Artificial Neural Networks
Lecture Notes

Prof. Dr. Sen Cheng

Winter Semester 2019/20

7 Artificial neural networks

7.1 Components of artifical neural networks

ANN were originally inspired by the architecture and functioning of the brain. Hence the name. The basic building blocks of ANN are neuron-like units. A unit i is connected to multiple input units j's via connections that have different weights w_{ij} (Fig. 1). [Note that the meaning of the indices i and j here are different from the previous usage.] The input units send a signal h_j^{l-1}. The unit sums the weighted inputs

$$a_i^l = \sum_j w_{ij} h_j^{l-1} + b_i^{l-1} \tag{1}$$

and applies an *activation function* ϕ to compute its activation $h_i^l = \phi(a_i)$.

$$h_i^l = \phi^l \left(\sum_j w_{ij}^{l-1} h_j^{l-1} + b_i^{l-1} \right) \tag{2}$$

b_i^{l-1} is the bias of unit i. All weights and biases in all the layers are considered the parameters θ of the neural network,

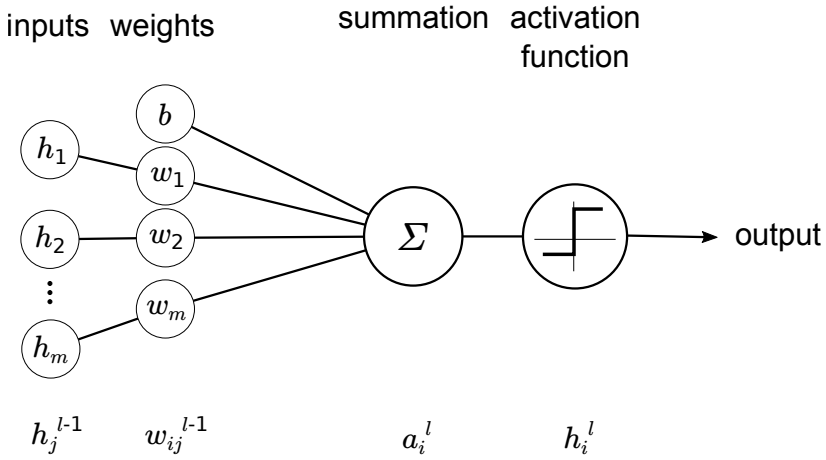


Figure 1: Units used in neural networks.

which are learned during optimization. They are essential for the function that the network represents.

The meaning of the index l differs depending on the architecture of the network. Taking another inspiration from the brain, many units are connected to each other in a network. There are two general classes of neural networks: *feedforward* and *recurrent* networks (Fig. 2). In feedforward networks, all units can be arranged in layers such that information processing proceeds strictly from layer $l-1$ to l (Fig. 2). This means that the activity propagates through the network in one pass, i.e., all units perform the computation in Eq. 2 once for each input. The first layer $l = 1$ is called the *input layer*, their activity is set to the input $h^1 = x$. The intermediate layers are called the *hidden layers*.

The last layer $l = m$ is called the *output layer*, their activity represents the prediction of the network. The number of layers m and neurons n^l as well as the activation functions ϕ^l are considered hyperparameters.

In contrast to feedforward networks, information processing in recurrent networks can return to the same unit. In that case, the index l represents time and Eq. 2 describes how activity in one time step determines the network activity in the next. Inputs can be provided at the beginning and the network will continue its computation by itself. It is not so clear when the computations should stop. Generally the computations proceed until either 1. the network itself reaches a state that indicates "stop" or 2. the network reaches a steady state, in which either the activity of the units stops changing, or it might still change in individual units, but the mean remains constant. We'll discuss recurrent networks later and focus on feedforward networks first.

