Artificial Neural Networks

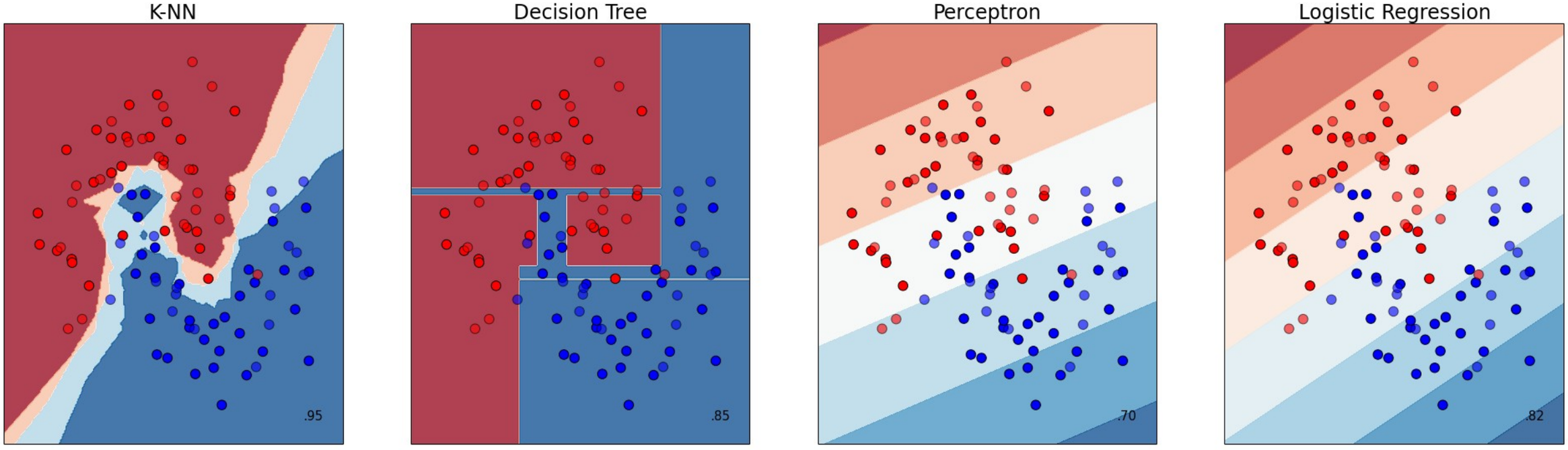
Research Skills: Machine Learning

Grzegorz Chrupała
g.chrupala @ uvt.nl

Landscape of supervised ML

Algorithm	Boundary	Learning
K-NN	Non-linear	Memorization
Decision Trees	Non-linear	Specialized rule learning
Perceptron	Linear	Mistake-driven
Logistic regression	Linear	Error function minimization (SGD)
Neural networks	**Non-linear**	**Error function minimization (SGD)**

Classification boundaries



CAVEATS

- Non-linear boundaries

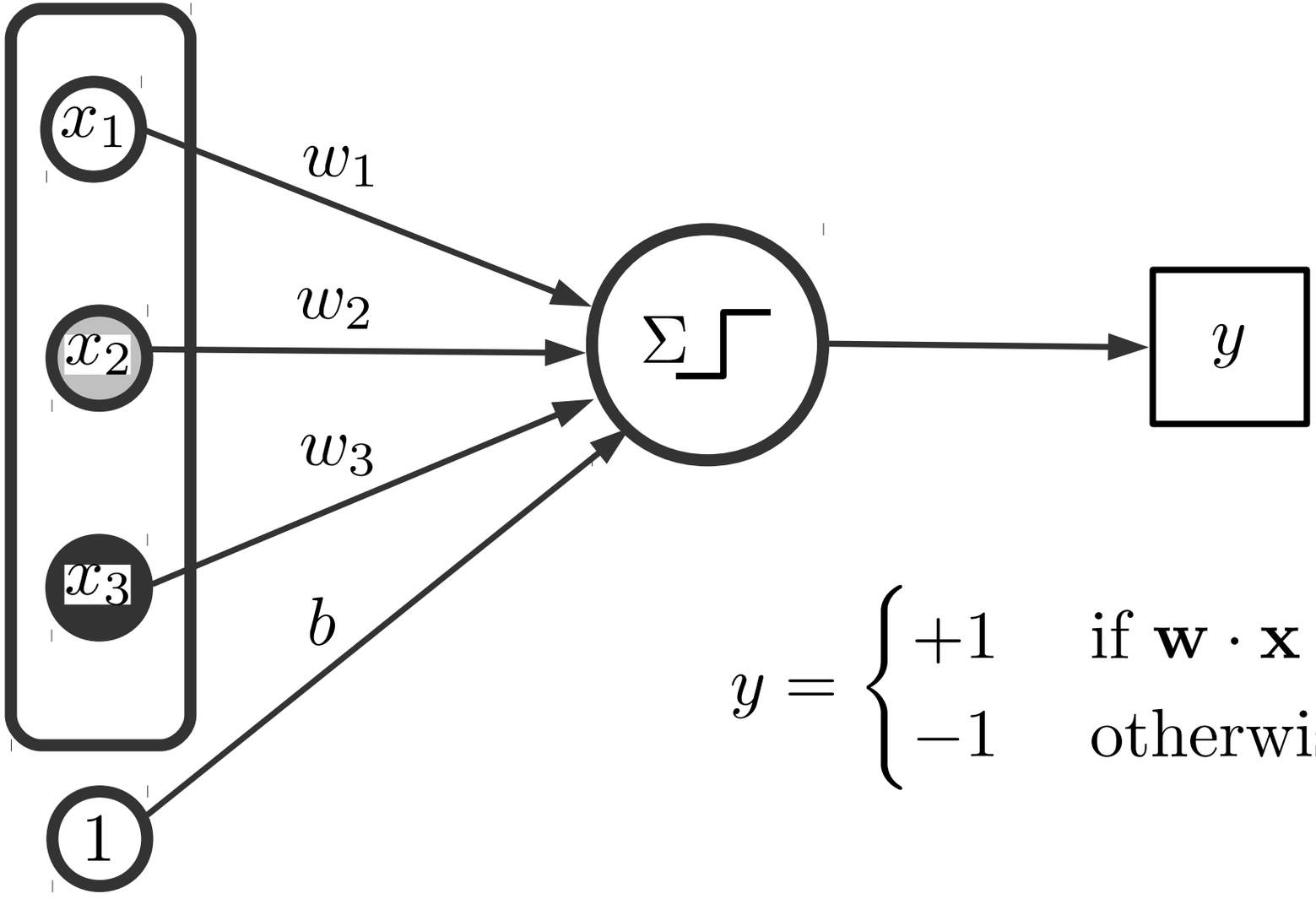
  – important for low-dimensional data
- In real life, often easier to get results with **Logistic Regression** than **K-NN**

Artificial neural networks

- Can learn non-linear decision boundaries
  - and approximate any computable function
- Learn **intermediate representations**

