Deep learning
Feed-forward neural networks

Ole Winther

Bioinformatics Centre, Department of Biology
University of Copenhagen (UCph)

Dept for Applied Mathematics and Computer Science
Technical University of Denmark (DTU)

November 18, 2025

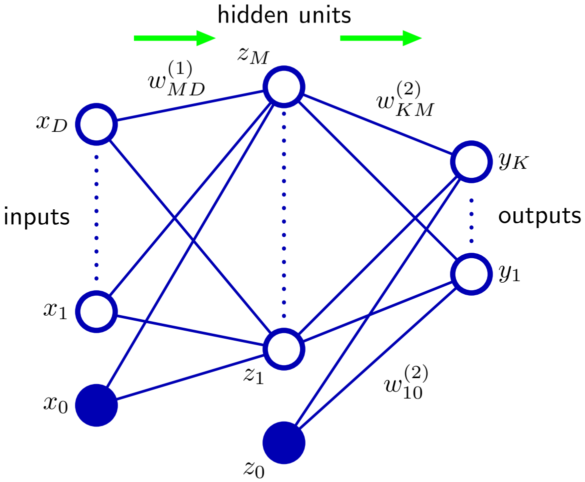
Objectives of lecture

- Feed-forward neural network (FFNN)
- Next week:
- Training with error back-propagation
- We only need to understand the principles
- Autograd - automated differentiation handles the derivation for us!

Many thanks to Tapani Raiko for making and sharing first version of these slides!

Part 1:
Feed-forward neural networks

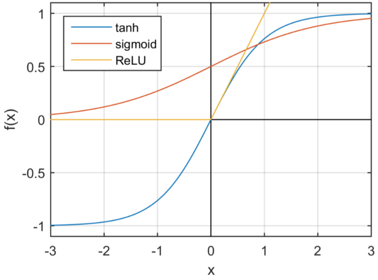
Feed forward neural networks



$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=0}^{M} w_{kj}^{(2)} \underbrace{f\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)}_{z_j}\right)$$

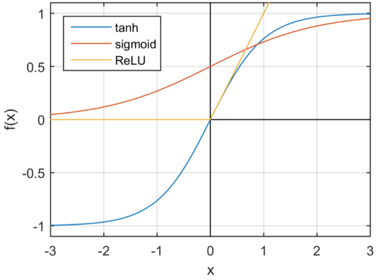
Non-linearity and training

- Linear activation functions will give a linear network.
- Logistic function $\sigma(a) = \frac{1}{1+e^{-a}}$
- Hyperbolic tangent $\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$
- Rectified linear $\mathrm{relu}(a) = \max(0, a)$



Non-linearity and training

- Linear activation functions will give a linear network.
- Logistic function $\sigma(a) = \frac{1}{1+e^{-a}}$
- Hyperbolic tangent $\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$
- Rectified linear $\mathrm{relu}(a) = \max(0, a)$



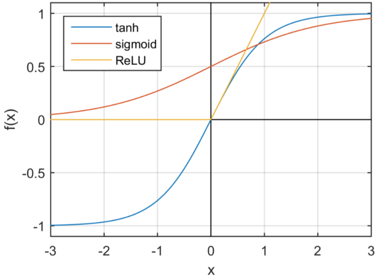
- Supervised learning
- Labeled training set

$$\mathcal{D} = \{(x_i, t_i) | i = 1, \ldots, n\} .$$

- Input x_i and target t_i.

Non-linearity and training

- Linear activation functions will give a linear network.
- Logistic function $\sigma(a) = \frac{1}{1+e^{-a}}$
- Hyperbolic tangent $\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$
- Rectified linear $\mathrm{relu}(a) = \max(0, a)$



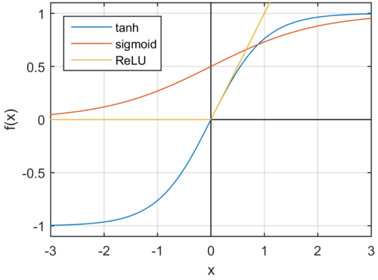
- Supervised learning
- Labeled training set

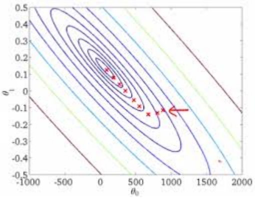
$$\mathcal{D} = \{(x_i, t_i) | i = 1, \ldots, n\} \ .$$

