Karlsruhe Institute of Technology

Neural Networks

Oskar Taubert (SCC)

SCC

Neural Network Concept

- (very) remotely brain inspired computational system
- directed graph, encoding an ordered system of simple mathematical transformations
- successor of the perceptron concept (i.e. logistic regression)
- more complicated ’fit’ i.e. universal function approximator
- usually supervised machine learning

Motivation

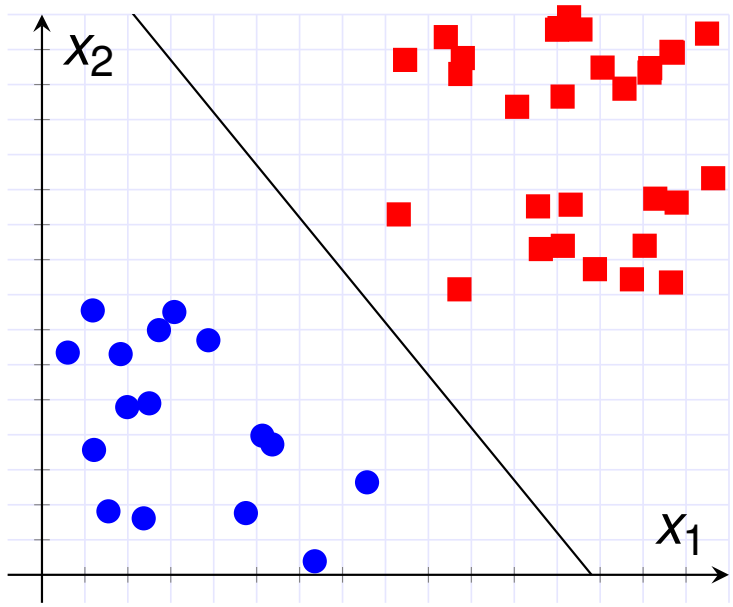
- automation of tasks machines are traditionally bad at
  - image recognition
  - natural language processing
  - …
- image processing challenges e.g. character recognition (MNIST)

Perceptron

decide whether an image depicts a 0:

$$o = \Theta(w \cdot x + b) \text{ where}$$

- Output: $o \in \{0, 1\}$
- Input: $x \in \mathbb{R}^{|\text{Pixels}|}$
- Parameter: $w \in \mathbb{R}^{|\text{Pixels}|}$
- Parameter: $b \in \mathbb{R}$

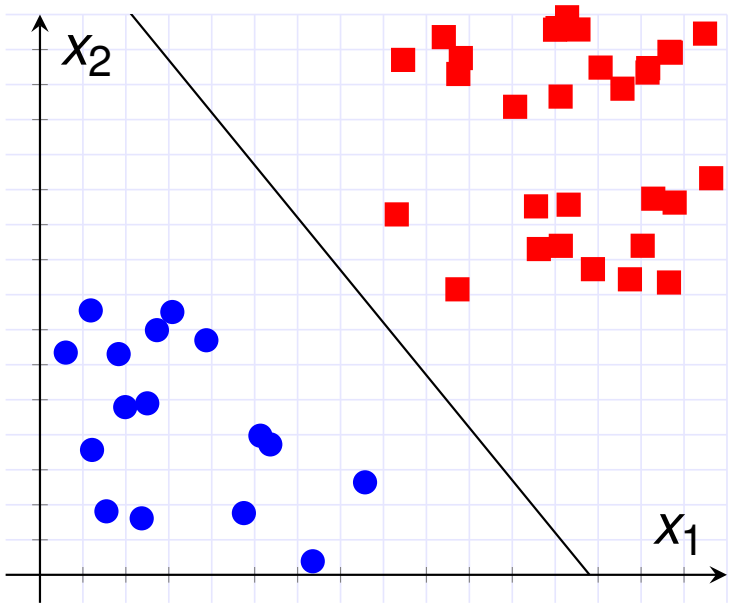


Perceptron

decide whether an image depicts a 0:

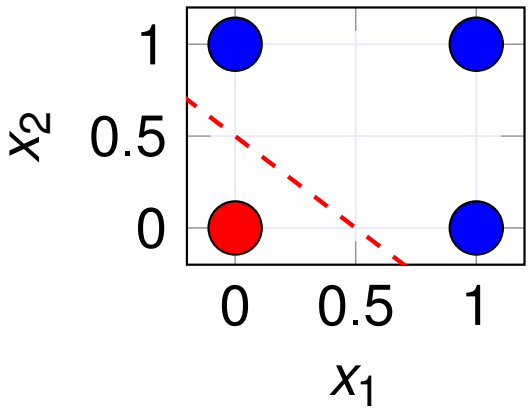
$$o = \Theta(w \cdot x + b) \text{ where}$$

- Output: $o \in \{0, 1\}$
- Input: $x \in \mathbb{R}^{|\text{Pixels}|}$
- Parameter: $w \in \mathbb{R}^{|\text{Pixels}|}$
- Parameter: $b \in \mathbb{R}$



Problem: Non-linear decision boundaries

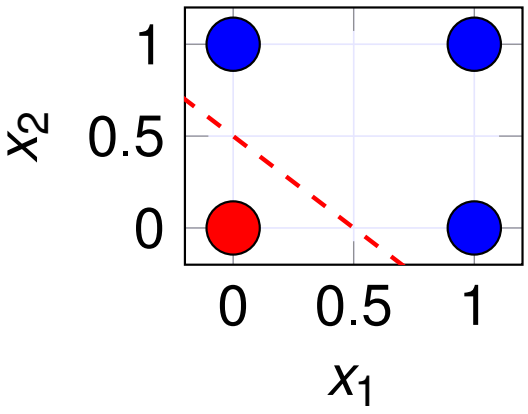
XOR

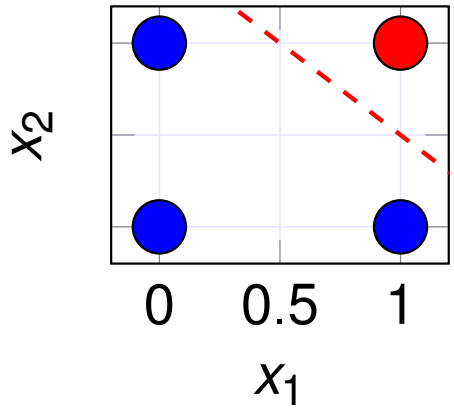


OR-gate

$$h_1 = x_1 + x_2 - 1$$

XOR





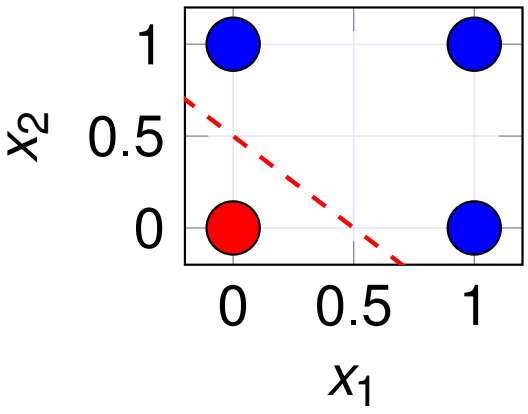
OR-gate

$$h_1 = x_1 + x_2 - 1$$

NAND-gate

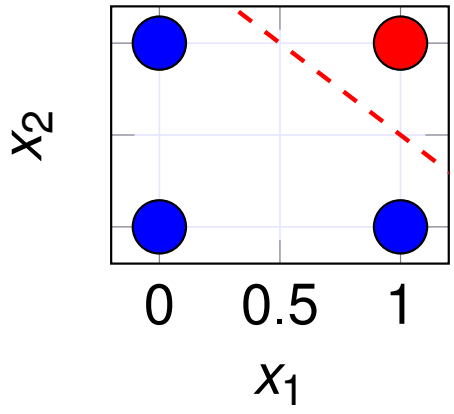