7.2 Activation function

The activation function is inspired by the threshold behavior of biological neurons, but also plays an essential role in ANN. It must be nonlinear in any complex neural network (Fig. 3). More on that below. The model with a threshold function

$$\tau(t) = \begin{cases} 0 & t \leq 0 \\ 1 & t > 0 \end{cases} \tag{3}$$

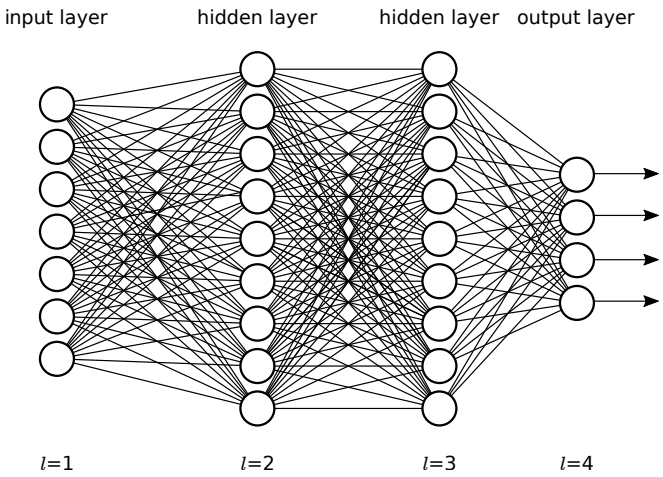
as activation function, $\phi(t) = \tau(t)$ is called a McCulloch–Pitts neuron (McCulloch & Pitts, 1943).

7.2.1 Nonlinearity

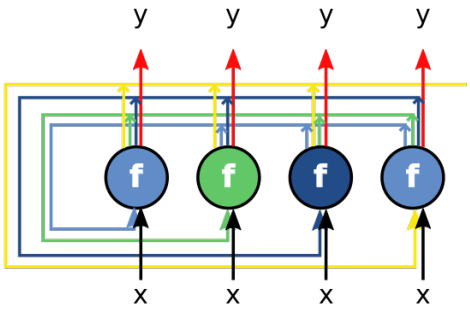
It doesn't make sense to have more than one layer with a linear activation function. Since each layer applies a function f^l on its inputs, we can write the computation of the entire network as:

$$\hat{y} = f(x) = f^m(f^{m-1}(\cdots f^2(x)\cdots)). \tag{4}$$

If two activation functions ϕ^l and ϕ^{l-1} were linear, $f^l \circ f^{l-1}$ would also be a linear function, since a linear function of a linear function is also a linear function. In that case, the two layers could be replaced by a single layer without losing any functionality. If all activation functions, were linear then the composition $f^m \circ f^{m-1} \circ \cdots \circ f^2$ would be a linear functin of the input x. That means that the network would simply represent a linear function $y = f(x)$. That would be equivalent to linear regression, which we solved before. That's why any meaningful neural network has to use nonlinear activation functions. However, that nonlinearity means that the optimization problem to fit the model parameters is much harder than in linear regression. That's why even though the idea of neural networks sketched above had been around since the 1940's, nobody knew until the 1980's how to train such neural networks.



(a) Feedforward network.



(b) Recurrent network.

Figure 2: Examples of artifical neural networks.