- Input x_i and target t_i.
- Minimize training error by (stochastic) gradient descent

Non-linearity and training

- Linear activation functions will give a linear network.
- Logistic function $\sigma(a) = \frac{1}{1+e^{-a}}$
- Hyperbolic tangent $\tanh(a) = \frac{e^{a}-e^{-a}}{e^{a}+e^{-a}}$
- Rectified linear $\mathrm{relu}(a) = \max(0, a)$





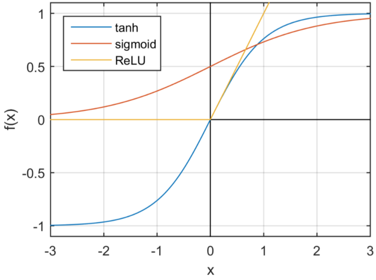
- Supervised learning
- Labeled training set

$$\mathcal{D} = \{(x_i, t_i) | i = 1, \ldots, n\} \ .$$

- Input x_i and target t_i.
- Minimize training error by (stochastic) gradient descent

Non-linearity and training

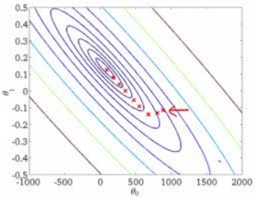
- Linear activation functions will give a linear network.
- Logistic function $\sigma(a) = \frac{1}{1+e^{-a}}$
- Hyperbolic tangent $\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$
- Rectified linear $\mathrm{relu}(a) = \max(0, a)$

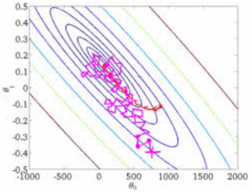


- Supervised learning
- Labeled training set

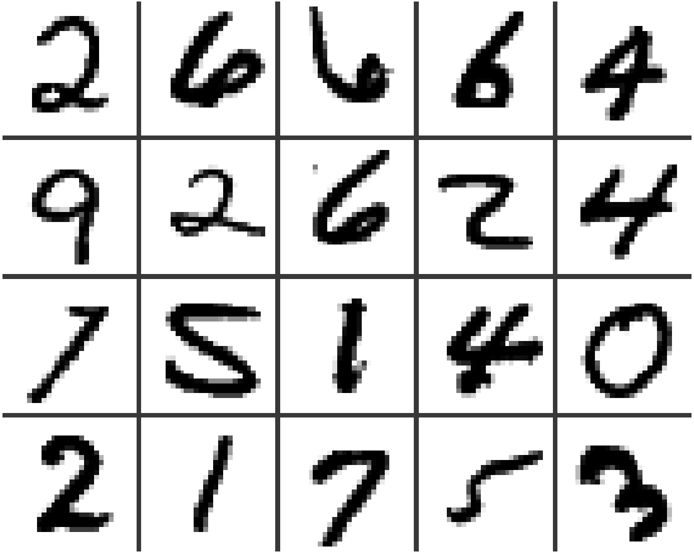
$$\mathcal{D} = \{(x_i, t_i) | i = 1, \ldots, n\} \ .$$

- Input x_i and target t_i.
- Minimize training error by (stochastic) gradient descent



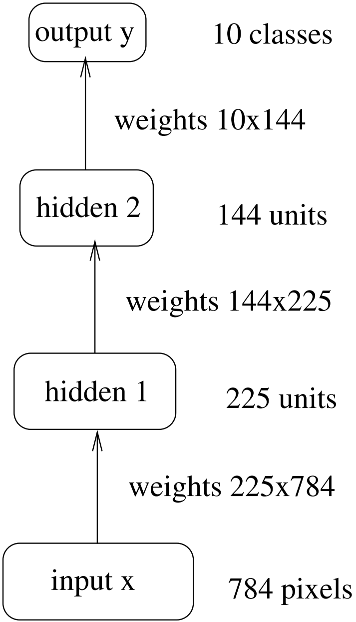


Example: MNIST handwritten digits



Train a network to classify 28×28 images.
Data: 60000 input images $\mathbf{x}(n)$ and labels $t(n)$.
Example model gives around 1.2% test error.