$$h_2 = -x_1 - x_2 - 1.5$$

XOR



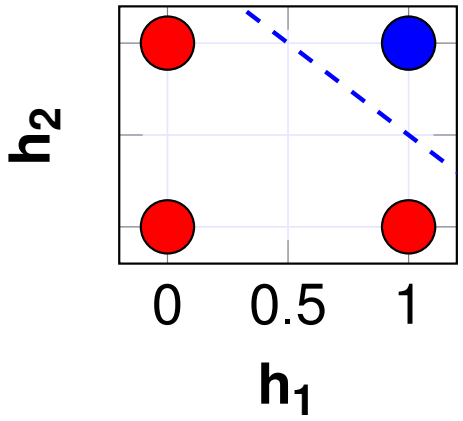
OR-gate

$$h_1 = x_1 + x_2 - 1$$



NAND-gate

$$h_2 = -x_1 - x_2 - 1.5$$



AND-gate

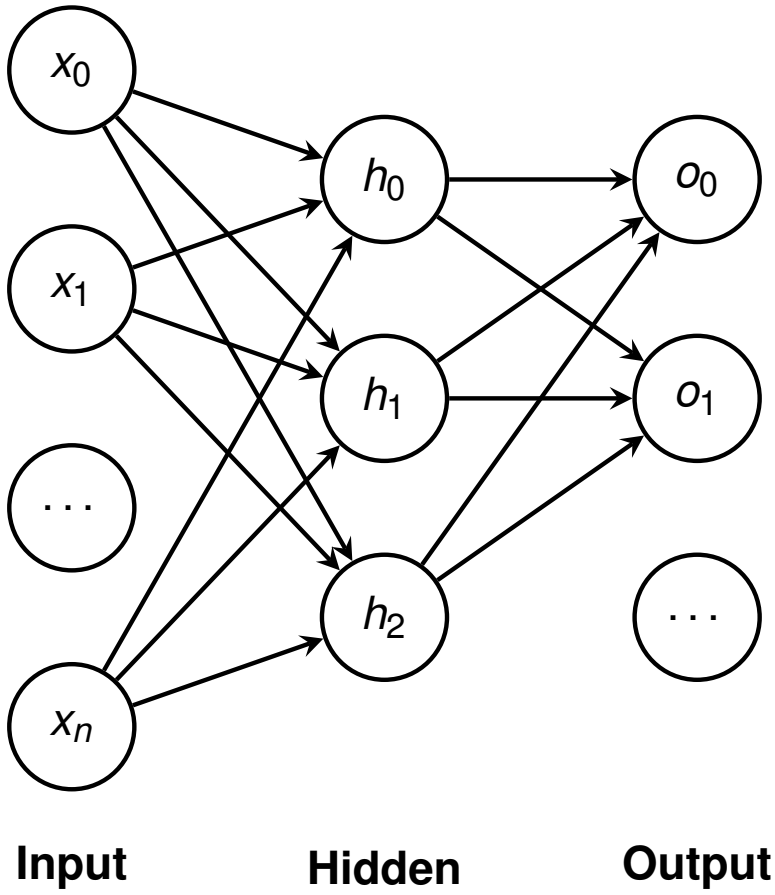
$$\hat{y} = \mathbf{h_1} + \mathbf{h_2} - 1.5$$

Multilayer Perceptron

$$o = f(W \cdot h + b)$$

$$h = g(V \cdot x + c)$$

- $o \in \mathbb{R}^{10}$
- $W \in \mathbb{R}^{10 \times 3}$
- $h \in \mathbb{R}^{3}$
- $b \in \mathbb{R}^{10}$
- $V \in \mathbb{R}^{|\text{Pixels}| \times 3}$
- $x \in \mathbb{R}^{|\text{Pixels}|}$
- $c \in \mathbb{R}^{3}$

