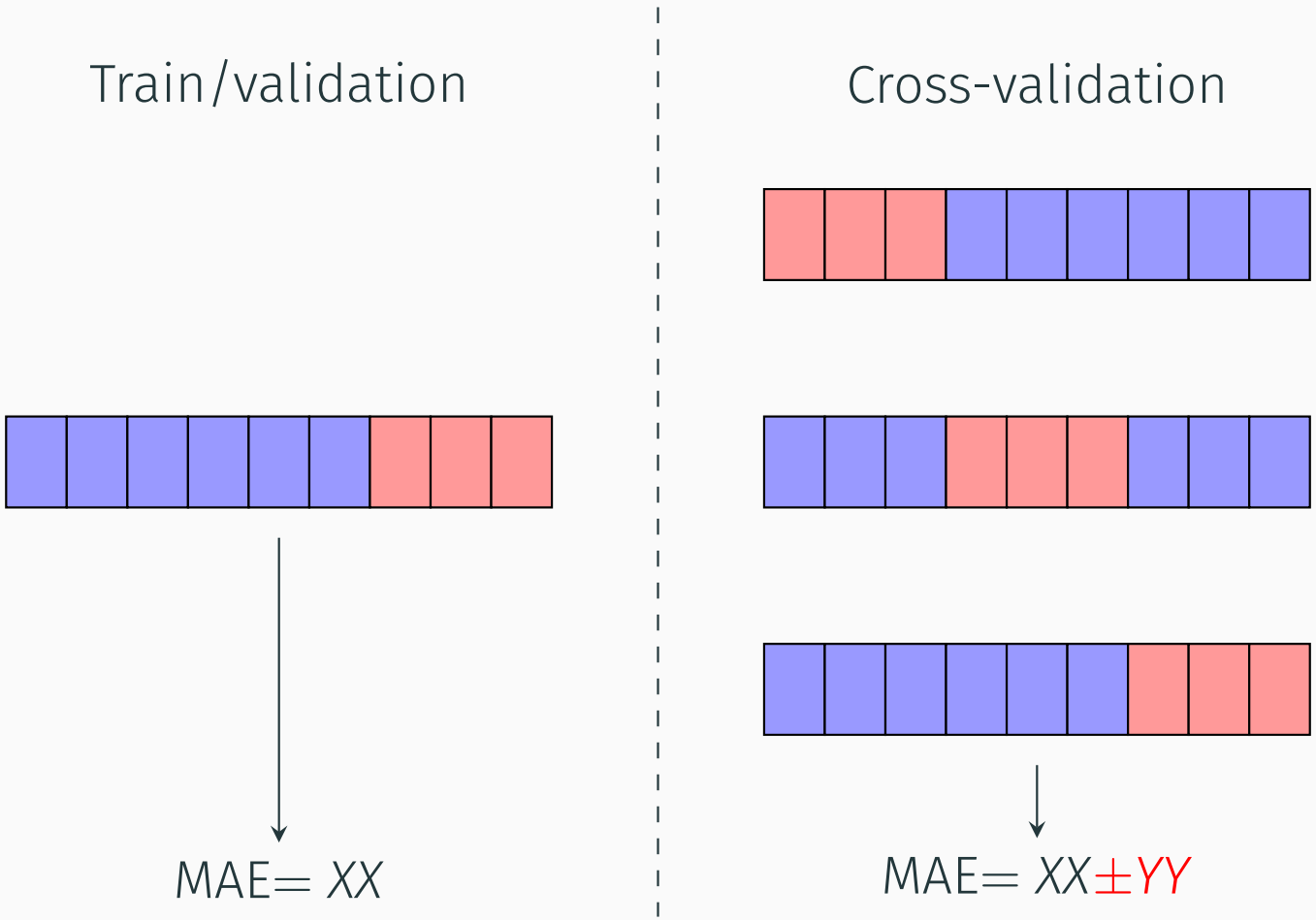
Train/validation



Cross-validation



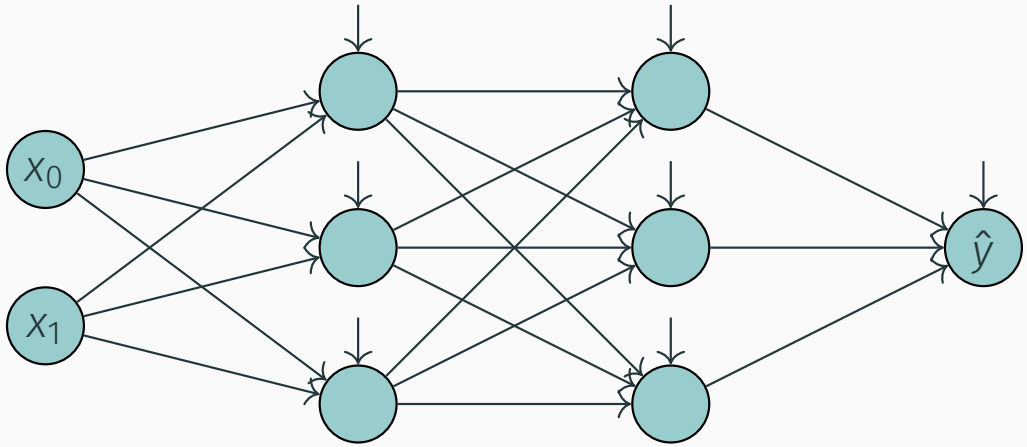
Validation procedures



Deep learning

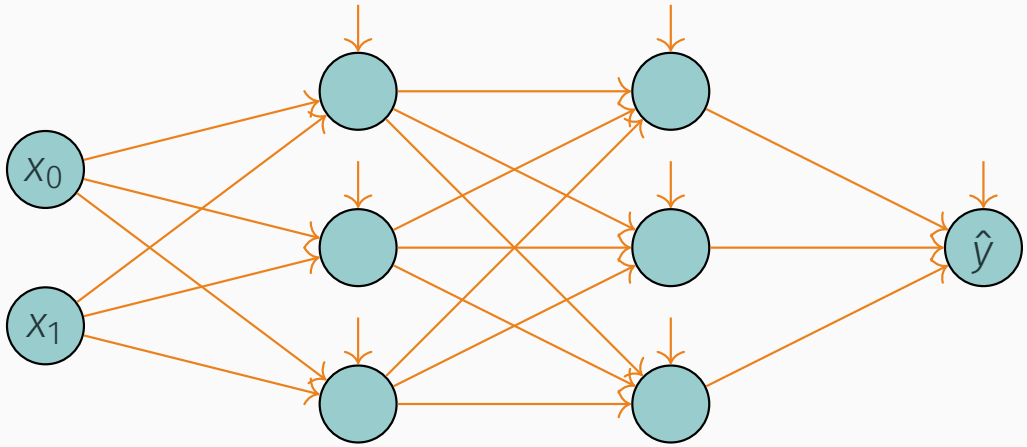
UNIVERSITY
OF OSLO

Deep learning: Training



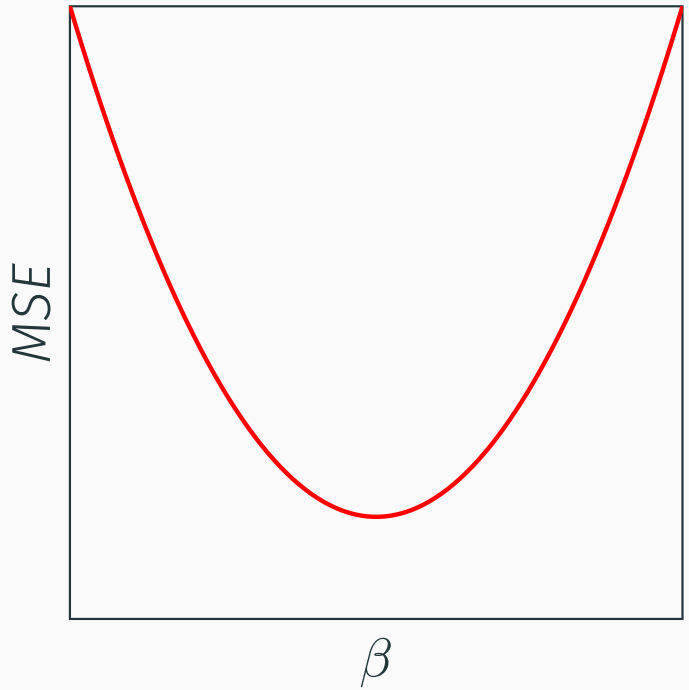
$$
\begin{aligned}
\hat{y} = max(0, & w^2_{0,0} * max(0, w^1_{0,0} * max(0, w^0_{0,0} * x_0 + w^0_{1,0} * x_1 + b_{0,0}) + \\
& w^1_{1,0} * max(0, w^0_{0,1} * x_0 + w^0_{1,1} * x_1 + b_{0,1}) + \\
& w^1_{2,0} * max(0, w^0_{0,2} * x_0 + w^0_{1,2} * x_1 + b_{0,2}) + \\
& b_{1,0}) + \\
& w^2_{1,0} * max(0,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