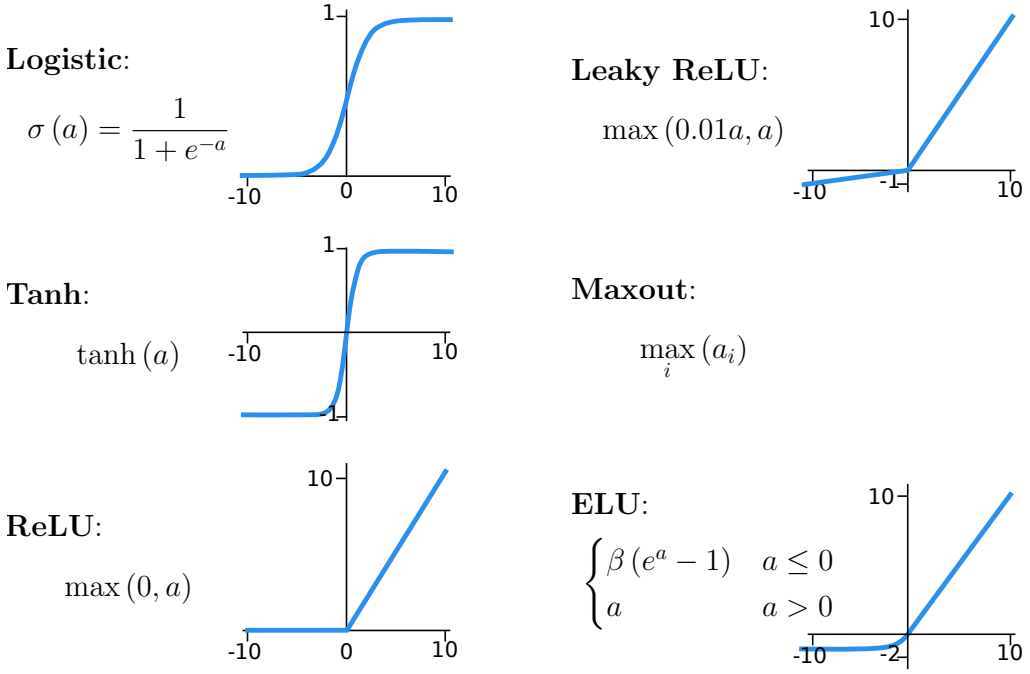


Figure 3: Activation functions. ReLU: Rectified Linear Unit. ELU: Exponential Linear Unit.

7.3 Comparison between biological neurons and ANN

Superficially, biological neurons and units in an ANN appear to be rather similar: they both integrate the signal arriving from other units, apply a non-linear activation function and send an output to other units in a network. This is, of course, not an coincidence since ANN were defined the way they were to model biological neurons. However, there are some fundamental differences between them, so that ANN are no longer thought to be good models of the brain.

1. ANN units perform computations and transmit information instantaneously, whereas neurons use complex dynamical processes.
2. ANN units are synchronous, whereas neurons are asynchronous.
3. Summation of inputs in ANN is uniform and linear, whereas it is neither uniform nor linear in biological neurons (*dendritic processing*).
4. The output of an ANN unit is continuous, whereas neurons generate discrete spikes.
5. The weights in an ANN can be changed arbitrarily, e.g. gradient descent, whereas synapses can use only local information and are constrained by biological processes.
6. ANN usually have separate training and test phases, where weight are adjusted and constant, respectively. Plasticity cannot be switched off in a biological neuron.
7. The input-output response of biological neurons change in time (*adaptation*) and is changed by neuromodulators.

Summary:

		Artificial NN	**Biological NN**
1	Computations, Transmission	Instantaneous	Complex dynamics
2	Synchroneous	Yes	No
3	Input summation	Homogeneous, linear	Inhomogeneous, nonlinear
4	Output	Continuous variable	Discrete spikes
5	Weight changes	Arbitrary	Local, biological constraints
6	Training/Testing	Separate	Concurrent
7	Stability over time	Yes	No: adaptation, neuromodulation

7.4 Universal approximation theorem

Why should we bother training a feedforward neural network? Because they are powerful function approximators! In fact, they can approximate any continuous functions, under some mild conditions.

