Introduction to inverse and ill-posed problems: linear models for regression. Regularized and non-regularized neural networks.

Lecture 10

Supervised learning

- Linear regression as part of supervised learning.
- Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal).
- A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances.

Linear regression

- The goal of regression is to predict the value of one or more continuous target variables t by knowing the values of input vector x.
- The linear regression is similar to the solution of linear least squares problem.
- We will revise solution of linear least squares problem in terms of linear regression.

Reference literature: Miroslav Kurbat, *An Introduction to Machine Learning*, Springer, 2017.

Christopher M. Bishop, *Pattern recognition and machine learning*, Springer, 2009.

L. Beilina, E. Karchevskii, M. Karchevskii, *Numerical Linear Algebra: Theory and Applications*, Springer, 2017 – see link to GitHub with Matlab code

Linear model for regression versus least squares problem

The simplest linear model for regression is

$$f(x,\omega) = \omega_0 \cdot 1 + \omega_1 x_1 + ... + \omega_N x_N, \tag{1}$$

Here, $\{\omega_i\}, i = 0, ..., N$ are weights with bias parameter ω_0, $\{x_i\}, i = 1, ..., N$ are observation points. Target values (known data) are $\{t_i\}, i = 1, ..., N$ which correspons to $\{x_i\}, i = 1, ..., N$. The goal is to predict the value of t for a new value of x.
The linear model (1) can be written in the form

$$f(x,\omega) = \omega_0 \cdot 1 + \sum_{i=1}^{N} \omega_i \varphi_i(x) = \omega_0 + \omega^T \varphi(x), \tag{2}$$

where $\varphi_j(x), j = 0, ..., N$ are known basis functions with $\varphi_0(x) = 1$.

Linear model for regression versus least squares problem

Our goal is to minimize the sum of squares

$$E(\omega) = \frac{1}{2}\sum_{n=1}^{N}(t_n - \omega^T\varphi(x_n))^2 := \frac{1}{2}\|t - \omega^T\varphi(x)\|_2^2. \tag{3}$$

to find

$$\min_{\omega} \frac{1}{2}\|t - \omega^T\varphi(x)\|_2^2 \tag{4}$$

The problem (4) is a typical least squares problem of the minimizing the squared residuals

$$\min_{\omega} \frac{1}{2}\|r(\omega)\|_2^2 = \min_{\omega} \frac{1}{2}\|t - \omega^T\varphi(x)\|_2^2 \tag{5}$$

with the residual $r(\omega) = t - \omega^T\varphi(x)$.

Linear model for regression versus least squares problem

The test functions $\varphi(x)$ form the such called design matrix A

$$A = \begin{bmatrix} 1 & \varphi_1(x_1) & \varphi_2(x_1) & \dots & \varphi_M(x_1) \\ 1 & \varphi_1(x_2) & \varphi_2(x_2) & \dots & \varphi_M(x_2) \\ 1 & \varphi_1(x_3) & \varphi_2(x_3) & \dots & \varphi_M(x_3) \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 1 & \varphi_1(x_N) & \varphi_2(x_N) & \dots & \varphi_M(x_N) \end{bmatrix}. \quad (6)$$

and regression problem (or least squares problem) can be written as

$$\min_{\omega} \frac{1}{2}\|r(\omega)\|_2^2 = \min_{\omega} \frac{1}{2}\|A\omega - t\|_2^2 \quad (7)$$

where A is of the size $N \times M$ with $N > M$, t is the target vector t is of the size N, and ω is vector of weights fo the size M.

Normal Equations

Our goal is to minimize the residual $\|r(\omega)\|_2^2 = \|A\omega - t\|_2^2$. To find minimum of the error function (3) and derive the *normal equations*, we look for the ω where the gradient of $\|A\omega - t\|_2^2 = (A\omega - t)^T(A\omega - t)$ vanishes, or where $\|r'(\omega)\|_2^2 = 0$. So we want

