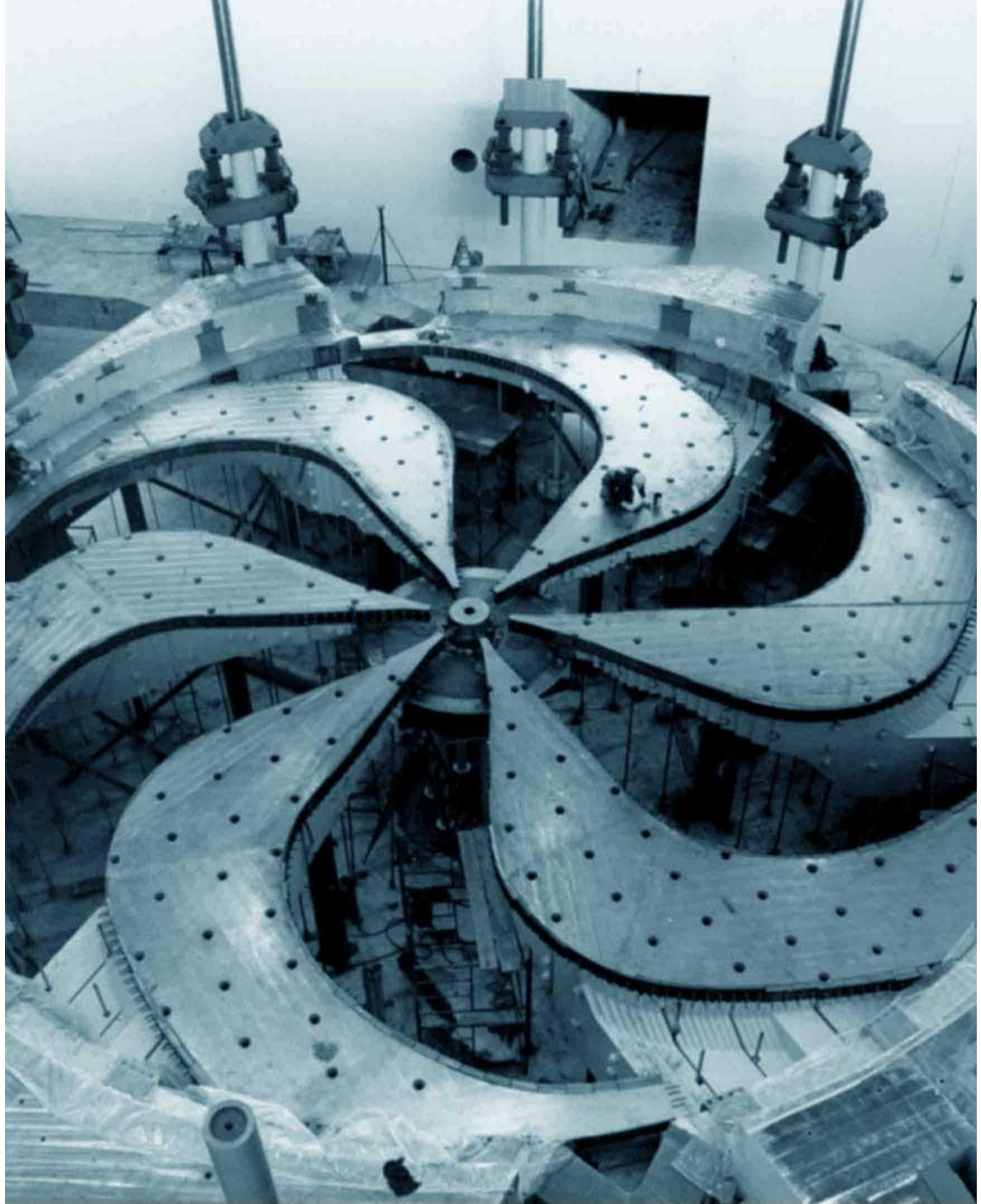
Welcome to the ML hands-on session!

Wojtek Fedorko

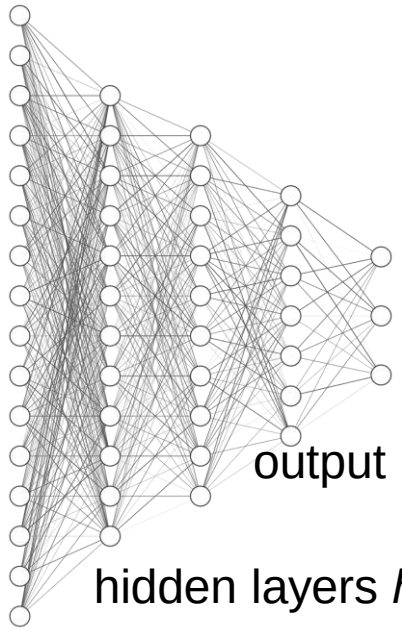


2019-08-23

About this Tutorial

- Focus on technical aspects
- Convince you that deep learning is not that hard and can be useful
- Show you some common obstacles
- Absolute minimum ‘theory’ background
  - But that will only get you so far…
- Now go to: https://github.com/wfedorko/Science_Week_ML_tutorial

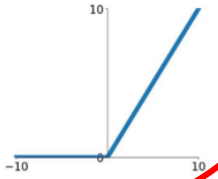
Multi-layer perceptron (MLP) at a glance