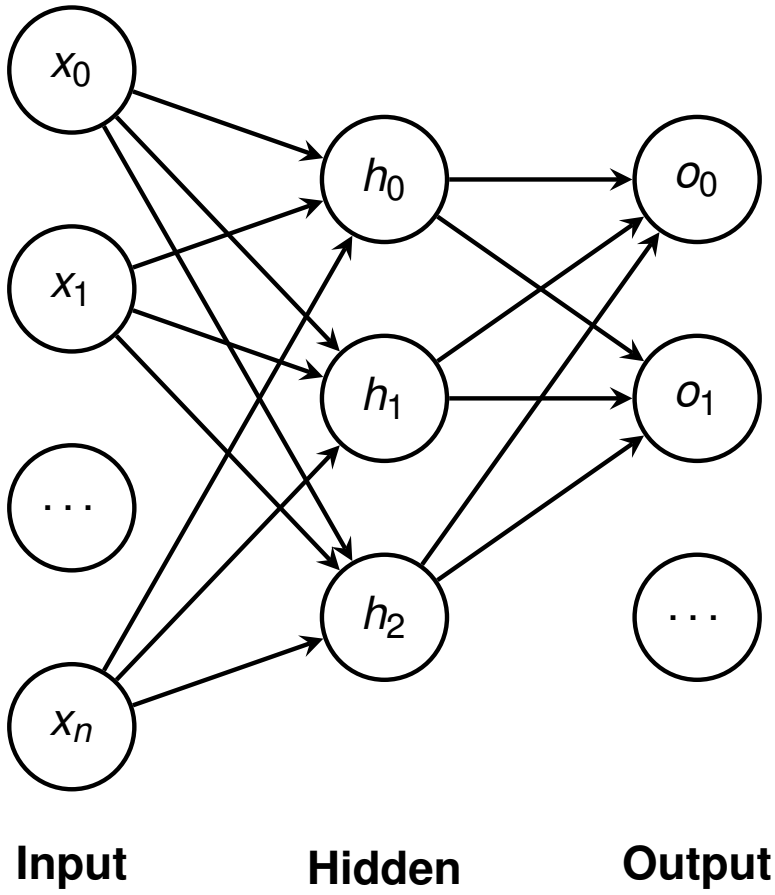


Multilayer Perceptron

$$o = f(W \cdot h + b)$$

$$h = g(V \cdot x + c)$$

- f and g?
- Values for W and V?



Training

$$\text{parameters} = \{W, V, b, c\}$$

$$\text{Error measure: } E(o, t) = (o - t)^2$$

Training

$$\text{parameters} = \{W, V, b, c\}$$

$$\text{Error measure: } E(o, t) = (o - t)^2$$

$$\text{parameters} \leftarrow \text{parameters} - \lambda \frac{\partial \text{error}}{\partial \text{parameters}}$$

Training

$$\text{parameters} = \{W, V, b, c\}$$

$$\text{Error measure: } E(o, t) = (o - t)^2$$

$$\text{parameters} \leftarrow \text{parameters} - \lambda \frac{\partial \text{error}}{\partial \text{parameters}}$$

$$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial W} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial f}\frac{\partial f}{\partial W}h$$

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial b} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial f}\frac{\partial f}{\partial b}$$

Training

$$\text{parameters} = \{W, V, b, c\}$$

$$\text{Error measure: } E(o, t) = (o - t)^2$$

$$\text{parameters} \leftarrow \text{parameters} - \lambda \frac{\partial \text{error}}{\partial \text{parameters}}$$

$$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial W} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial f}\frac{\partial f}{\partial W}h$$

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial b} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial f}\frac{\partial f}{\partial b}$$

$$\frac{\partial E}{\partial V} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial f}\frac{\partial f}{\partial h}\frac{\partial h}{\partial g}\frac{\partial g}{\partial V}x$$