$$
\begin{aligned}
0 &= \lim_{\|e\|\to 0} \frac{(A(\omega+e)-t)^T(A(\omega+e)-t) - (A\omega - t)^T(A\omega - t)}{\|e\|_2} \\
&= \lim_{\|e\|\to 0} \frac{((A\omega - t)+Ae)^T((A\omega - t)+Ae) - (A\omega - t)^T(A\omega - t)}{\|e\|_2} \\
&= \lim_{\|e\|\to 0} \frac{\|(A\omega - t)+Ae\|_2^2 - \|A\omega - t\|_2^2}{\|e\|_2} \\
&= \lim_{\|e\|\to 0} \frac{\|A\omega - t\|_2^2 + 2\|A\omega - t\|_2 \cdot \|Ae\|_2 + \|Ae\|_2^2 - \|A\omega - t\|_2^2}{\|e\|_2} \\
&= \lim_{\|e\|\to 0} \frac{2e^T(A^TA\omega - A^Tt) + e^TA^TAe}{\|e\|_2}
\end{aligned}
$$

Normal Equations

The second term in

$$0 = \|r'(\omega)\|_2^2 = \lim_{\|e\| \to 0} \frac{2e^T(A^TA\omega - A^Tt) + e^TA^TAe}{\|e\|_2} \tag{8}$$

$\frac{|e^TA^TAe|}{\|e\|_2} \leq \frac{\|A\|_2^2\|e\|_2^2}{\|e\|_2} = \|A\|_2^2\|e\|_2$ approaches 0 as e goes to 0, so the factor $A^TA\omega - A^Tt$ in the first term must also be zero, or $A^TA\omega = A^Tt$. This is a system of M linear equations for M unknowns, the normal equations. The gradient (8) in the discrete form will be

$$0 = \omega^T(\sum_{n=1}^{N} \varphi^T(x_n)\varphi(x_n)) - \sum_{n=1}^{N} t_n\varphi^T(x_n), \tag{9}$$

where $\varphi(x_n) = \varphi_1(x_n), ..., \varphi_M(x_n), n = 1, ..., N$ are elements of the matrix A given by (6).

Normal Equations

Thus, normal equations are

$$A^T A\omega = A^T t, \tag{10}$$

which is a symmetric linear system of the $M \times M$ equations. Normal equations in the discrete form can be written as

$$A^T A\omega = A^T t, \tag{11}$$

Using $\frac{1}{2}\|r(\omega)\|_2^2 = \frac{1}{2}\|A\omega - t\|_2^2$ we can compute the Hessian matrix $H = A^T A$. If the Hessian matrix $H = A^T A$ is positive definite, then ω is indeed a minimum. We can show that the matrix $A^T A$ is positive definite if, and only if, the columns of A are linearly independent, or when $rank(A) = M$.
If the matrix A has a full rank ($rank(A) = M$) then the system (10) is of the size M-by-M and is symmetric positive definite system of normal equations. It has the same solution ω as the least squares problem $\min_\omega \|A\omega - t\|_2^2$.

Study of bias parameter ω_0

Let us rewrite (3) making the bias parameter ω_0 explicit

$$E(\omega) = \frac{1}{2}\sum_{n=1}^{N}(t_n - \omega^T\varphi(x_n))^2 = \frac{1}{2}(\sum_{n=1}^{N}(t_n - \omega_0 - \sum_{j=1}^{M}\omega_j\varphi_j(x_n))^2. \quad (12)$$

Now we take derivative $E'(\omega_0) = 0$ to get

$$0 = E'(\omega_0) = (\sum_{n=1}^{N}(t_n - \omega_0 - \sum_{j=1}^{M}\omega_j\varphi_j(x_n))(-1), \quad (13)$$

or

$$0 = E'(\omega_0) = \sum_{n=1}^{N} t_n - N\omega_0 - (\omega_1\sum_{n=1}^{N}\varphi_1(x_n) + ... + \omega_M\sum_{n=1}^{N}\varphi_M(x_n)). \quad (14)$$

Study of bias parameter ω_0

Now solving (14) for ω_0 we get

$$\omega_0 = \bar{t} - \sum_{j=1}^{M} \omega_j \bar{\varphi}_j, \tag{15}$$

where

$$\begin{aligned} \bar{t} &= \frac{1}{N} \sum_{n=1}^{N} t_n, \\ \bar{\varphi}_j &= \frac{1}{N} \sum_{n=1}^{N} \varphi_j(x_n). \end{aligned} \tag{16}$$

Using (15) we conclude that bias ω_0 balance the difference between the averages of the target values and the weighted sum of the averages of the basis functions.