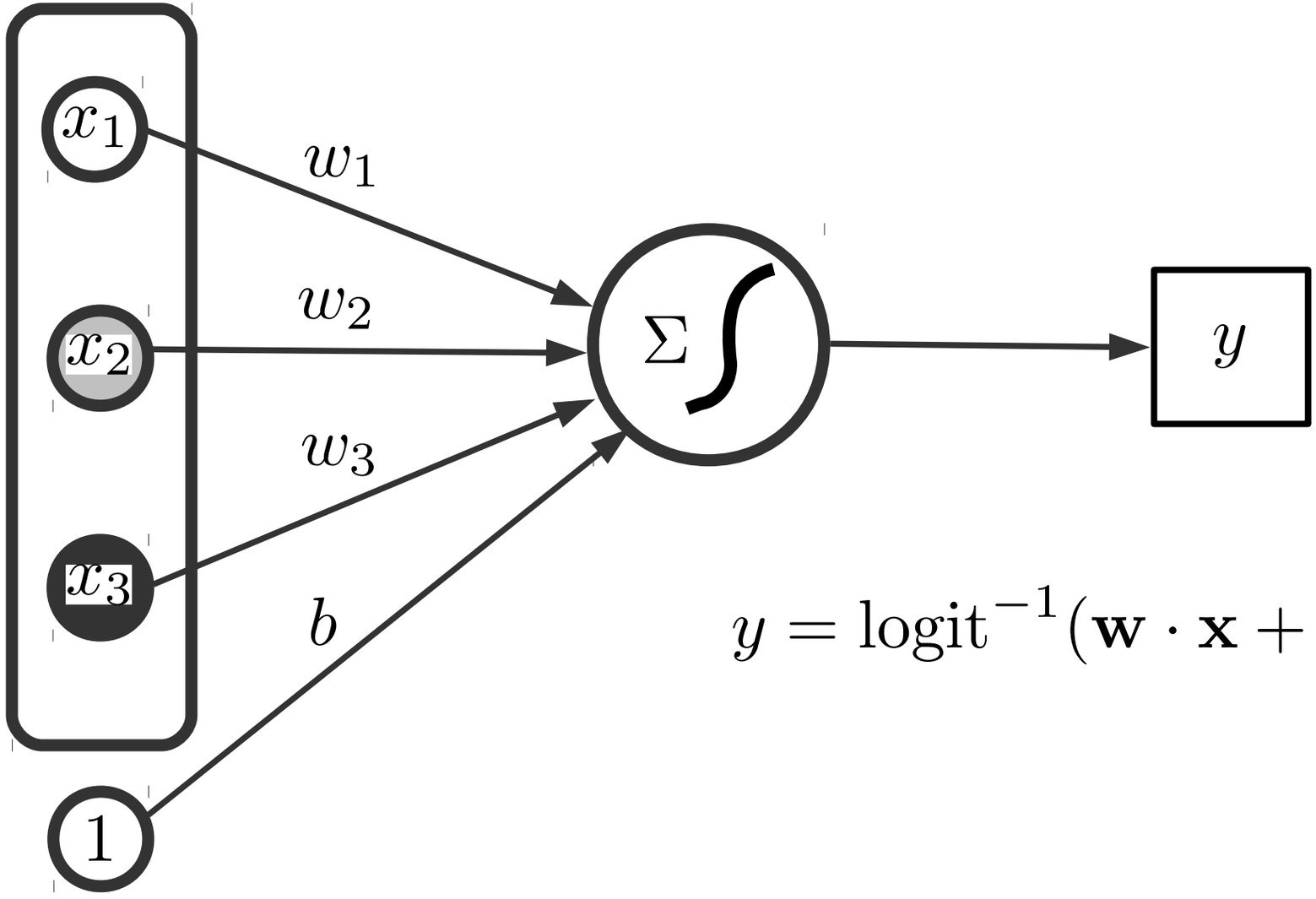
(Vast topic – we only scratch the surface)

Perceptron



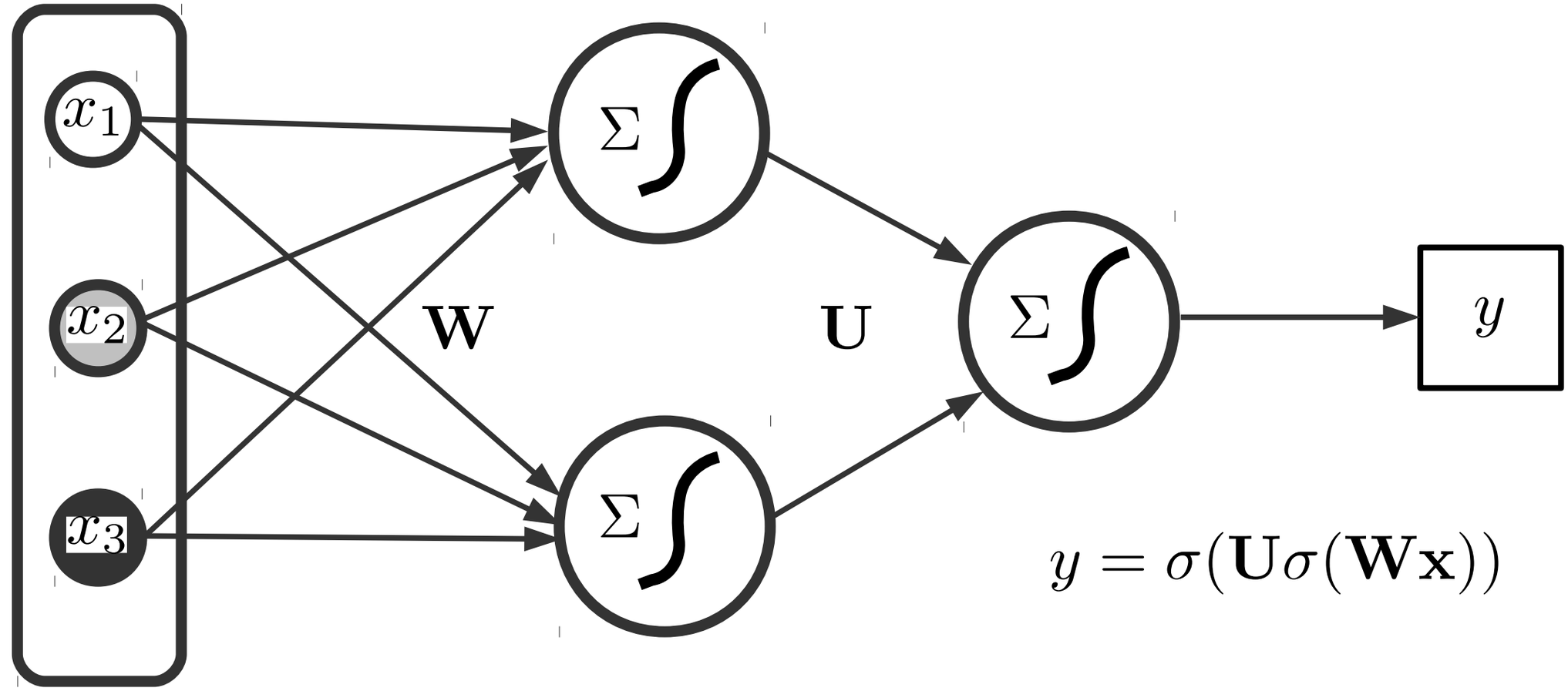
$$y = \begin{cases} +1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Logistic regression