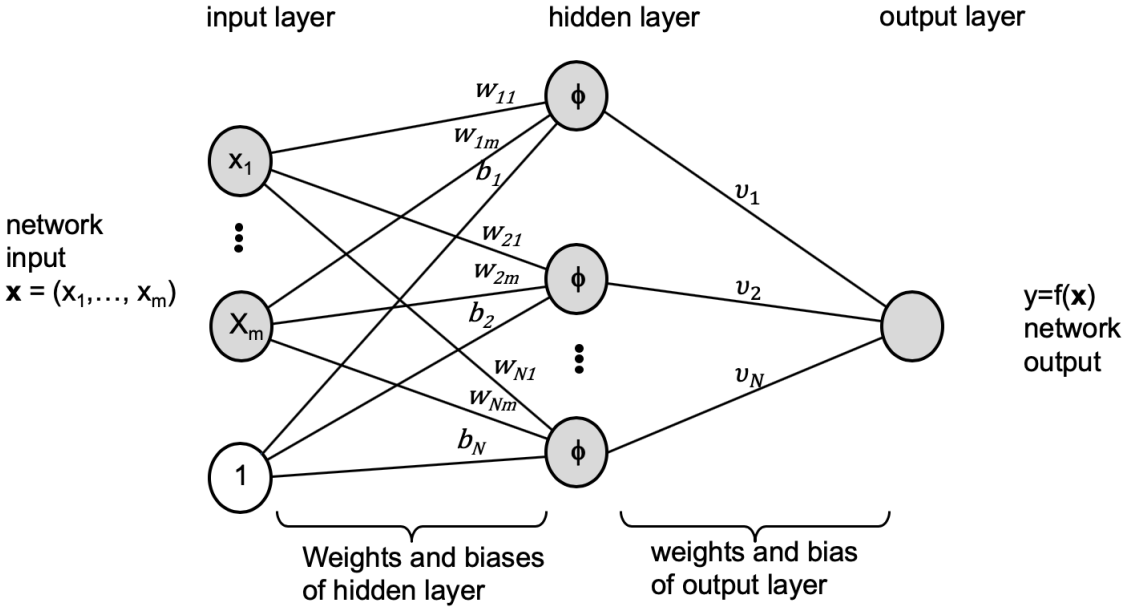


Figure 4: Neural network in the universal approximation theorem with one hidden layer. Note that the activation function is only applied to the hidden layer, the output layer computes only a weighted sum.

Theorem 1 *Let $\phi : R \to R$ be a nonconstant, bounded, and continuous function (the activation function). Let I_m denote a compact subset of R^m. The space of real-valued continuous functions on I_m is denoted by $C(I_m)$. Then, given any $\varepsilon > 0$ and any function $g \in C(I_m)$, there exist an integer N, real constants $v_i, b_i \in R$ and real vectors $w_i \in R^m$ for $i = 1, \ldots, N$, such that we may define:*

$$f(x) = \sum_{i=1}^{N} v_i \phi\left(w_i^T x + b_i\right) \tag{5}$$

as an approximation of the function g; that is,

$$|f(x) - g(x)| < \varepsilon \tag{6}$$

for all $x \in I_m$.

It is hard to believe that a simple 2-layer network with one hidden layer and linear output layer (Fig. 4) is so powerful, but this theorem has been proven mathematically. Unfortunately, the various proofs are not constructive, i.e., they show that the approximation is always possible, but don't tell us how to find the right network parameters. Here, we will only sketch an intuitive proof of the theorem.

1. An ANN with a threshold function as activation function, $\phi(x) = \tau(x + b_i)$, can approximate any function $g(x)$. Two elements can be summed to produce a localized bar Fig. 5. Then pairs can be added to produce multiple bars of different heights at different locations.

2. Any sigmoid function $\phi(x) = \sigma(m_i x + b_i)$ can approximate a threshold function by choosing very large m_i.

3. Therefore replacing the threshold activation function in the network by a sigmoid function also approximates the function $g(x)$. This substituion increases the error, but since it's bounded, the new network still provides an appropriate approximation.

This argument works in 2 or more dimensions as well (Fig. 6), just the construction will become more and more complex. The above illustration only works for sigmoid-type activation functions, but other constructions are possible for nonsigmoid activation functions.