Data fitting

In this example we present the typical application of least squares or regression called data or curve fitting problem. This problem appears in statistical modelling and experimental engineering when data are generated by laboratory or other measurements.
Suppose that we have data points $(x_i, t_i), i = 1, ..., N$, and our goal is to find the vector of weights ω of the size M which will fit best to the target data t_i of the model function $f(x_i, \omega)$, in the least squares sense:

$$\min_{\omega} \sum_{i=1}^{N} (t_i - f(x_i, \omega))^2 = \min_{\omega} \|t - f(x, \omega)\|_2^2. \tag{17}$$

If the function $f(x, \omega)$ is linear then we can solve the problem (17) using least squares, or regression, method.

Recall, that the function $f(x, \omega)$ is linear if we can write it as a linear combination of the functions $\phi_j(x), j = 0, ..., M$ as:

$$f(x, \omega) = \omega_0\phi_0(x) + \omega_1\phi_1(x) + \omega_2\phi_2(x) + ... + \omega_M\phi_M(x). \quad (18)$$

Functions $\phi_j(x), j = 0, ..., M$ are called basis functions which are known.
Usually, in regression models we choose $\phi_0(x) = 1, \omega_0$ is called bias parameter.
Let now the matrix A will have entries $a_{ij} = \phi_j(x_i), i = 1, ..., N; j = 1, ..., M$.
Then a linear data fitting problem takes the form

$$A\omega \approx t \quad (19)$$

Elements of the matrix A are created by basis functions $\phi_j(x), j = 1, ..., M$. We will consider now different examples of choosing basis functions $\phi_j(x), j = 1, ..., M$.

Problem of the fitting to a polynomial

In the problem of the fitting to a polynomial

$$f(x, \omega) = \sum_{i=1}^{d} \omega_i x^{i-1} \tag{20}$$

of degree $d-1$ to data points (x_i, t_i), $i = 1, ..., N$, basis functions $\phi_j(x), j = 1, ..., M$ can be chosen as $\phi_j(x) = x^{j-1}, j = 1, ..., M$. The matrix A constructed by these basis functions in a polynomial fitting problem is a Vandermonde matrix:

$$A = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{d-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{d-1} \\ 1 & x_3 & x_3^2 & \dots & x_3^{d-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \dots & x_N^{d-1} \end{bmatrix}. \tag{21}$$

Here, $x_i, i = 1, ..., N$ are discrete points on the interval for $x = [x_{left}, x_{right}]$.

Suppose, that we choose $d = 4$ in (17). Then we can write the polynomial as $f(x, \omega) = \sum_{i=1}^{4} \omega_i x^{i-1} = \omega_1 + \omega_2 x + \omega_3 x^2 + \omega_4 x^3$ and our data fitting problem (19) for this polynomial takes the form

$$A\omega = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & x_N^3 \end{bmatrix} \cdot \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ \dots \\ t_N \end{bmatrix}. \tag{22}$$

The right hand side of the above system represents measurements or function which we want to fit. Our goal is to find such weights $\omega = \{\omega_1, \omega_2, \omega_3, \omega_4\}$ which will minimize the residual

$$r_i = f(x_i, \omega) - t_i, i = 1, ..., N.$$

Since we want minimize squared 2-norm of the residual, or

$$\|r\|_2^2 = \sum_{i=1}^{N} r_i^2 = \sum_{i=1}^{N} (f(x_i, \omega) - t_i)^2,$$

with a linear model function $f(x_i, \omega)$ then we will solve the linear least squares problem.

Regularized linear regression or least squares problem

In the regularized linear regression or least squares problem the goal is to minimize the regularized sum of squares

$$E(\omega) = \frac{1}{2}\sum_{n=1}^{N}(t_n - \omega^T\varphi(x_n))^2 + \frac{\gamma}{2}\omega^T\omega := \frac{1}{2}\|t - \omega^T\varphi(x)\|_2^2 + \frac{\gamma}{2}\|\omega\|_2^2. \quad (23)$$

to find

$$\min_{\omega} \frac{1}{2}\|t - \omega^T\varphi(x)\|_2^2 + \frac{\gamma}{2}\|\omega\|_2^2 \quad (24)$$

The problem (24) is a regularized least squares, or regression, problem of the minimizing the regularized squared residuals

$$\min_{\omega} \frac{1}{2}\|r(\omega)\|_2^2 + \frac{\gamma}{2}\|\omega\|_2^2 = \min_{\omega} \frac{1}{2}\|t - \omega^T\varphi(x)\|_2^2 + \frac{\gamma}{2}\|\omega\|_2^2 \quad (25)$$

with the residual $r(\omega) = t - \omega^T\varphi(x)$ and regularization parameter γ.
In the machine learning comminity γ is called weight decay seance in the sequential learning algorithms this parameter encourages weight values go to zero. In statistics, γ is an example of a parameter shrinkage method since it shrinks parameter values to zero.