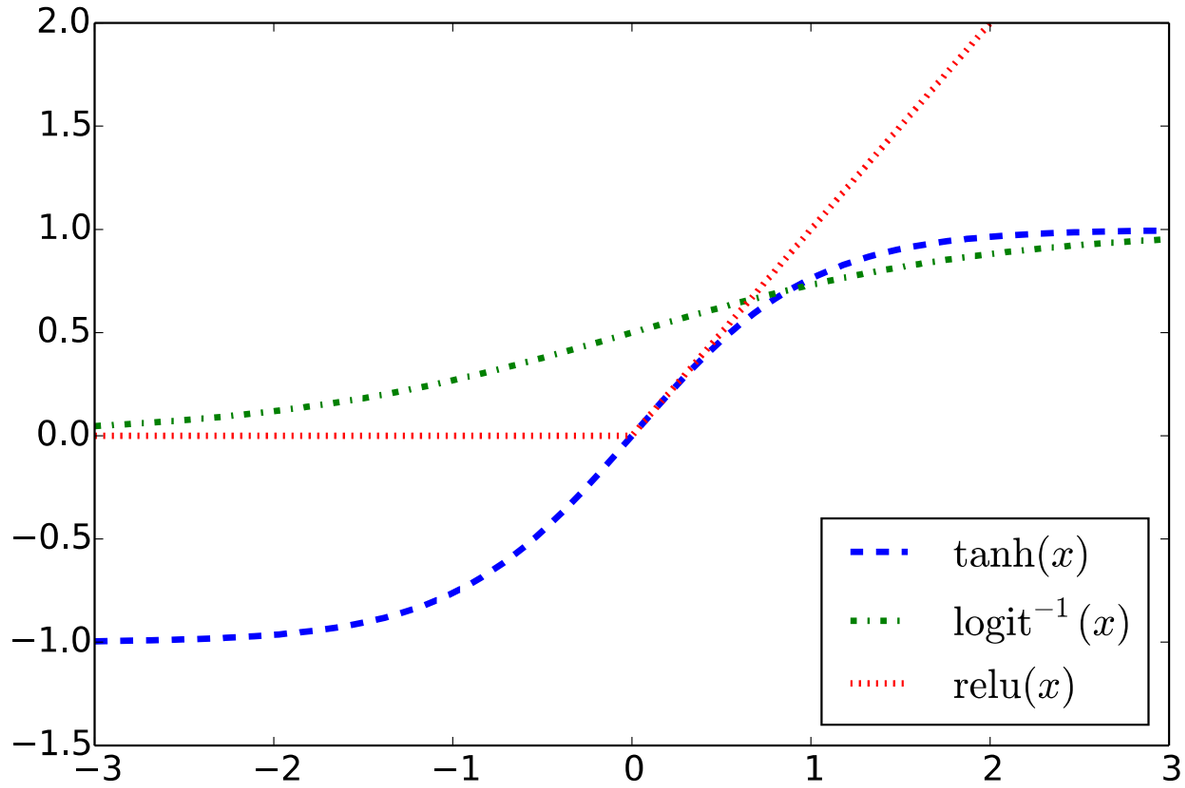


$$y = \text{logit}^{-1}(\mathbf{w} \cdot \mathbf{x} + b)$$

Neural network
(Multilayer perceptron)