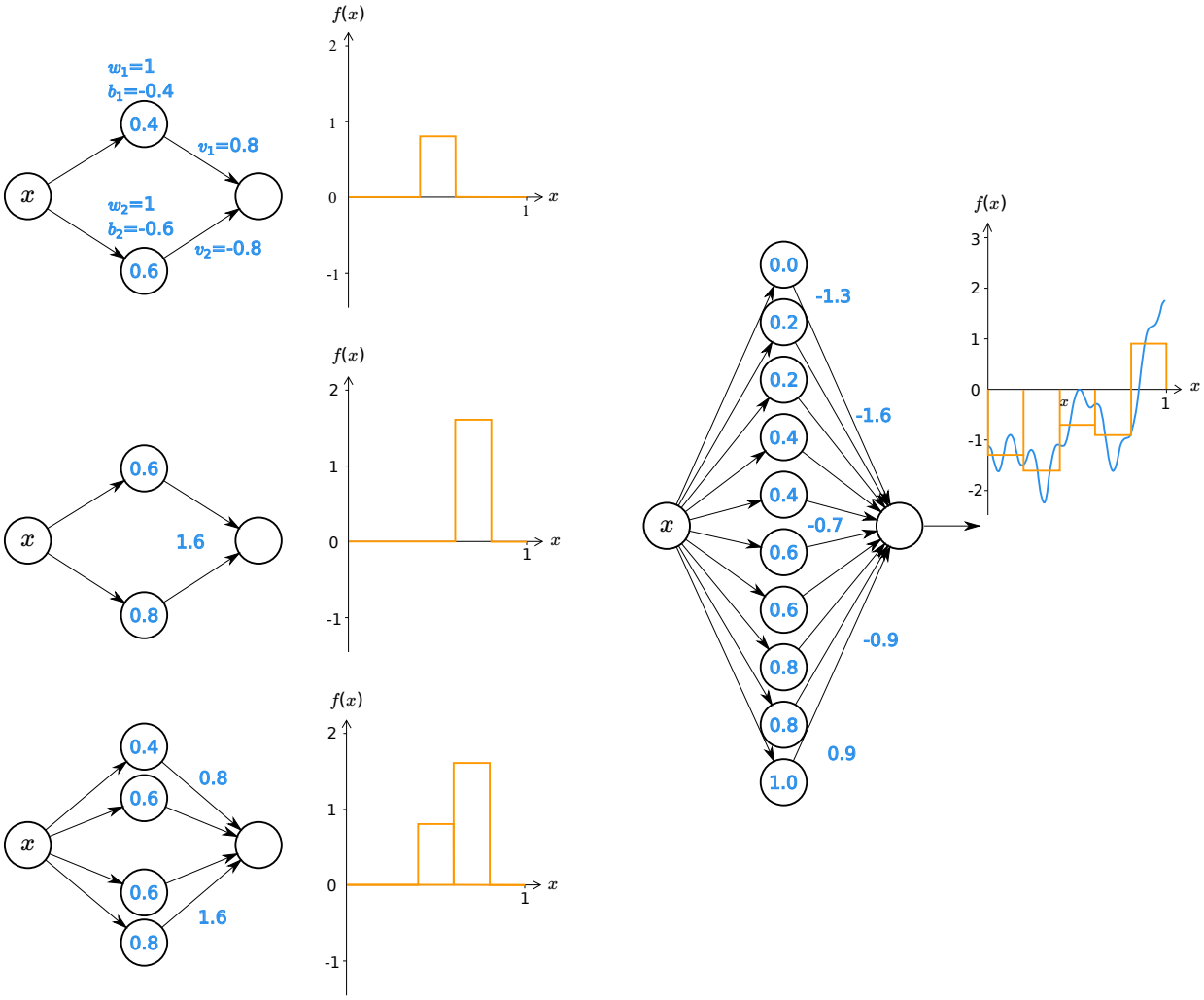


Figure 5: Function approximation in 1-d using splines.

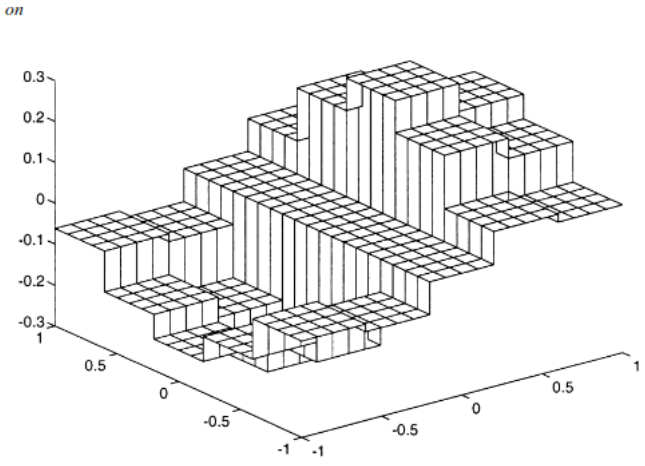


Figure 6: Function approximation in 2 dimensions.

References

McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5*(4), 115–133.