Regularized linear regression or least squares problem

Let now the matrix A will have entries $a_{ij} = \phi_j(x_i), i = 1, ..., N; j = 1, ..., M$. Recall, that functions $\phi_j(x), j = 0, ..., M$ are called basis functions which are known. Then the regularized least squares problem takes the form

$$\min_{\omega} \frac{1}{2}\|r(\omega)\|_2^2 + \frac{\gamma}{2}\|\omega\|_2^2 = \min_{\omega} \frac{1}{2}\|A\omega - t\|_2^2 + \frac{\gamma}{2}\|\omega\|_2^2 \qquad (26)$$

Normal Equations for regularized linear regression or least squares problem

To minimize the regularized squared residuals

$$\min_{\omega} \frac{1}{2}\|r(\omega)\|_2^2 + \frac{\gamma}{2}\|\omega\|_2^2 = \min_{\omega} \frac{1}{2}\|A\omega - t\|_2^2 + \frac{\gamma}{2}\|\omega\|_2^2 \tag{27}$$

we will derive the *normal equations*. To do this we look for the ω where the gradient of $\frac{1}{2}\|A\omega - t\|_2^2 + \frac{\gamma}{2}\|\omega\|_2^2 = \frac{1}{2}(A\omega - t)^T(A\omega - t) + \frac{\gamma}{2}\omega^T\omega$ vanishes. In other words, we consider

$$\begin{aligned} 0 = &\frac{1}{2}\lim_{\|e\|\to 0} \frac{(A(\omega + e) - t)^T(A(\omega + e) - t) - (A\omega - t)^T(A\omega - t)}{\|e\|_2} \\ &+ \lim_{\|e\|\to 0} \frac{\frac{\gamma}{2}(\omega + e)^T(\omega + e) - \frac{\gamma}{2}\omega^T\omega}{\|e\|_2} = \end{aligned}$$

Normal Equations for regularized linear regression or least squares problem

$$= \frac{1}{2} \lim_{\|e\| \to 0} \frac{\|(A\omega - t) + Ae\|_2^2 - \|A\omega - t\|_2^2}{\|e\|_2} + \lim_{\|e\| \to 0} \frac{\frac{\gamma}{2}(\|\omega + e\|_2^2 - \|\omega\|_2^2)}{\|e\|_2}$$

$$= \frac{1}{2} \lim_{\|e\| \to 0} \frac{\|A\omega - t\|_2^2 + 2\|A\omega - t\|_2 \cdot \|Ae\|_2 + \|Ae\|_2^2 - \|A\omega - t\|_2^2}{\|e\|_2}$$

$$+ \frac{\gamma}{2} \lim_{\|e\| \to 0} \frac{\|\omega\|_2^2 + 2\|\omega\|_2 \cdot \|e\|_2 + \|e\|_2^2 - \|\omega\|_2^2}{\|e\|_2}$$

$$= \frac{1}{2} \lim_{\|e\| \to 0} \frac{2e^T(A^T A\omega - A^T t) + e^T A^T Ae}{\|e\|_2} + \frac{\gamma}{2} \lim_{\|e\| \to 0} \frac{2e^T\omega + e^T e}{\|e\|_2}$$

Normal Equations

The term

$$\frac{|e^T A^T A e|}{\|e\|_2} \leq \frac{\|A\|_2^2 \|e\|_2^2}{\|e\|_2} = \|A\|_2^2 \|e\|_2 \tag{28}$$

approaches 0 as e goes to 0.
Similarly, the term

$$\frac{|e^T e|}{\|e\|_2} = \frac{\|e\|_2^2}{\|e\|_2} \tag{29}$$

approaches 0 as e goes to 0.
Using these facts, we finally get

$$0 = \lim_{\|e\| \to 0} \frac{e^T (A^T A \omega - A^T t)}{\|e\|_2} + \frac{\gamma e^T \omega}{\|e\|_2}$$

so the factor $A^T A \omega - A^T t + \gamma \omega$ must also be zero, or

$$(A^T A + \gamma I)\omega = A^T t$$

This is a system of M linear equations for M unknowns, the normal equations for regularized least squares.