Activation functions

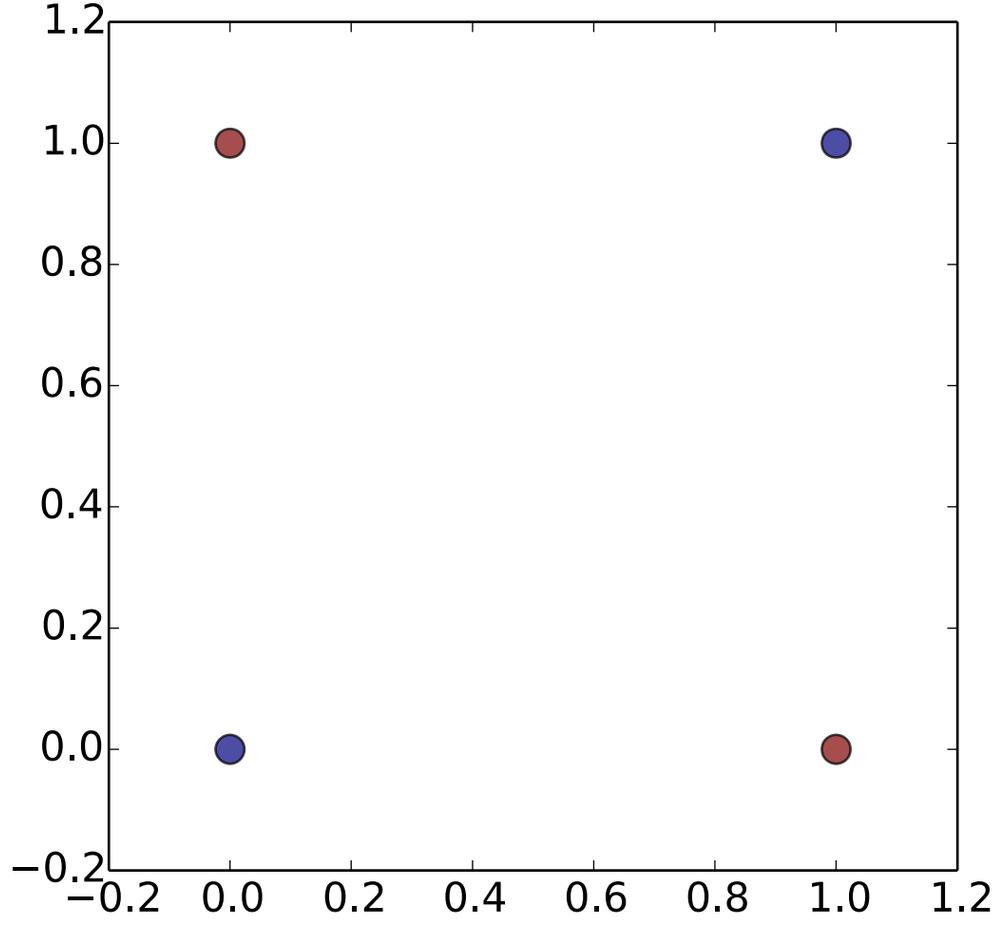


Universal function approximator

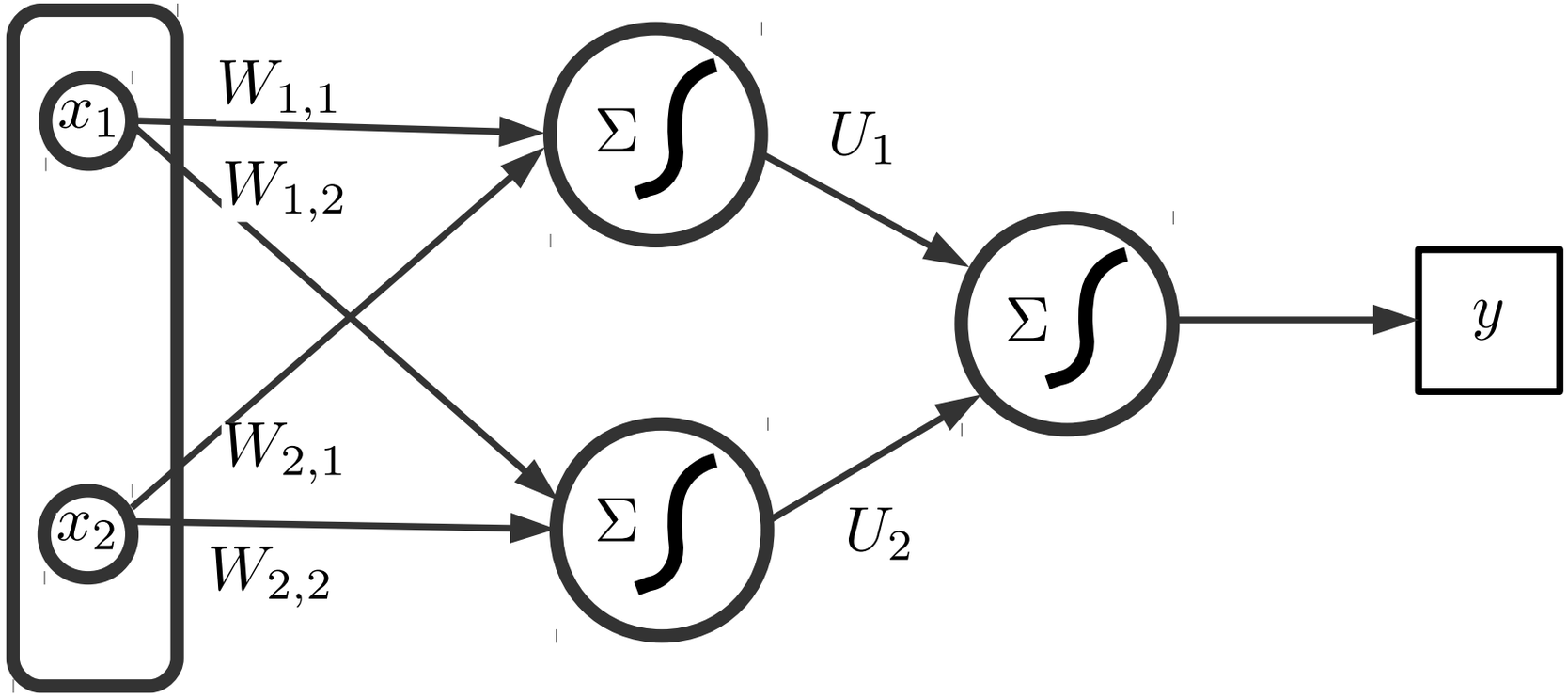
- In principle, feedforward neural nets can represent any computable function
- The problem is finding the right weights!

XOR classifier

- xor(1,1) = 0
- xor(1,0) = 1
- xor(0,1) = 1
- xor(0,0) = 0



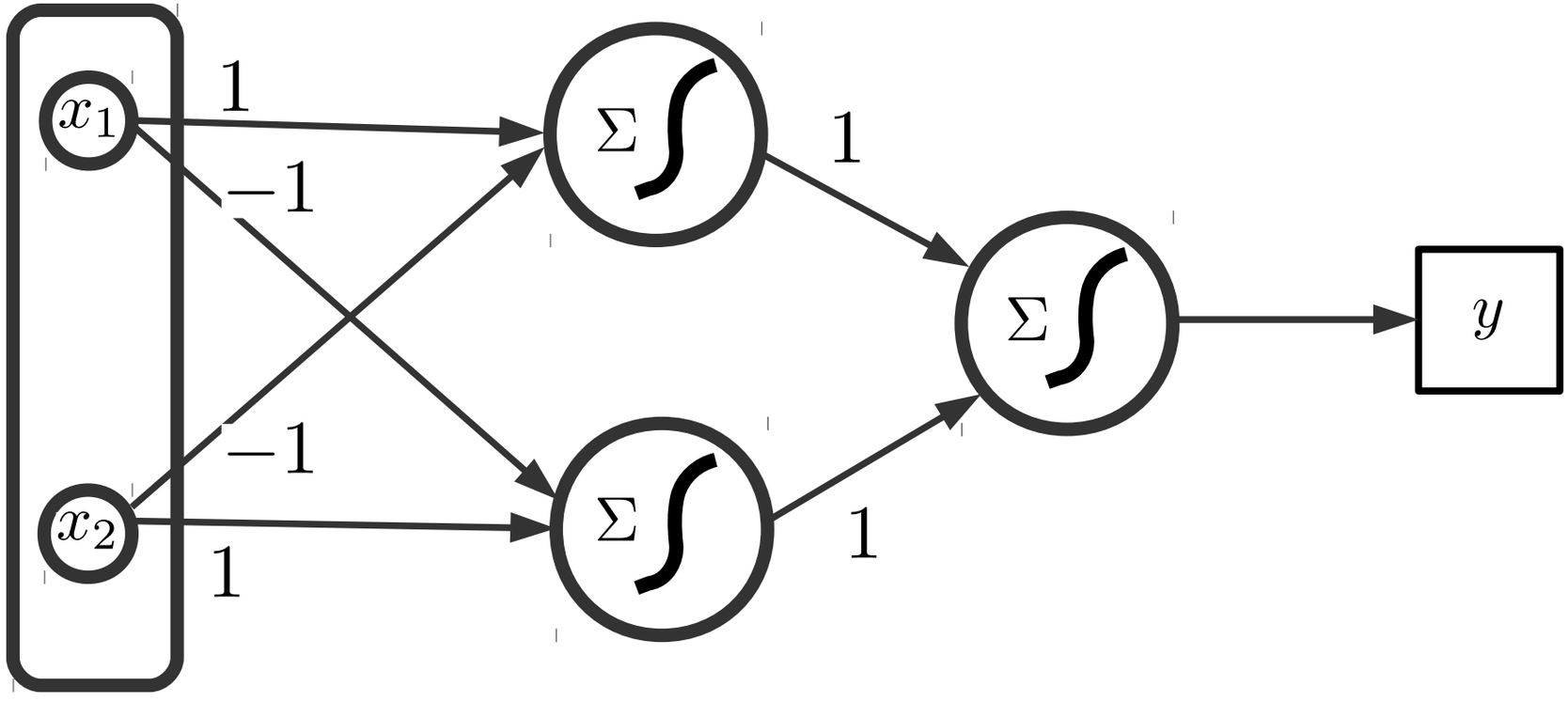
Representing XOR



$$y = \sigma\left(\sigma(x_1 W_{1,1} + x_2 W_{2,1}) U_1 + \sigma(x_1 W_{1,2} + x_2 W_{2,2}) U_2\right)$$