Example Network



$$\mathbf{y} = \mathbf{h}^{(3)} = \text{softmax}(\mathbf{W}^{(3)}\mathbf{h}^{(2)} + \mathbf{b}^{(3)})$$

$$\mathbf{h}^{(2)} = \text{relu}(\mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)})$$

$$\mathbf{h}^{(1)} = \text{relu}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

$$\text{softmax}(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

$$\text{relu}(z) = max(0, z)$$

Softmax

- Softmax function

$$\mathrm{softmax}(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

  has two nice properties:
- $\mathrm{softmax}(\mathbf{z})_i \geq 0$
- $\sum_i \mathrm{softmax}(\mathbf{z})_i = 1$

Softmax

- Softmax function

$$\text{softmax}(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

  has two nice properties:
- $\text{softmax}(\mathbf{z})_i \geq 0$
- $\sum_i \text{softmax}(\mathbf{z})_i = 1$
- MNIST, output labels: $0, 1, \ldots, 9$.
- Output of network

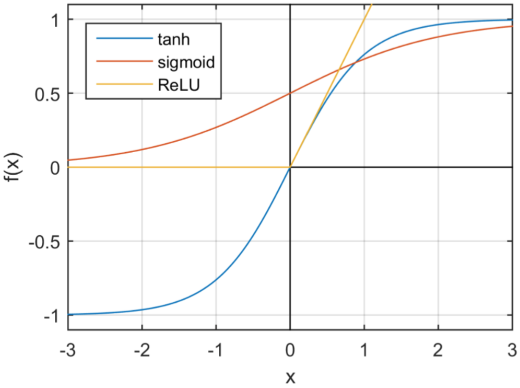
$$\mathbf{y} = \text{softmax}(\mathbf{W}^{(3)}\mathbf{h}^{(2)} + \mathbf{b}^{(3)})$$

  interpreted as class(-conditional) probabilities:
- So given input $\mathbf{x}$, according to the model, the probability of digit i is:

$$p(\text{digit} = i|\mathbf{x}) = y_{i+1}$$

- with $i = 0, \ldots, 9$.

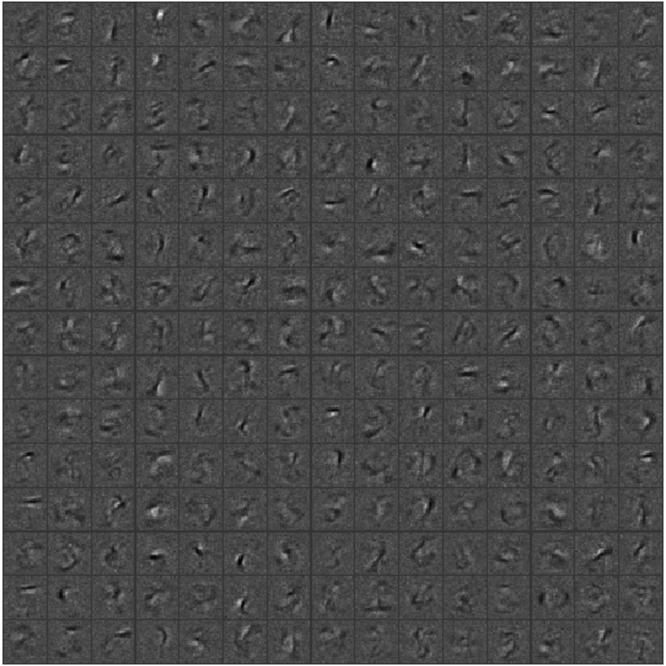
On activation functions



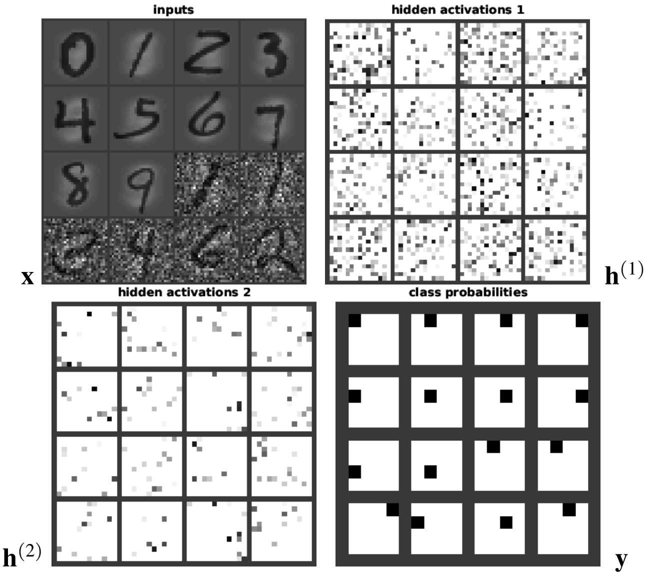
- $\mathrm{relu}(z) = \max(0, z)$ is replacing old sigmoid and tanh.
- Note that identity function would lead into:

$$\begin{aligned}
\mathbf{h}^{(2)} &= \mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)} \\
&= \mathbf{W}^{(2)}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)} \\
&= (\mathbf{W}^{(2)}\mathbf{W}^{(1)})\mathbf{x} + (\mathbf{W}^{(2)}\mathbf{b}^{(1)} + \mathbf{b}^{(2)}) \\
&= \mathbf{W}'\mathbf{x} + \mathbf{b}'
\end{aligned}$$

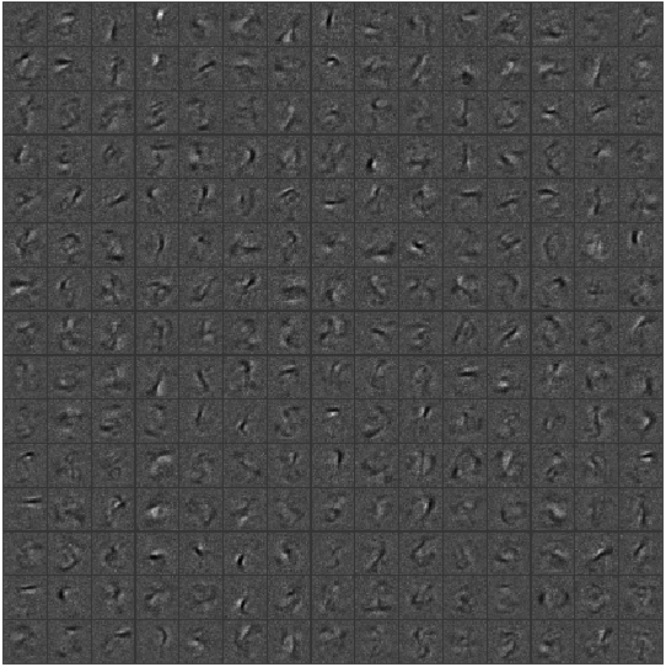
Weight matrix $\mathbf{W}^{(1)}$ size 225×784



Signals $\mathbf{x} \to \mathbf{h}^{(1)} \to \mathbf{h}^{(2)} \to \mathbf{h}^{(3)}$



Weight matrix $\mathbf{W}^{(1)}$ size 225×784



Part 2:

Doing yourself - understand how feed-forward neural networks approximate functions

How a feed-forward neural network learns XOR

- Consider two-layer network with two hidden units:

$$y(\mathbf{x}) = \Theta\left(W_1^{(2)} h_1^{(1)} + W_2^{(2)} h_2^{(1)} + b_1^{(1)}\right)$$

$$h_1^{(1)} = \Theta\left(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + b_1^{(1)}\right)$$

$$h_2^{(1)} = \Theta\left(W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + b_2^{(1)}\right)$$

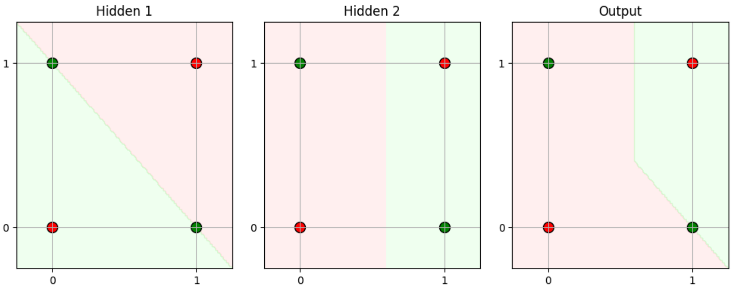
- Step function activation function: $\Theta(z) = 1$ if $z \geq 0$ and 0 otherwise