Artificial neural networks

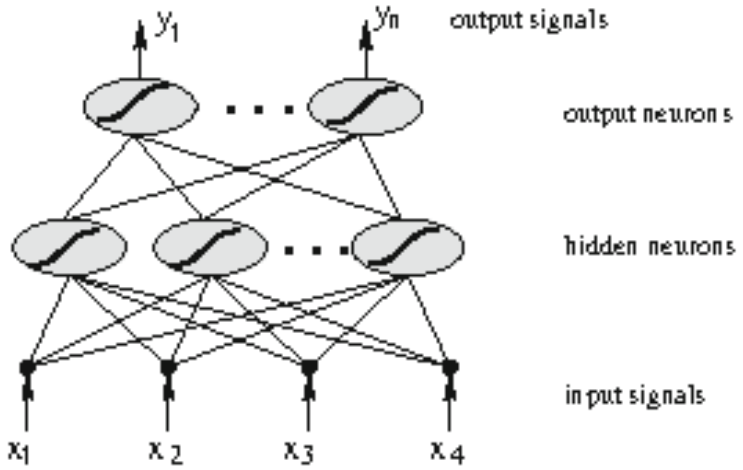


Figure: Example of neural network which contains two interconnected layers (M. Kurbat, *An Introduction to machine learning*, Springer, 2017.)

- In an artificial neural network simple units - neurons- are interconnected by weighted links into structures of high performance.
- Multilayer perceptrons and radial basis function networks will be discussed.

Neurons

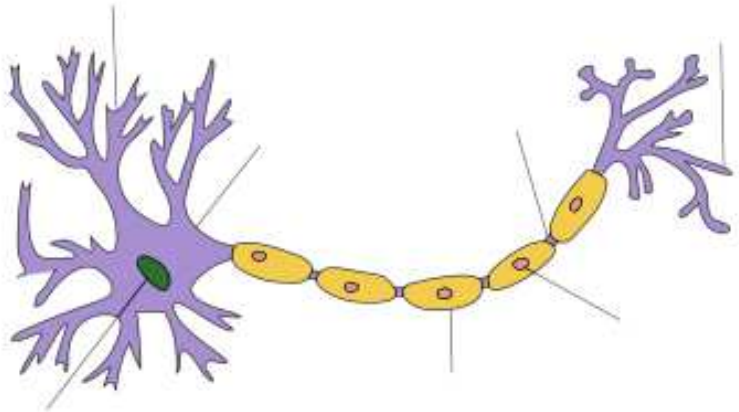


Figure: Structure of a typical neuron (Wikipedia).

- A neuron, also known as a nerve cell, is an electrically excitable cell that receives, processes, and transmits information through electrical and chemical signals. These signals between neurons occur via specialized connections called synapses.
- An artificial neuron is a mathematical function which presents a model of biological neurons, resulting in a neural network.

Artificial neurons

- Artificial neurons are elementary units in an artificial neural network. The artificial neuron receives one or more inputs and sums them to produce an output (or activation, representing a neuron's action potential which is transmitted along its axon).
- Each input is separately weighted by weights ω_{kj}, and the sum $\sum_k \omega_{kj} x_k$ is passed as an argument $\Sigma = \sum_k \omega_{kj} x_k$ through a non-linear function $f(\Sigma)$ which is called the activation function or transfer function.
- Assume that attributes x_k are normalized and belong to the interval $[-1, 1]$.