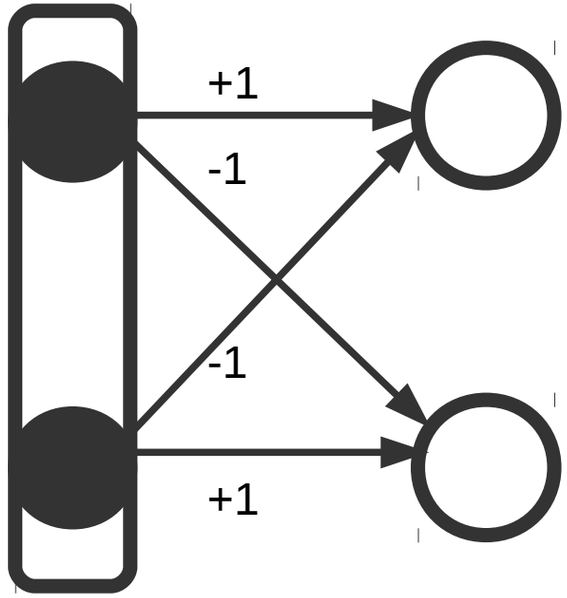
$$\text{where } \sigma(x) = \begin{cases} 1 & \text{if } x \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

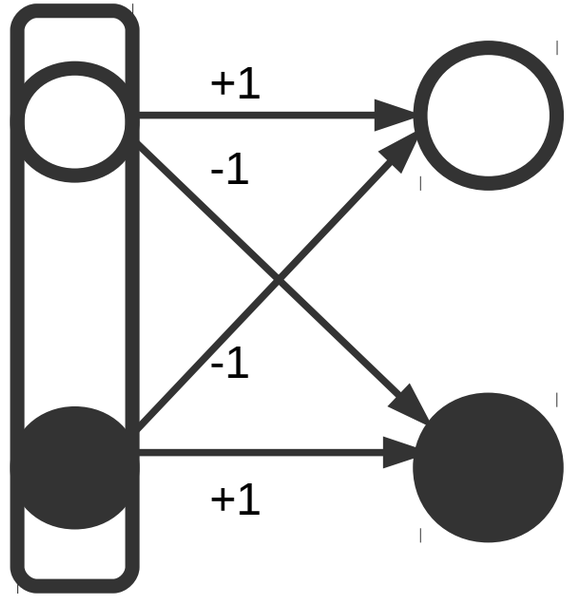
Representing XOR



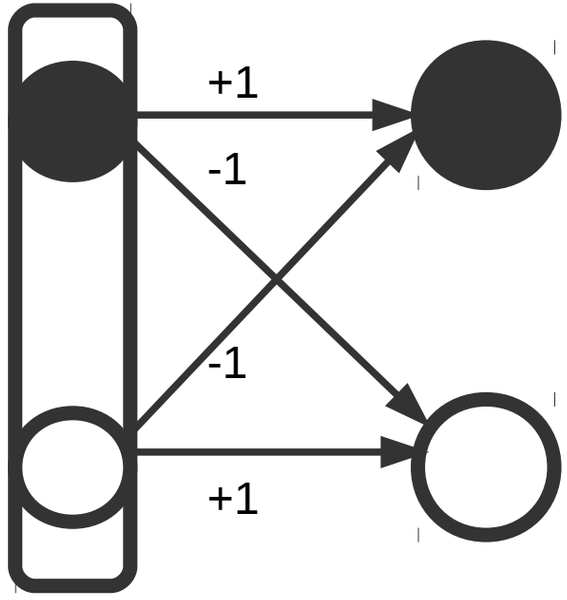
Calculate the activation of each unit for each of the four possible input patterns