Training

More general:

$$E(t, f_n(W_n \cdot f_{n-1}(\dots f_1(W_1 \cdot x))))$$

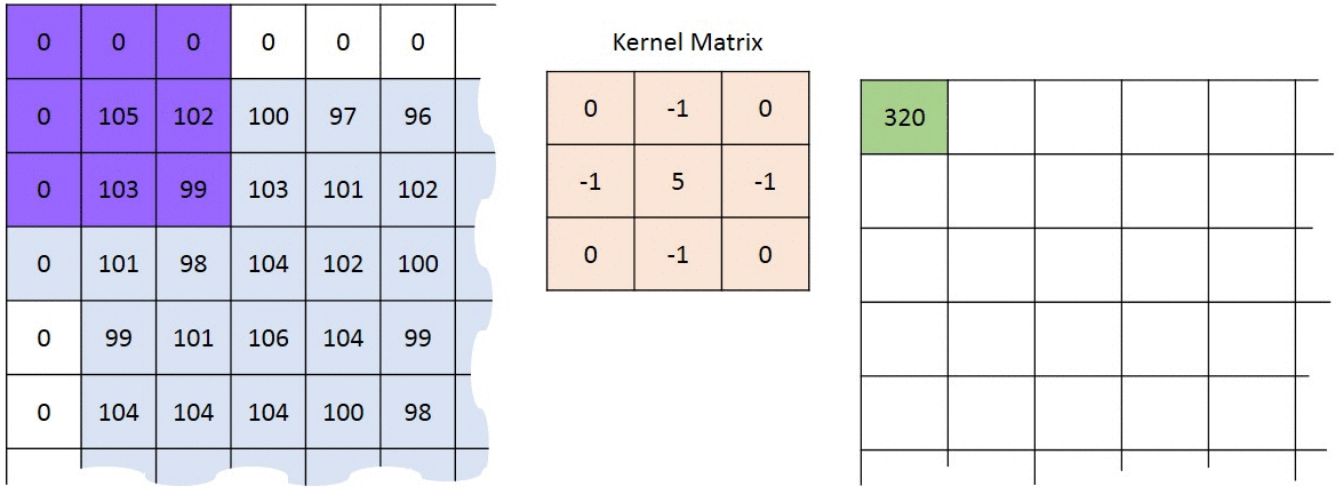
$$\frac{\partial E}{\partial W_i} = \delta_i \cdot h_{i-1}$$

$$\delta_{i-1} = \delta_i \, \partial f_{i-1}|_{W_{l-1}} \cdot W_i$$

Error functions

- Regression: MSE, KL-divergence
- Classification: Cross Entropy, NLL-loss
- Segmentation: Hinge-losses, Overlap/Dissimilarity losses