Artificial neurons

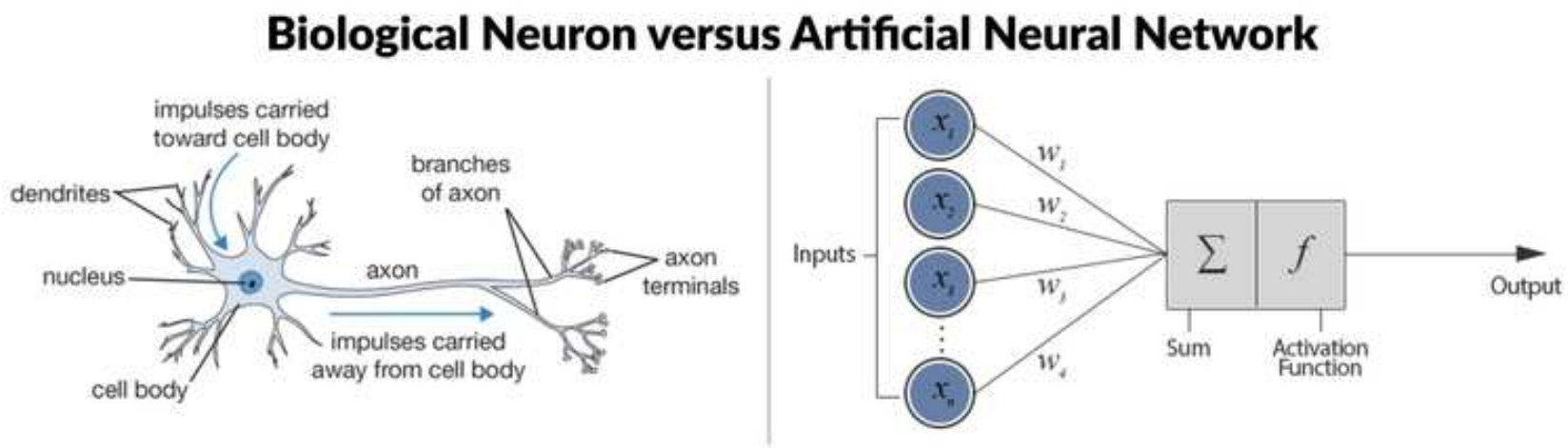


Figure: Perceptron neural network consisting of one neuron (source: DataCamp(datacamp.com)).

Each input is separately weighted by weights ω_{kj}, and the sum $\sum_k \omega_{kj} x_k$ is passed as an argument $\Sigma = \sum_k \omega_{kj} x_k$ through a non-linear function $f(\Sigma)$ which is called the activation function or transfer function.

Artificial neurons: transfer functions

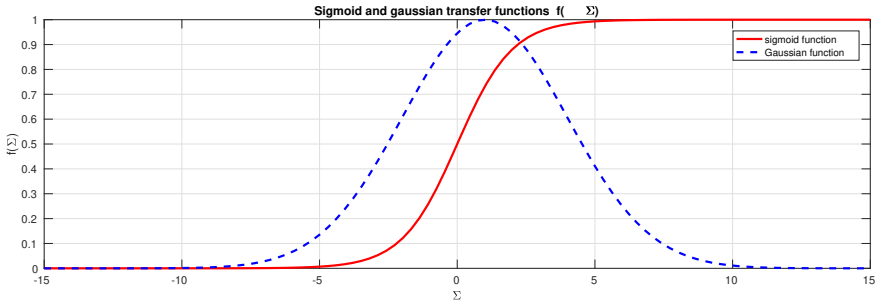


Figure: Sigmoid and Gaussian (for $b = 1, \sigma = 3$ in (32)) transfer functions.

- Different transfer (or activation) functions $f(\Sigma)$ with $\Sigma = \sum_k \omega_{kj} x_k$ are used. We will study sigmoid and gaussian functions.
- Sigmoid function:

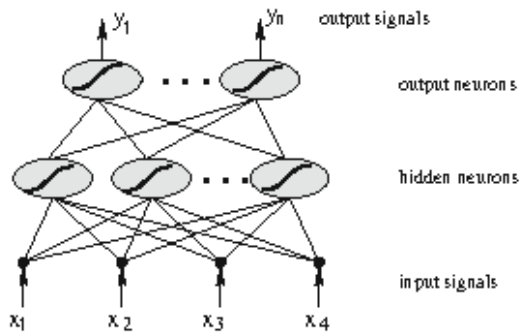
$$f(\Sigma) = \frac{1}{1 + e^{-\Sigma}} \tag{31}$$

- Gaussian function centered at b for a given variance σ^2

$$f(\Sigma) = \frac{e^{-(\Sigma - b)^2}}{2\sigma^2} \tag{32}$$

Forward propagation

Example of neural network called multilayer perceptron (one hidden layer of neurons and one output layer). (M. Kurbat, *An Introduction to machine learning*, Springer, 2017.)