Hidden layer as a feature detector

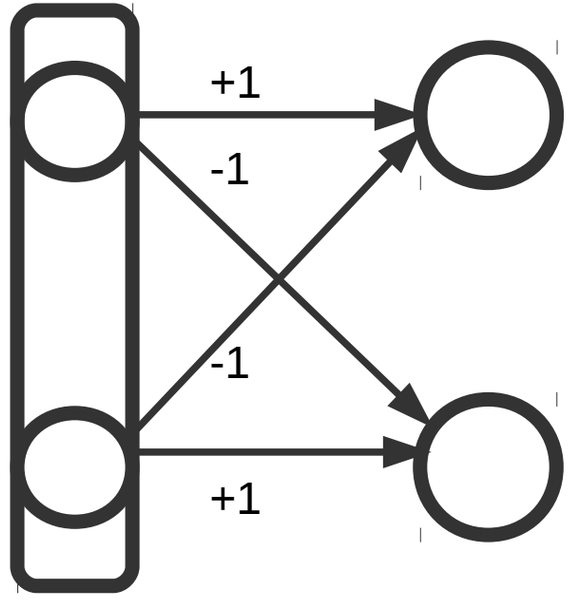




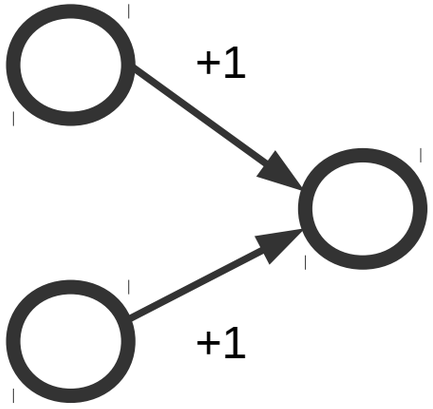
x_2 on, x_1 off

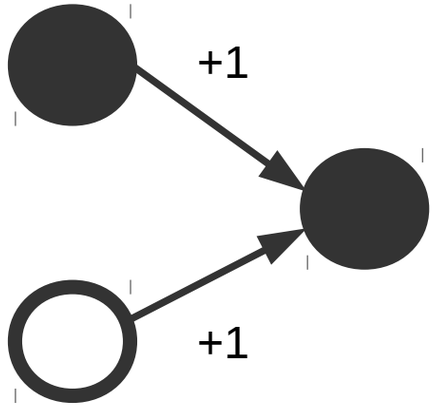


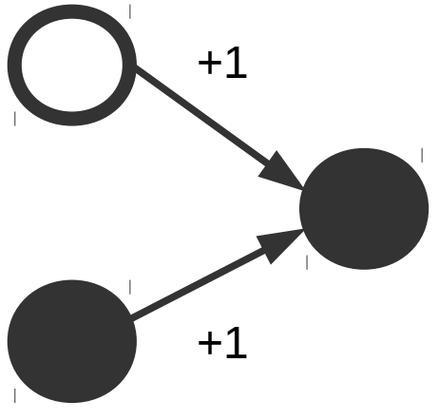
x_1 on, x_2 off



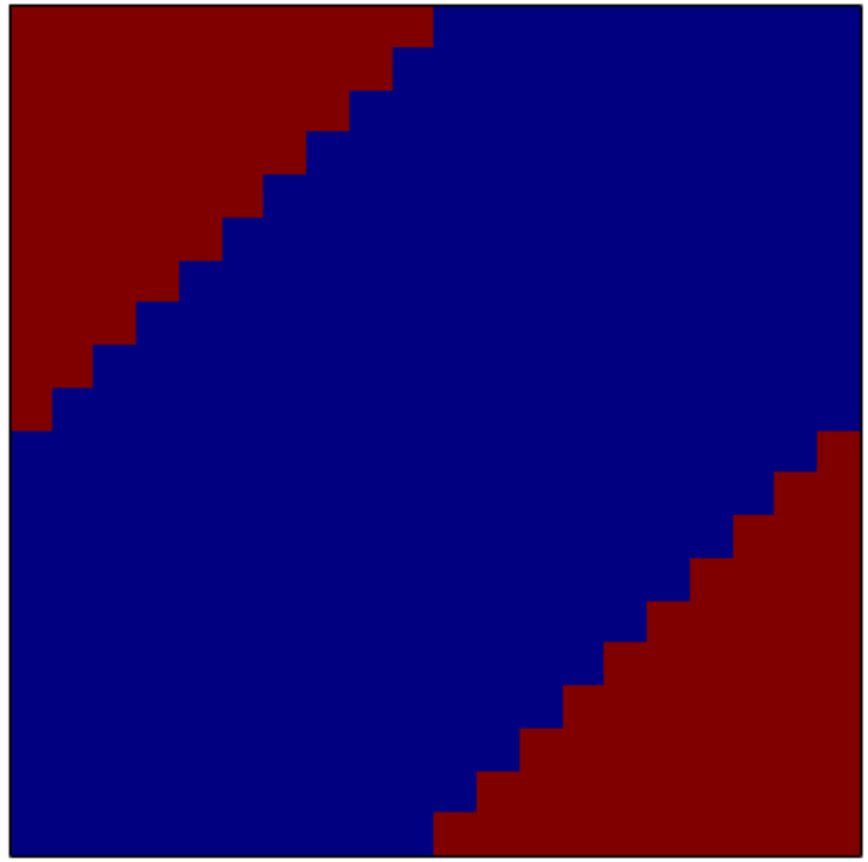
Hidden layer becomes input to output layer







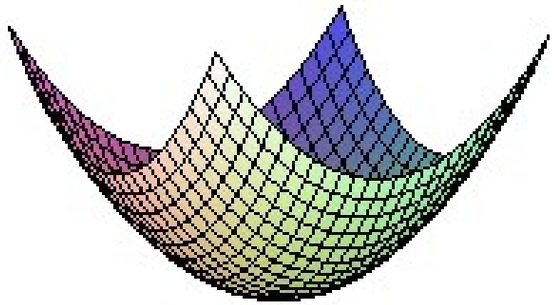
XOR classification

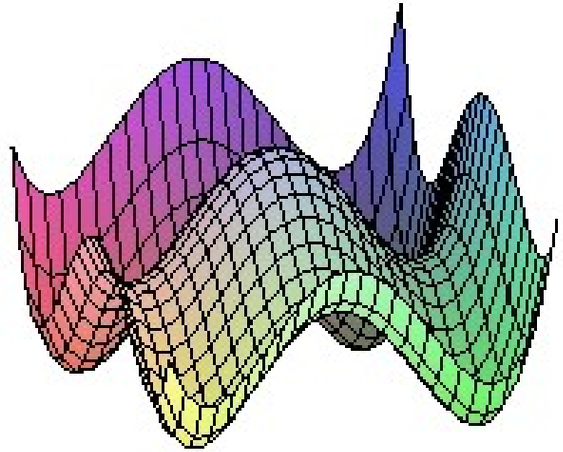


Finding weights

- Define error function
  - **log loss** or **squared loss**
- Find weights which minimize it
  - go down the gradient of the error function
  - for example **W** and **U** for this net: $y = \sigma(\mathbf{U}\sigma(\mathbf{W}\mathbf{x}))$
- known as **BACKPROPAGATION**

Non-convexity





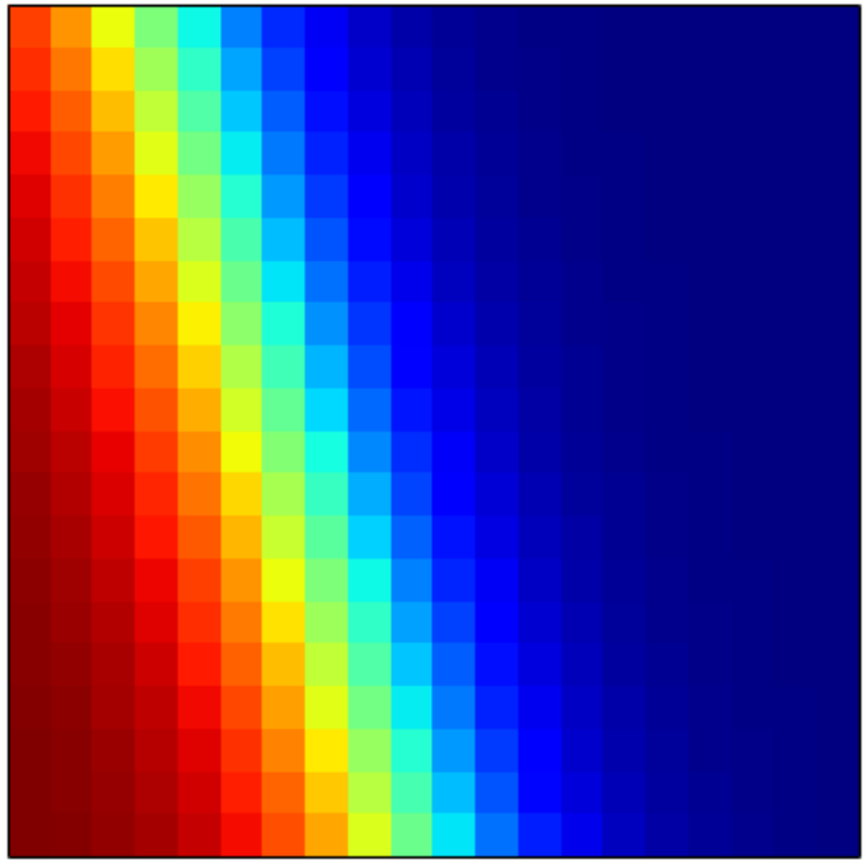
- Linear models have **convex** error functions
  - single global minimum
- NN's error function is **non-convex**
  - complicated curvature
  - local minima

Backpropagation important considerations

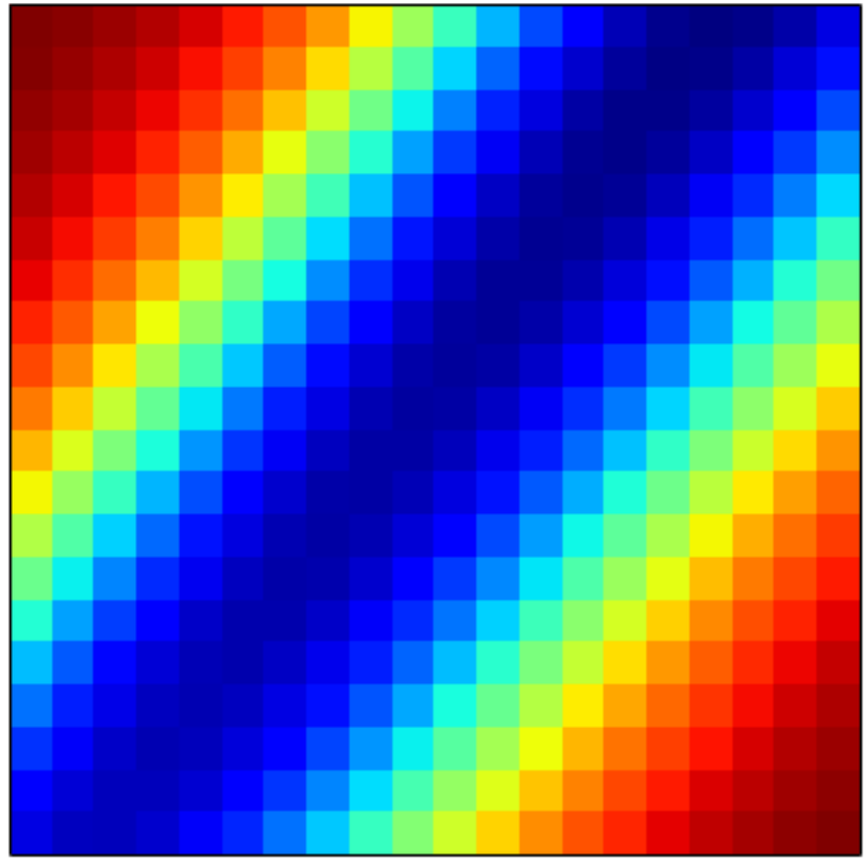
- Standardizing of input features
- Random initialization of weights
- Choice of activation function
- Adapting learning rate
- Number and size of hidden layers

