14. Neural networks

- introduction
- single-neuron training
- the backpropagation algorithm

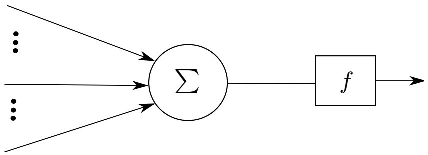
Neural network success

neural networks have found tremendous success in many real life applications such as

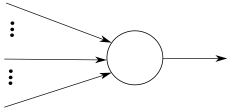
- speech recognition
- image classifications
- recommendations systems
- cancer cell detection
- ...etc

Neuron

an *artificial neural network* is a system composed of interconnected simple subsystems called *neurons*

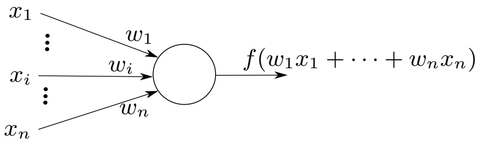


- neuron symbol:



- the output of the neuron is a function of the sum of the inputs
- the function f at the output is called the *activation function*

Single-neuron output



the output of a single-neuron can be represented by the map from $\mathbb{R}^n \to \mathbb{R}$:

$$y = f\left(\sum_{i=1}^{n} w_i x_i\right) = f(\boldsymbol{x}^T \boldsymbol{w}) \quad (14.1)$$

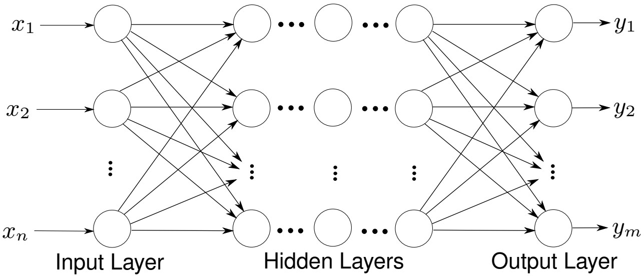
- f is the activation function
- w_i is the weight multiplied by input x_i
- $\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$ the vector of inputs
- $\boldsymbol{w} = (w_1, \dots, w_n) \in \mathbb{R}^n$ is the vector of weights

Activation functions

- *Linear* (no activation): $f(v) = v$
- *Softplus*:
$$f(v) = \log(1 + e^v)$$
- *Sigmoid or logistic, soft step*:
$$f(v) = \frac{1}{1 + e^{-v}}$$
- *Binary step*:
$$f(v) = \begin{cases} 0 & v \leq 0 \\ 1 & v > 0 \end{cases}$$
- *Rectified linear unit* (ReLU):
$$f(v) = \begin{cases} 0 & v \leq 0 \\ v & v > 0 \end{cases}$$

Feedforward neural network

in a *feedforward neural network*, the neurons are interconnected in layers and the data flow in only one direction



- the first layer in the network is called the *input layer*
- the last layer is called the *output layer*
- the layers in between the input and output layers are called *hidden layers*

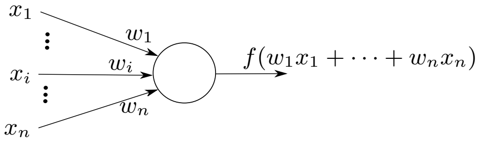
Training a neural network

- a neural network is a mapping from $\mathbb{R}^n$ to $\mathbb{R}^m$, where n is the number of inputs $x_1, \ldots, x_n$ and m is the number of outputs $y_1, \ldots, y_m$
- suppose that we are given a map $F : \mathbb{R}^n \to \mathbb{R}^m$ that we wish to approximate by a given neural network
- then, a neural network woth apprpiate weights our task boils down to selecting the interconnection weights in the network appropriately; this task is referred to as *training* of the neural network or *learning* by the neural network
- input-output data of the given map are used to train the neural network
- we train the neural network by adjusting the weights such that the map that is implemented by the network is close to the desired map F