Convolutions



Convolution with horizontal and vertical strides = 1

Figure: *

Convolutions

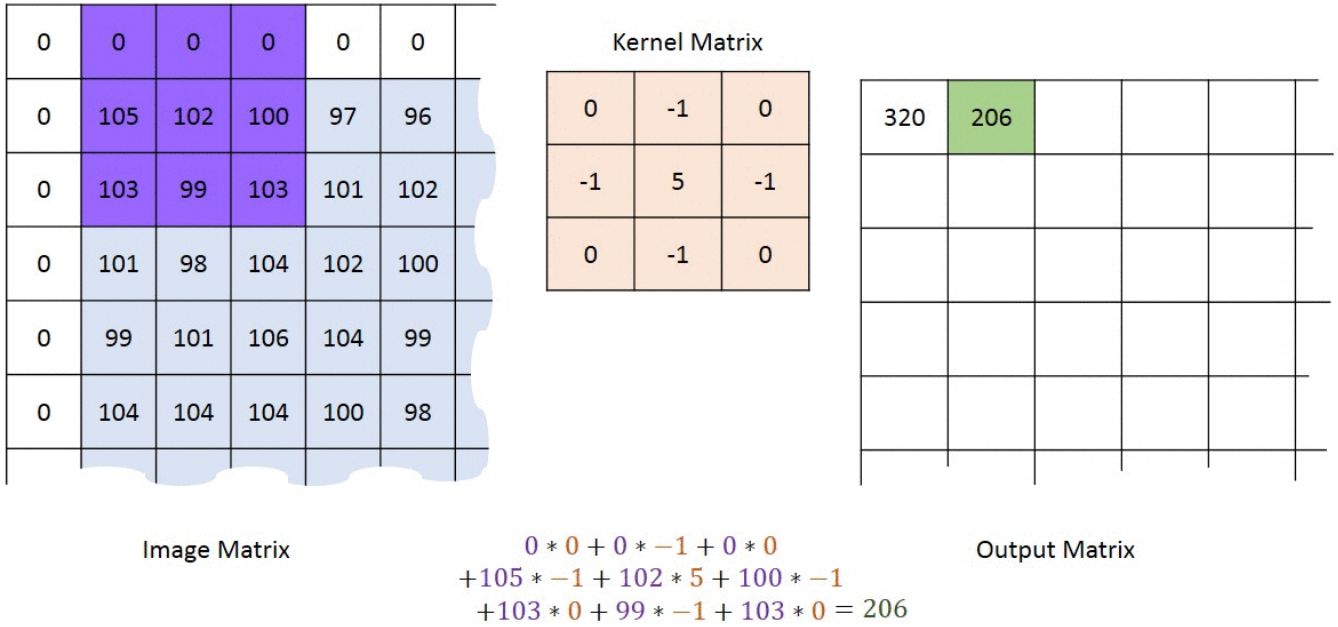
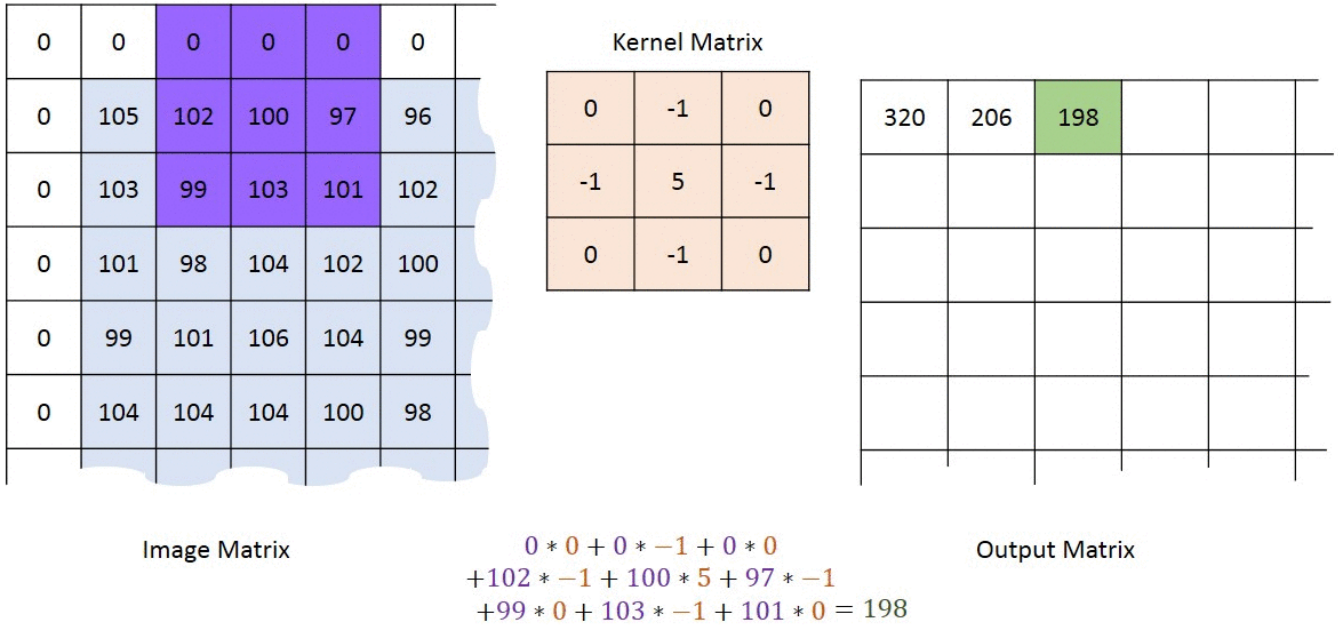


