IE 678 Deep Learning

02 – Feedforward Neural Networks

Part 2: Feedforward Neural Networks

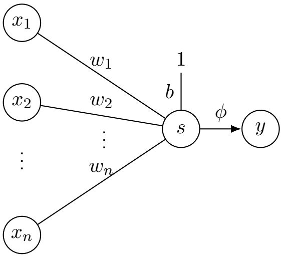
Prof. Dr. Rainer Gemulla

Universität Mannheim

Version: 2024-1

Artificial neuron

- An **artifical neuron** (AN) is a function $f : \mathbb{R}^n \to \mathbb{R}$
  - Inputs are vectors $\boldsymbol{x} \in \mathbb{R}^n$
  - Output is a value $y \in \mathbb{R}$, called **activation**
- f is taken from a family of functions that is parameterized by
  - A **weight vector** $\boldsymbol{w} \in \mathbb{R}^n$ (one weight per input)
  - A **bias** $b \in \mathbb{R}$
  - A **transfer function** or **activation function** $\phi : \mathbb{R} \to \mathbb{R}$
- Basic structure: $y = \phi(\boldsymbol{w}^\top \boldsymbol{x} + b)$
  - Computes the weighted sum $s = \boldsymbol{w}^\top \boldsymbol{x} + b$ of its inputs and bias
  - Passes s through the transfer function to obtain output y
  - ϕ can be deterministic or stochastic
- As before: bias can be replaced by an additional input $x_0 = 1$ and corresponding weight $w_0 = b$



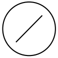
Types of artificial neurons

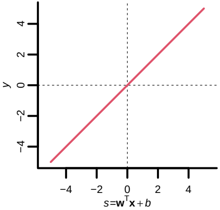
- The **type** of an AN is determined by its transfer function ϕ
- An AN of a given type can represent the family of functions

$$F_\phi = \left\{ \boldsymbol{x} \to \phi(\boldsymbol{w}^\top \boldsymbol{x} + b) \mid \boldsymbol{w} \in \mathbb{R}^n, b \in \mathbb{R} \right\}$$

- Each function in this family can be represented by its bias and weight vector
- We will see later that types are usually specified up-front, whereas weights are learned
- The simplest type of neuron is a **constant neuron**
  - No inputs; output fixed value $x \in \mathbb{R}$
  - Notation (from now on): x

Example: Linear neuron / identity

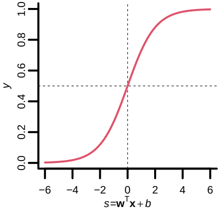
- Uses $\phi(s) = s$
- Notation: 



- Simple but computationally limited
- We often but not always want non-linear transfer functions

Example: Logistic neuron

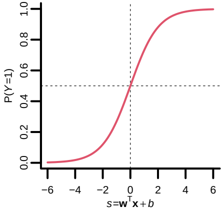
- Use logistic function $\phi(s) = \sigma(s) \stackrel{\text{def}}{=} \frac{1}{1+\exp(-s)}$
- Notation: 



- Gives a real-valued output that is smooth and bounded in $[0, 1]$
  - Negative activations mapped to value < 0.5
  - 0 activation mapped to 0.5
  - Positive activation mapped to value > 0.5
- Non-linear

Example: Stochastic binary neuron

- Also use logistic function
- But output of the logistic function is treated as a probability of producing a spike (1)
- I.e,. $\phi(s) = \begin{cases} 1 & \text{with probability } \sigma(s) \\ 0 & \text{otherwise} \end{cases}$



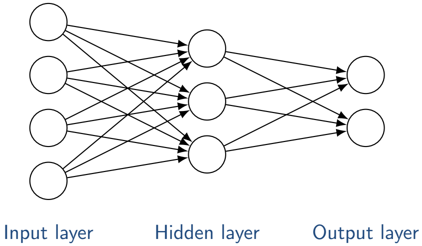
- Defines a probability distribution over outputs
- Other neurons can also be made stochastic

What is an artificial neural network?

- A network of artificial neurons
  - A set of (artificial) neurons
  - Connections between neurons (directed or undirected)
- Many different **architectures**
  - How many neurons? Of which type?
  - Are there **output neurons**?
  - Are there **hidden neurons** (neither input nor output)?
  - Which neurons are connected?
  - Are connections directed or undirected?
  - Are there cycles?
- Picking the right architecture for the problem at hand is important and requires skill/thought/compute power
  $\rightarrow$ **Architecture engineering**
- Can represent a wide range of functions (*universal approximation theorem*)

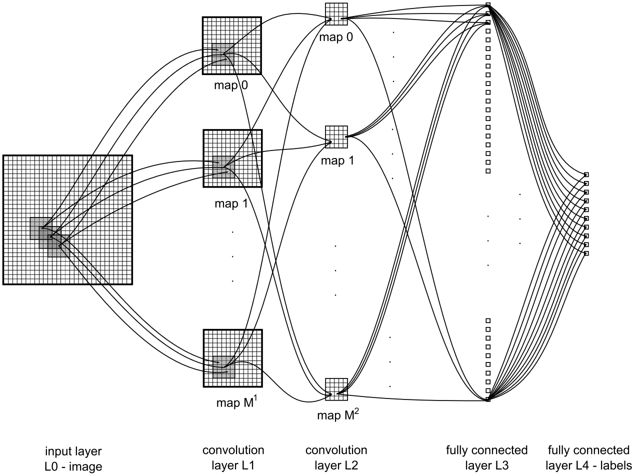
Feedforward neural networks

- A **feedforward neural network** (FNN) is an ANN in which
  - All connections are directed, and
  - There are no cycles (i.e., forms a DAG)
- Neurons usually grouped in **layers**
  - Input neurons: no incoming edges (first layer)
  - Output neurons: no outgoing edges (last layer)
  - Hidden neurons: all others (layer = maximum distance from input)
  - Layers do not need to be fully connected
  - Traditionally: edges only between subsequent layers
    (but: edges that skip layers are allowed, too)
- Example: an FNN with one hidden layer (omitting bias inputs)



MNIST, best performer (2011), architecture

Deep convolutional neural network (no preprocessing)



Learning

- Once we settled on an architecture, we need to learn connection strengths (weights)

- Simple approach
  - Supervised learning with labeled training examples
  - Use a suitable notion of model performance (e.g., loss function, likelihood)
  - Learn all weights jointly
    $\rightarrow$ As before: ERM/RRM, MLE/MAP, Bayesian inference
  - Most common: empirical/regularized risk minimizaton

- We will see: simple approach is often "not enough"
  - High model complexity
  - Limited (labeled) training data

- Values of hidden units can be thought of as features, but which features are good is unknown and needs to be learned
  - This makes learning hard
  - Note: embeddings typically refer to the values of a "certain" hidden layer in a larger FNN (more later)

FNNs, more generally

- Choice of neuron type(s) important
  - Influences expressability
  - Influences "learnability"
  - Many more types of artificial neurons have been proposed
- Neurons may not follow the template discussed here
  - E.g., product neuron $y = \prod_i x_i$
  - E.g., max-pooling neuron $y = \max_i(x_i)$
- Layers may be more general
  - I.e., any parameterized function from $\mathbb{R}^n$ to $\mathbb{R}^m$
  - May compute multiple outputs jointly (e.g., softmax layer)
  - May have additional internal structure (e.g., Transformer layers)
- Will see: generally, FNNs represented as a "compute graph"