Solutions found with SDG for the XOR problem

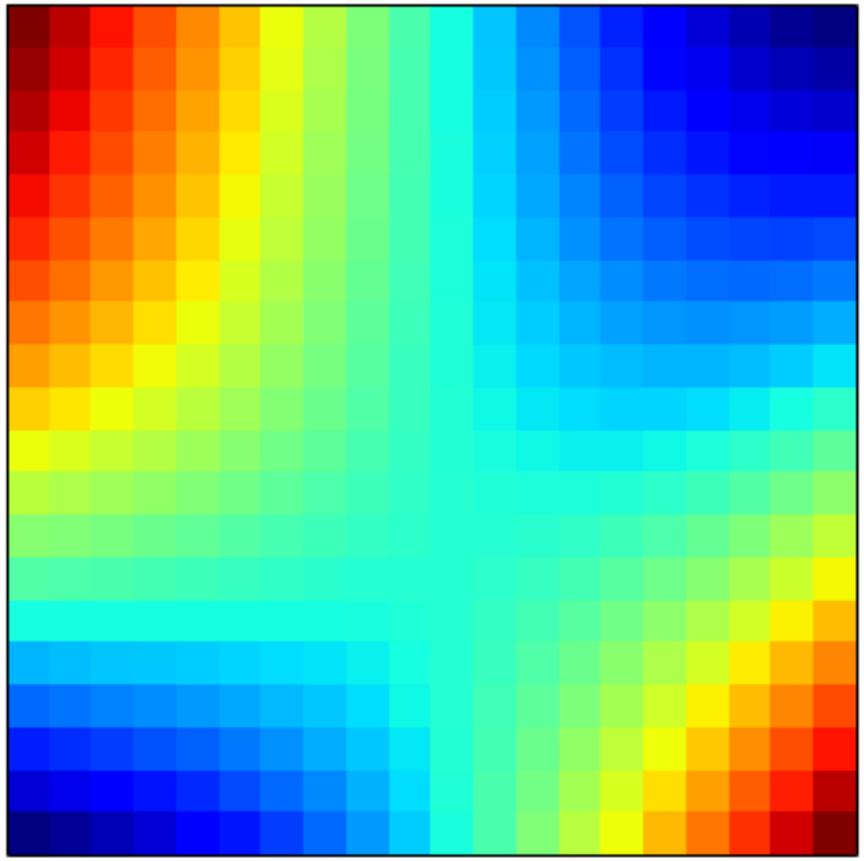
Logistic Regression



MLP (3 hidden units)



MLP (3 hidden units) – another solution

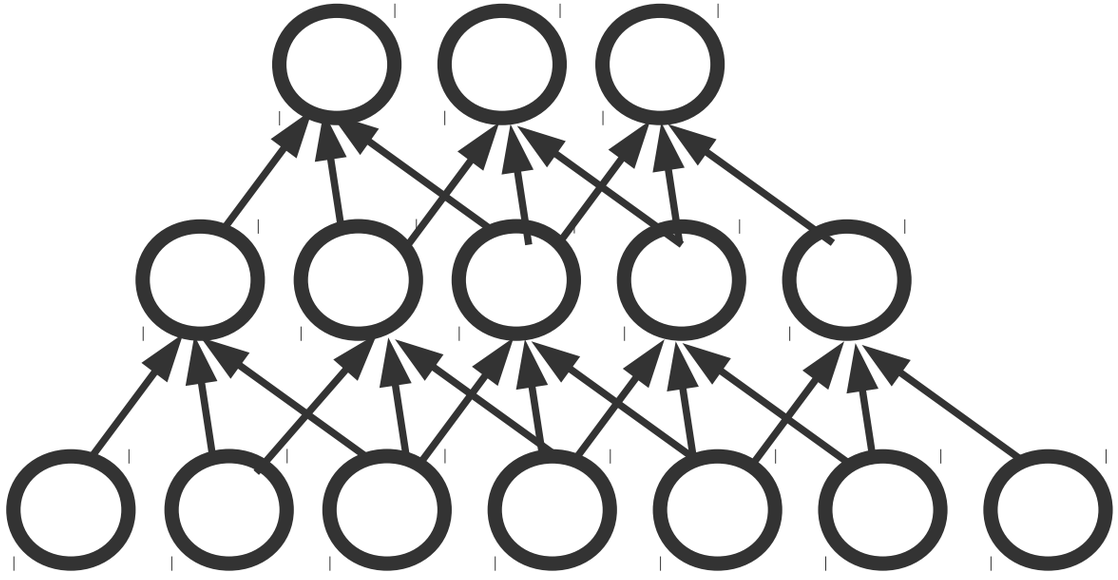


Other neural architectures

- Convolutional neural networks
  - Image and video applications
- Recurrent neural networks
  - Time-varying signals in general
  - speech and written text

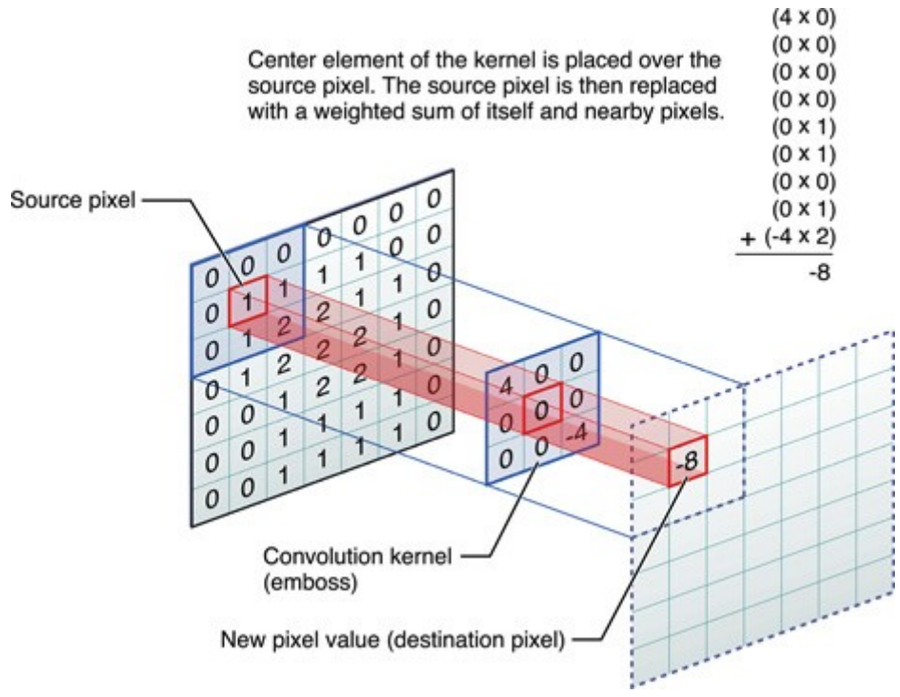
Convolutional networks

1D example – each unit connected to 3 units below



- Inspired by retina and receptive fields
- Neurons arranged in 2D pattern
- Units connected to spatially close units in layer below
- Weights are shared between units