Deep learning: Training

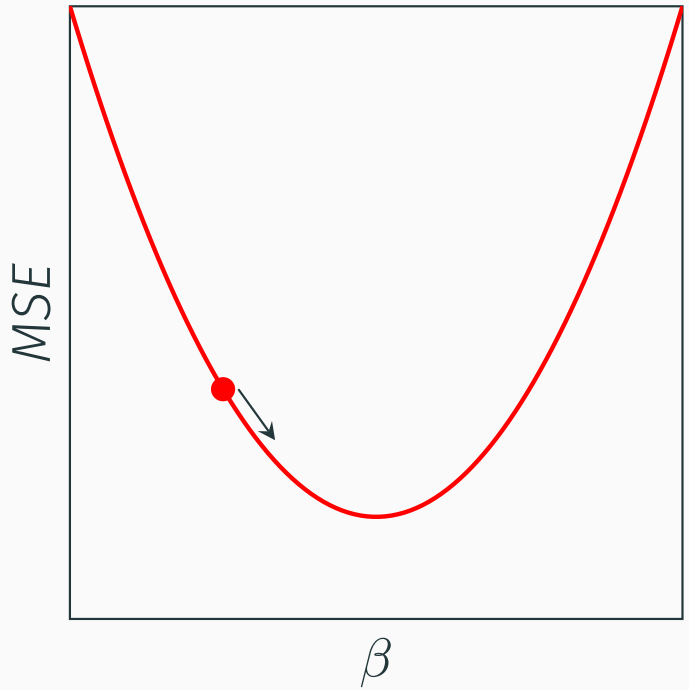


$$
\begin{aligned}
\hat{y} = max(0, & w^2_{0,0} * max(0, w^1_{0,0} * max(0, w^0_{0,0} * x_0 + w^0_{1,0} * x_1 + b_{0,0}) + \\
& w^1_{1,0} * max(0, w^0_{0,1} * x_0 + w^0_{1,1} * x_1 + b_{0,1}) + \\
& w^1_{2,0} * max(0, w^0_{0,2} * x_0 + w^0_{1,2} * x_1 + b_{0,2}) + \\
& b_{1,0}) + \\
& w^2_{1,0} * max(0,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

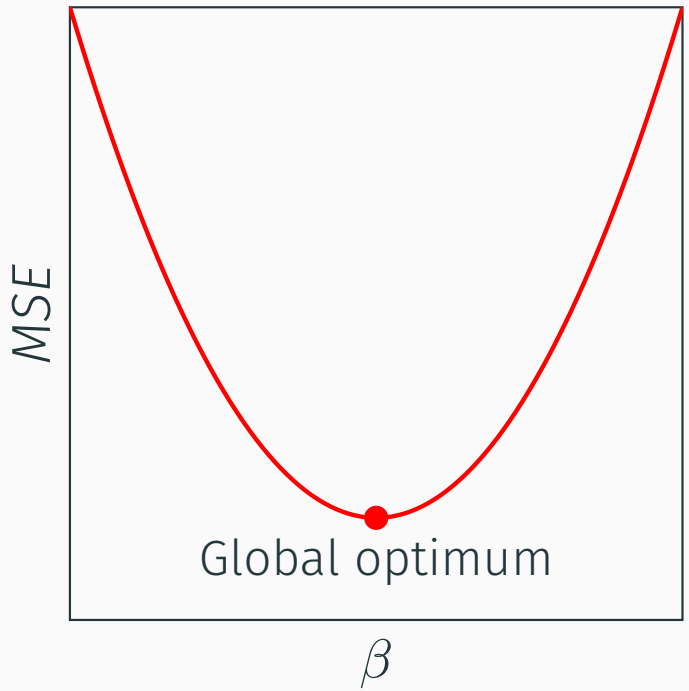
Deep learning: Training



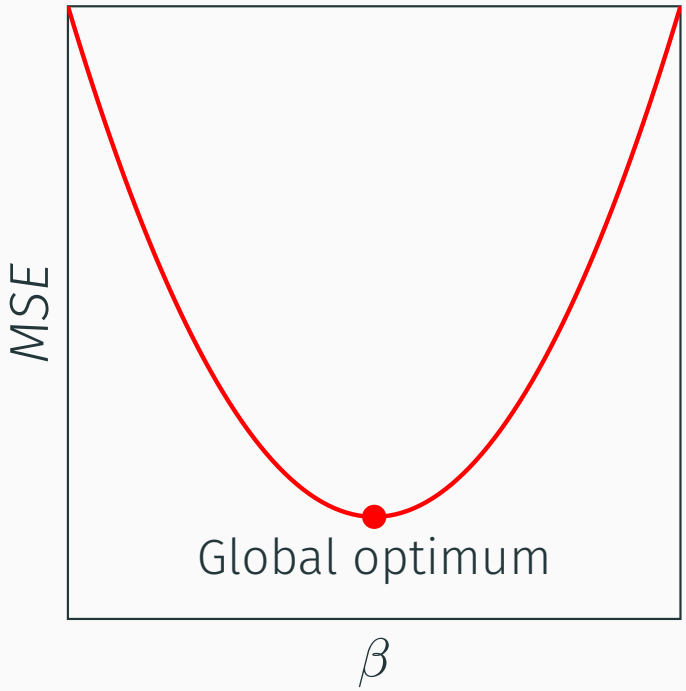
Deep learning: Training



Deep learning: Training



Deep learning: Training



- $|\beta| \rightarrow 10^6 - 10^{12}$
- $\beta_x \implies \beta_y$

Deep learning: Training

Learning representations by back-propagating errors

**David E. Rumelhart*, Geoffrey E. Hinton†
& Ronald J. Williams***

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].

Deep learning: Training

Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton†
& Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].