How a feed-forward neural network learns XOR

- Consider two-layer network with two hidden units:

$$
\begin{aligned}
y(\mathbf{x}) &= \Theta\left(W_1^{(2)} h_1^{(1)} + W_2^{(2)} h_2^{(1)} + b_1^{(1)}\right) \\
h_1^{(1)} &= \Theta\left(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + b_1^{(1)}\right) \\
h_2^{(1)} &= \Theta\left(W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + b_2^{(1)}\right)
\end{aligned}
$$

- Step function activation function: $\Theta(z) = 1$ if $z \geq 0$ and 0 otherwise
- Fetch XOR_NNSandbox.ipynb from Absalon and turn the knobs to learn XOR



x1	x2	h1	h2	target	pred	OK
0	0	1	0	0	0	✓
0	1	0	0	1	0	✗
1	0	0	1	1	1	✓
1	1	0	1	0	1	✗

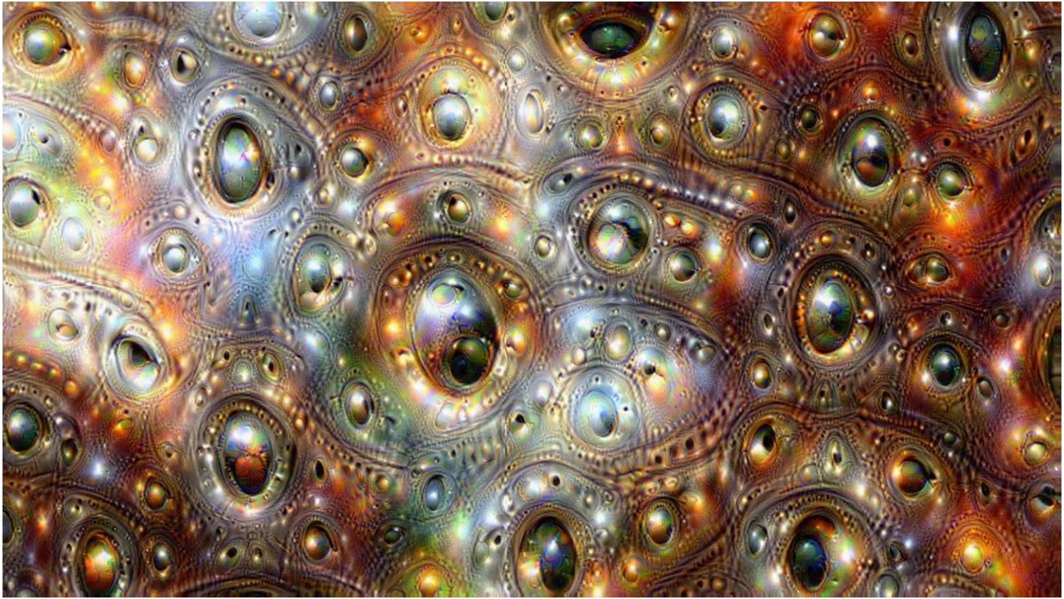
- Red is output 1 and green is output 0.
- We will also consider this problem in Assignment 1.

TensorFlow Playground

playground.tensorflow.org

References

- Online book: Michael Nielsen, Neural networks and deep learning
- Book also online: Ian Goodfellow and Yoshua Bengio and Aaron Courville, Deep learning
- LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton, Deep learning, Nature 521.7553 (2015): 436-444.
- Mnih, Volodymyr, et al., Human-level control through deep reinforcement learning, Nature 518.7540 (2015): 529-533.
- Alipanahi, Babak, et al., Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning, Nature biotechnology (2015).
- Silver, David, et al., Mastering the game of Go with deep neural networks and tree search, Nature 529.7587 (2016): 484-489.
- Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., & Bengio, Y. (2015), Show, attend and tell: Neural image caption generation with visual attention. arXiv preprint arXiv:1502.03044.
- Mansimov, Elman, et al., Generating Images from Captions with Attention. arXiv preprint arXiv:1511.02793 (2015).
- Larsen, Anders Boesen Lindbo, Soren Kaae Sonderby, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. arXiv preprint arXiv:1512.09300 (2015).
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville, Deep learning, 2016.
- Michael Nielsen, Neural Networks and Deep Learning
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional, NIPS, 2012. Neural Networks,



Thanks!
Ole Winther