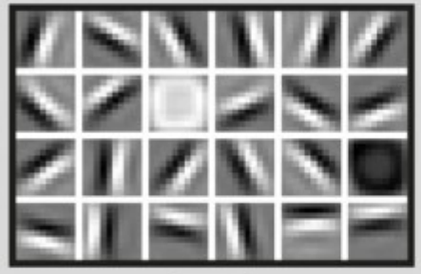
Convolutions, aka filters



FACIAL RECOGNITION

Deep-learning neural networks use layers of increasingly complex rules to categorize complicated shapes such as faces.

Layer 1: The computer identifies pixels of light and dark.

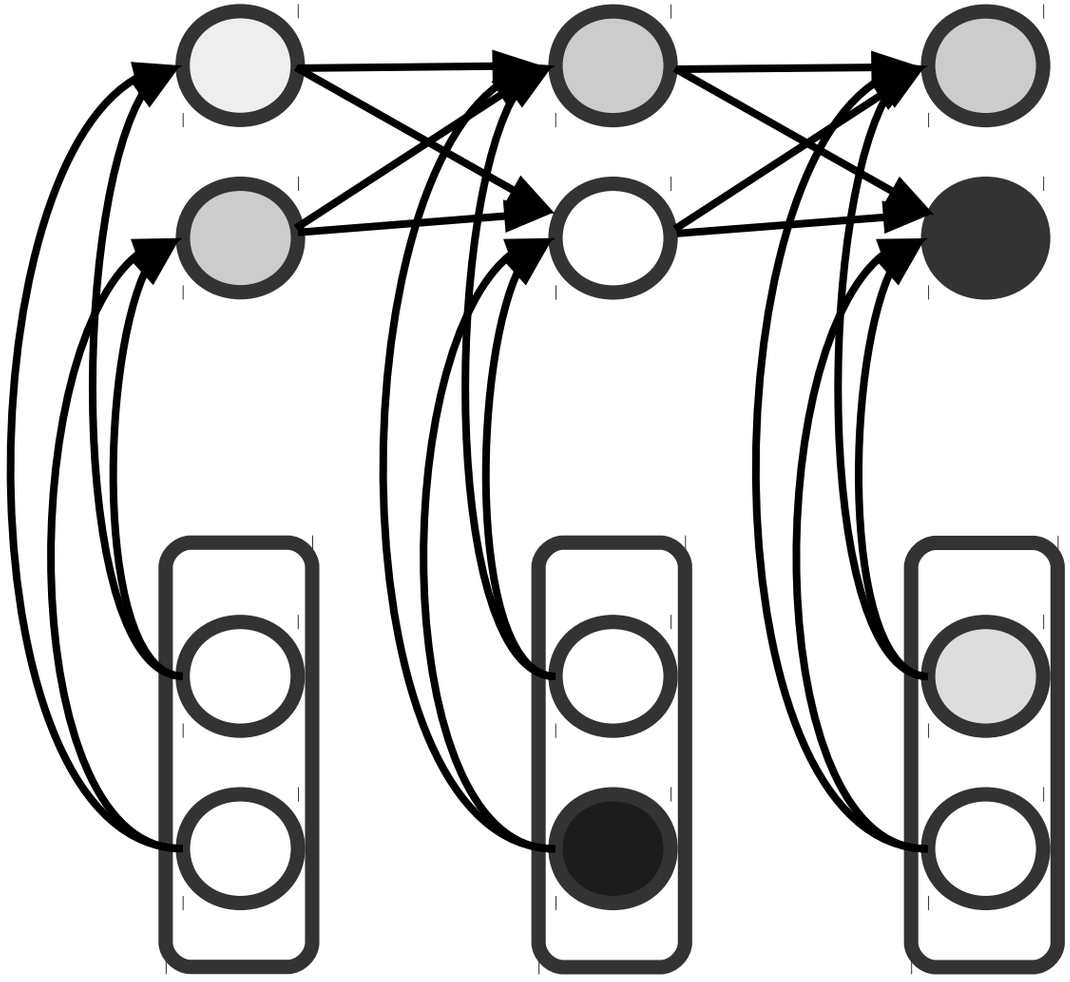


Layer 2: The computer learns to identify edges and simple shapes.

Layer 3: The computer learns to identify more complex shapes and objects.

Layer 4: The computer learns which shapes and objects can be used to define a human face.

Recurrent networks



- Dynamic hidden layer(s) evolve in time
- Weights are shared at each time-step

String	Nearest neighbors in embedding space			
should h	should d	will s	will m	should a
@justth	@neenu	@raven_	@lanae	@despic
maybe	u maybe y	cause i	wen i	when i

Multimodal networks

- Different types of networks can be combined
- Convolutional for images, recurrent for language

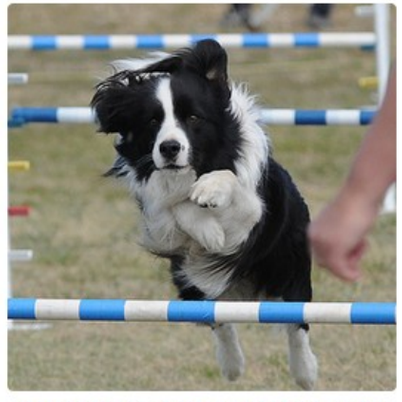
Describing images

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"girl in pink dress is jumping in air."



"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

Karpathy et al 2014

Imagining sentences

“A woman is guiding a brown dog around an obstacle course”

Work in progress with Akos Kadar and Afra Alishahi

Neural networks in practice

- Try a simpler (linear) model first
- Only start thinking about NNs if
  - you have lots of training data
  - have lots of time or a GPU
- Spend time learning how to train them

Neural Nets in Python

- PyBrain http://pybrain.org/
  - easy to install and use
  - mature
  - slow

- Theanets https://github.com/lmjohns3/theanets
  - many dependencies
  - impenetrable error messages
  - faster

Image credits

- Convex and non-convex functions https://plus.maths.org/content/convexity
- Convolutions https://developer.apple.com/library/ios/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html
- Facial recognition http://www.nature.com/news/computer-science-the-learning-machines-1.14481
- Example captions from Karpathy et al 2014 http://cs.stanford.edu/people/karpathy/deepimagesent/