Outline

- introduction
- **single-neuron training**
- the backpropagation algorithm

Single neuron training



- we wish to find the value of the weights $w_1, \ldots, w_n$ such that the neuron approximates a given map $F : \mathbb{R}^n \to \mathbb{R}$ map F as closely as possible
- we are given a training set consisting of p pairs $\{(\boldsymbol{x}_{d,1}, y_{d,1}), \ldots, (\boldsymbol{x}_{d,p}, y_{d,p})\}$, where $\boldsymbol{x}_{d,i} \in \mathbb{R}^n$ and $y_{d,i} \in \mathbb{R}, i = 1, \ldots, p$
- for each i, $y_{d,i} = F(\boldsymbol{x}_{d,i})$ is the "desired" output corresponding to the given input $\boldsymbol{x}_{d,i}$

Optimization formulation

$$\text{minimize} \quad (1/2)\sum_{i=1}^{p}\left(y_{d,i} - f(\boldsymbol{x}_{d,i}^T \boldsymbol{w})\right)^2$$

- variable $\boldsymbol{w} = (w_1, \ldots, w_n) \in \mathbb{R}^n$
- the choice of the method typically depends on the activation function f

Example: when f is the identity function, then the problem becomes

$$\text{minimize} \quad (1/2)\sum_{i=1}^{p}\left(y_{d,i} - \boldsymbol{x}_{d,i}^T \boldsymbol{w}\right)^2,$$

which is just a least squares problem:

$$\text{minimize} \quad (1/2)\|\boldsymbol{y}_d - \boldsymbol{X}_d^T \boldsymbol{w}\|^2$$

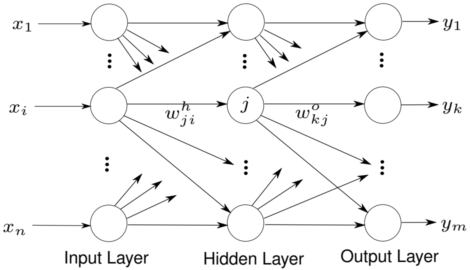
where

$$\boldsymbol{X}_d = [\boldsymbol{x}_{d,1} \cdots \boldsymbol{x}_{d,p}] \in \mathbb{R}^{n \times p} \quad \text{and} \quad \boldsymbol{y}_d \quad = (y_{d,1}, \ldots, y_{d,p}) \in \mathbb{R}^p$$

Outline

- introduction
- single-neuron training
- **the backpropagation algorithm**

Three-layered neural network



- n inputs x_i, $i = 1, \ldots, n$, and m outputs y_s, $s = 1, \ldots, m$
- l neurons in the hidden layer; the outputs of the neurons in the hidden layer are z_j, where $j = 1, \ldots, l$
- the inputs $x_1, \ldots, x_n$ are distributed to the neurons in the hidden layer
- we let $f^h_j : \mathbb{R} \to \mathbb{R}$ denote the activation functions of the neurons in the hidden layer where $j = 1, \ldots, l$, and f^o_s the activation functions of the neurons in the output layer by, where $s = 1, \ldots, m$

Mapping representation

given the hidden layers weights w^h_{ji} and the output layer weights w^o_{sj}, let us denote the input to the jth neuron in the hidden layer by v_j and the output of the jth neuron in the hidden layer by z_j; then, we have

$$v_j = \sum_{i=1}^{n} w^h_{ji} x_i,$$

$$z_j = f^h_j\Big(\sum_{i=1}^{n} w^h_{ji} x_i\Big)$$

the output from the sth neuron of the output layer is

$$y_s = f^o_s\left(\sum_{j=1}^{l} w^o_{sj} z_j\right)$$

therefore, the relationship between the inputs $x_i, i = 1, \ldots, n$, and the sth output y_s is given by

$$\begin{aligned} y_s &= f_s^o \left(\sum_{j=1}^{l} w_{sj}^o f_j^h \left(v_j \right) \right) \\ &= f_s^o \left(\sum_{j=1}^{l} w_{sj}^o f_j^h \left(\sum_{i=1}^{n} w_{ji}^h x_i \right) \right) \\ &= F_s \left(x_1, \ldots, x_n \right) \end{aligned}$$

the overall mapping that the neural network implements is therefore given by

$$\left[\begin{array}{c} y_1 \\ \vdots \\ y_m \end{array} \right] = \left[\begin{array}{c} F_1 \left(x_1, \ldots, x_n \right) \\ \vdots \\ F_m \left(x_1, \ldots, x_n \right) \end{array} \right]$$