- Composition of linear transformations and activation functions: $\boldsymbol{h_i = f(W h_{i-1} + b)}$
- Activation functions in the hidden layers:

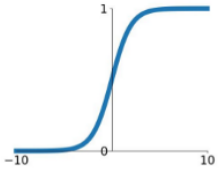
ReLU

$\max(0, x)$



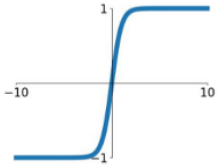
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



tanh

$\tanh(x)$



- Activation function on the output layer depends on the task

Tasks, output layer activations and appropriate loss functions: rules of thumb for supervised learning

- o - output node value, z - output node before activation function, y - target value, c - class index, M – number of categories
- Formulas for a single example (usually we need to calculate an average over batch)
- Motivation from Maximum Likelihood Principle : Ian Goodfellow https://www.deeplearningbook.org ch. 5

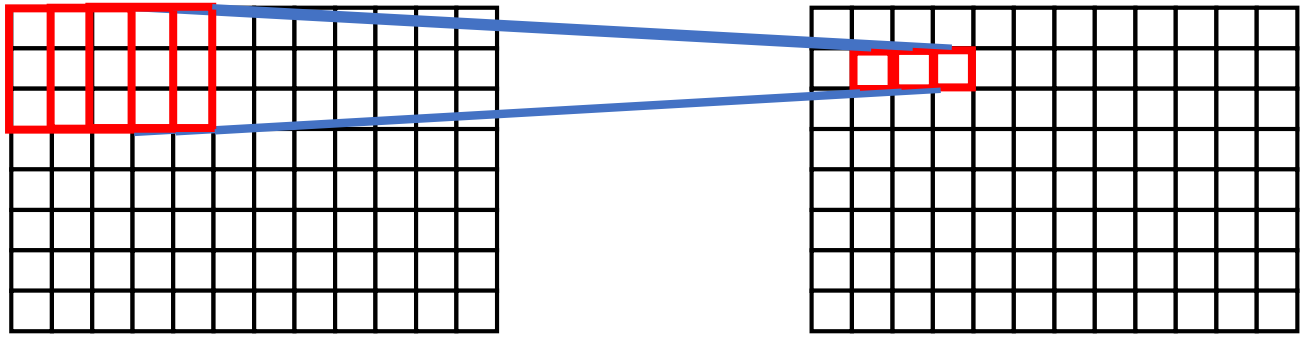
Task:	Binary classification	Multi-class classification	Regression
Output layer activation:	sigmoid $$o = \frac{1}{1+e^{-z}}$$	softmax $$o_c = \frac{e^{z_c}}{\sum_{j=1}^{M} e^{z_j}}$$	linear $$o = z$$
Loss function:	“Binary Cross Entropy” $$-(y\log(o) + (1-y)\log(1-o))$$	“Cross-Entropy” $$-\sum_{c=1}^{M} y_c \log(o_c)$$	“Mean Squared Error” $$(y-o)^2$$

Backpropagation and Stochastic Gradient Descent

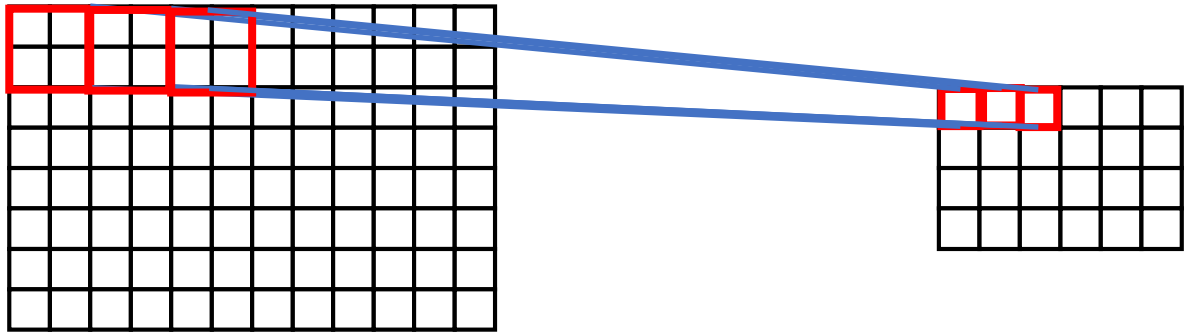
- Learning is an optimization problem: we want to find values of network parameters **W** that minimize loss L
- SGD:
  - Initialize **W**
  - Iterate until convergence:
    - At iteration i take a batch of examples and calculate the (average) loss L_i (forward pass)
    - Compute the gradient ∇L_i wrt $\mathbf{W}_i$
    - Update the weights $\mathbf{W}_{i+1} \leftarrow \mathbf{W}_i - \lambda \nabla L_i$

Backpropagation! – recursively applying chain rule compute derivatives starting from the loss

- Convolutional layers: slide learnable ‘filters’ across the image



- Pooling layers: take max or average of a window, reduce ‘size’ due to stride>1



Suggested Exercises

- Optimize simple CNN (hyperparameters)
  - Network architecture + learning parameters tuning
  - Kernel sizes
  - Batchnorm
  - Dropout
  - Learning rate/optimizer
  - Datanormalization
- ‘Advanced’ architectures
  - ResNet
  - Densenet
- Regression problem
  - ‘fit’ particle energy