Figure: *

Convolutions

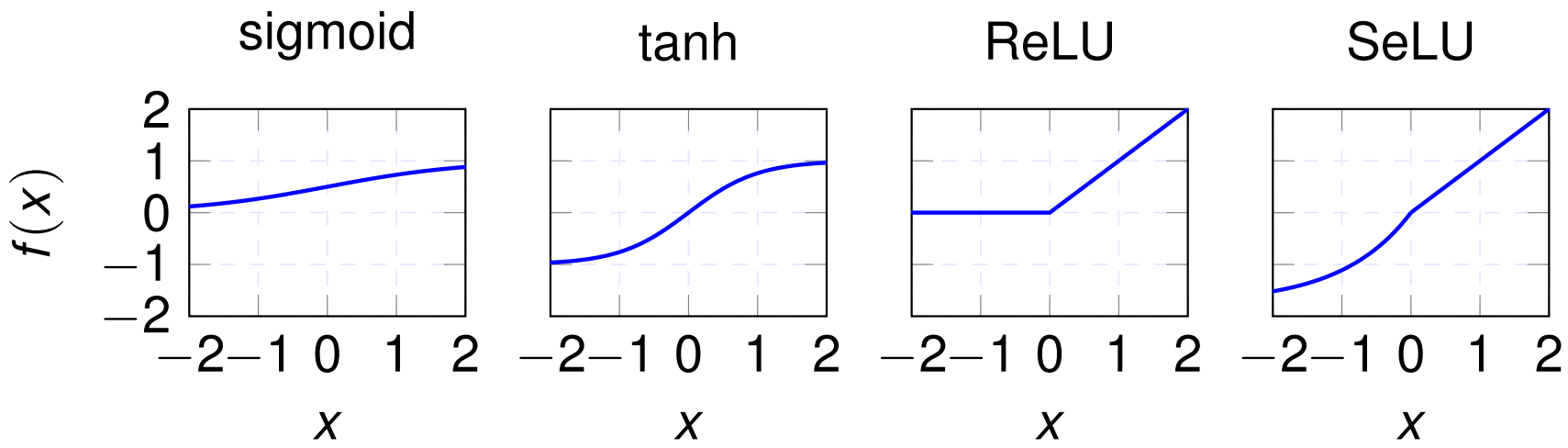


Convolution with horizontal and vertical strides = 1

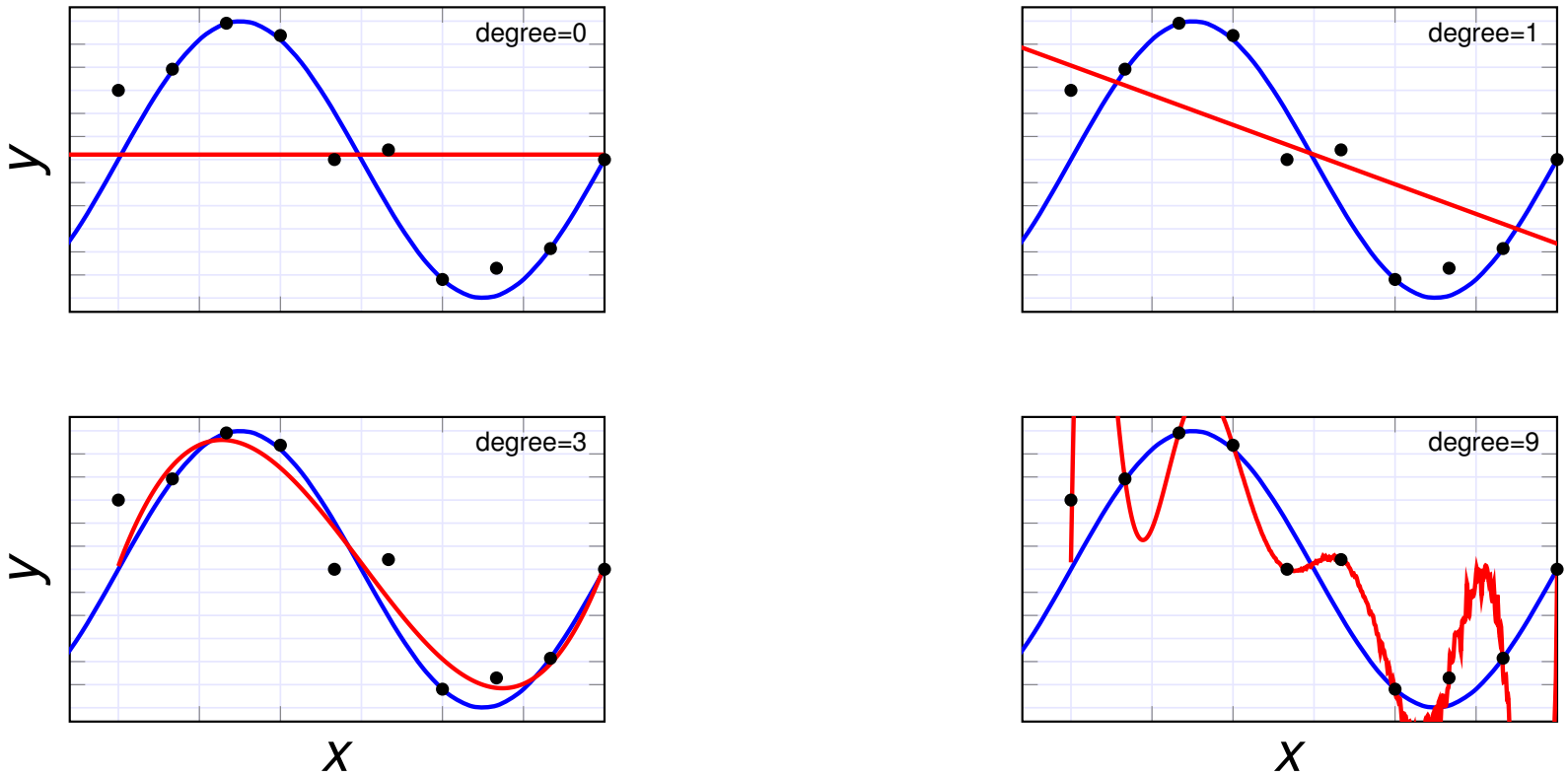
Figure: *

Activation Functions

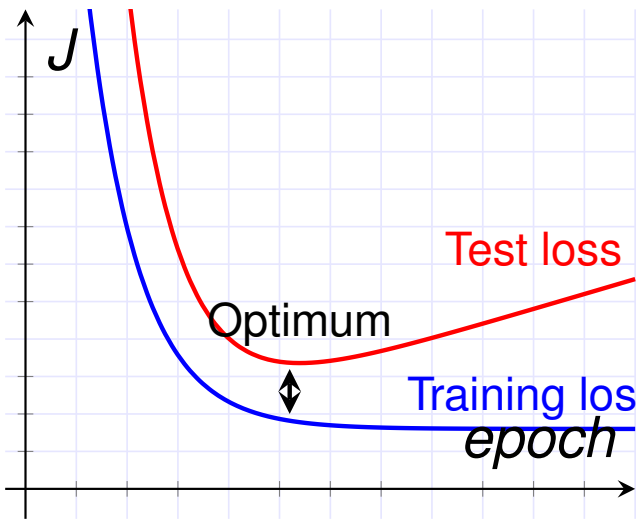
- Activation functions $f(x)$ introduce **non-linearity**, e.g. sigmoid
- Other non-linear choices, e.g. $tanh(x)$, $relu(x) = max(0, x)$, $softmax_i(x) = \frac{\exp(x_i)}{\sum_i \exp(x_i)}$, etc.
- Better numerical properties, e.g. avoid **vanishing gradient**



Regularization



Regularization



- early stopping
- weight decay
- weight sharing
- dropout
- batch normalization
- data augmentation
- more data

Hyperparameters

- guessing
- experience
- non-gradient based optimization
  - grid search
  - random search
  - particle swarm
  - genetic

Out of Scope

- residual models
- generative models
- recurrent models
- attention models
- lots
- reinforcement learning (next week)

Sources

- http://nyu-cds.sparksites.io/wp-content/uploads/2015/10/header_4@2x.png
- https://github.com/Markus-Goetz/gks-2019/blob/solutions/slides/slides.pdf