Training the neural network

- we are given the training set $(\boldsymbol{x}_d, \boldsymbol{y}_d)$, where $\boldsymbol{x}_d \in \mathbb{R}^n$ and $\boldsymbol{y}_d \in \mathbb{R}^m$
- the training of the neural network involves adjusting the weights of the network such that the output generated by the network for the given input $\boldsymbol{x}_d = (x_{d1}, \ldots, x_{dn})$ is as close to $\boldsymbol{y}_d$ as possible

The training problem

$$\text{minimize} \quad (1/2) \sum_{s=1}^{m} (y_{ds} - y_s)^2$$

- y_s, $s = 1, \ldots, m$, are the outputs of the neural network from the inputs $x_{d1}, \ldots, x_{dn}$
- this minimization is taken over

$$\boldsymbol{w} = \left\{ w_{ji}^h, w_{sj}^o : i = 1, \ldots, n, j = 1, \ldots, l, s = 1, \ldots, m \right\}$$

the neural network requires minimizing the objective function

$$E(\boldsymbol{w}) = (1/2)\sum_{s=1}^{m}\left(y_{ds} - y_s\right)^2$$

$$= (1/2)\sum_{s=1}^{m}\left(y_{ds} - f_s^o\left(\sum_{j=1}^{l} w_{sj}^o f_j^h\left(\sum_{i=1}^{n} w_{ji}^h x_{di}\right)\right)\right)^2.$$

- we can solve using the gradient method with stepsize η
- doing so leads to the backpropagation algorithm

The back-propagation algorithm

$$
\begin{aligned}
w_{sj}^{o(k+1)} &= w_{sj}^{o(k)} + \eta \delta_s^{(k)} z_j^{(k)} \\
w_{ji}^{h(k+1)} &= w_{ji}^{h(k)} + \eta \left(\sum_{p=1}^{m} \delta_p^{(k)} w_{pj}^{o(k)} \right) f_j^{h'} \left(v_j^{(k)} \right) x_{di}
\end{aligned}
$$

where η is the (fixed) step size and

$$
\begin{aligned}
v_j^{(k)} &= \sum_{i=1}^{n} w_{ji}^{h(k)} x_{di} \\
z_j^{(k)} &= f_j^h \left(v_j^{(k)} \right) \\
y_s^{(k)} &= f_s^o \left(\sum_{q=1}^{l} w_{sq}^{o(k)} z_q^{(k)} \right) \\
\delta_s^{(k)} &= \left(y_{ds} - y_s^{(k)} \right) f_s^{o'} \left(\sum_{q=1}^{l} w_{sq}^{o(k)} z_q^{(k)} \right)
\end{aligned}
$$

- the reason for the name backpropagation is that the output errors $\delta_1^{(k)}, \ldots, \delta_m^{(k)}$ are propagated back from the output layer to the hidden layer
- and are used in the update equation for the hidden layer weights
- **forward pass of the algorithm:** using the inputs x_{di} and the current set of weights, we first compute the quantities $v_j^{(k)}$, $z_j^{(k)}$, $y_s^{(k)}$, and $\delta_s^{(k)}$, in turn
- **reverse pass of the algorithm:** compute the updated weights using the quantities computed in the forward pass

References and further readings

- Edwin KP Chong and Stanislaw H Zak. *An Introduction to Optimization,* John Wiley & Sons, 2013, chapter 13.