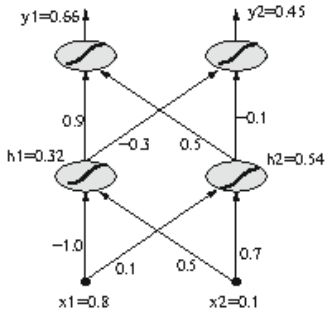
- Neurons in adjacent layer are fully interconnected.
- Forward propagation is implemented as

$$y_i = f(\Sigma_j \omega_{ji}^{(1)} x_j) = f(\Sigma_j \omega_{ji}^{(1)} \underbrace{f(\Sigma_k \omega_{kj}^{(2)} x_k)}_{x_j}), \tag{33}$$

where $\omega_{ji}^{(1)}$ and $\omega_{kj}^{(2)}$ are weights of the output and the hidden neurons, respectively, f is the transfer function.

Example of forward propagation through the network

Source: M. Kurbat, *An Introduction to machine learning*, Springer, 2017.



- Using inputs x_1, x_2 compute inputs of hidden-layer neurons:

$$x_1^{(2)} = 0.8 * (-1.0) + 0.1 * 0.5 = -0.75, \; x_2^{(2)} = 0.8 * 0.1 + 0.1 * 0.7 = 0.15$$

- Compute transfer function (sigmoid $f(\Sigma) = \frac{1}{1+e^{-\Sigma}}$ in our case):

$$h_1 = f(x_1^{(2)}) = 0.32, \; h_2 = f(x_2^{(2)}) = 0.54.$$

- Compute input of output-layer neurons

$$x_1^{(1)} = 0.32 * 0.9 + 0.54 * 0.5 = 0.56, \; x_2^{(1)} = 0.32 * (-0.3) + 0.54 * (-0.1) = -0.15.$$

- Compute outputs of output-layer neurons using transfer function (sigmoid in our case):

$$y_1 = f(x_1^{(1)}) = 0.66, \; y_2 = f(x_2^{(1)}) = 0.45.$$

Backpropagation of error through the network

Our goal is to find optimal weights $\omega_{ji}^{(1)}$ and $\omega_{kj}^{(2)}$ in forward propagation

$$y_i = f(\Sigma_j \omega_{ji}^{(1)} x_j) = f(\Sigma_j \omega_{ji}^{(1)} \underbrace{f(\Sigma_k \omega_{kj}^{(2)} x_k)}_{x_j}). \tag{34}$$

To do this we introduce functional

$$F(\omega_{ji}^{(1)}, \omega_{kj}^{(2)}) = \frac{1}{2}\|t_i - y_i\|^2 = \frac{1}{2}\sum_{i=1}^{m}(t_i - y_i)^2. \tag{35}$$

Here, $t = t(x)$ is the target vector which depends on the concrete example x. In the domain with m classes the target vector $t = (t_1(x), ..., t_m(x))$ consists of m binary numbers such that

$$t_i(x) = \begin{cases} 1, & \text{example } x \text{ belongs to } i\text{-th class,} \\ 0, & \text{otherwise.} \end{cases} \tag{36}$$

Examples of target vector and mean square error

Let there exist three different classes c_1, c_2, c_3 and x belongs to the class c_2. Then the target vector is $t = (t_1, t_2, t_3) = (0, 1, 0)$.
The mean square error is defined as

$$E = \frac{1}{m}\|t_i - y_i\|^2 = \frac{1}{m}\sum_{i=1}^{m}(t_i - y_i)^2. \tag{37}$$

Let us assume that we have two different networks to choose from, every network with 3 output neurons corresponding to classes c_1, c_2, c_3. Let $t = (t_1, t_2, t_3) = (0, 1, 0)$ and for the example x the first network output is $y_1 = (0.5, 0.2, 0.9)$ and the second network output is $y_2 = (0.6, 0.6, 0.7)$.

$$E_1 = \frac{1}{3}\sum_{i=1}^{3}(t_i - y_i)^2 = \frac{1}{3}((0-0.5)^2 + (1-0.2)^2 + (0-0.9)^2)) = 0.57,$$

$$E_2 = \frac{1}{3}\sum_{i=1}^{3}(t_i - y_i)^2 = \frac{1}{3}((0-0.6)^2 + (1-0.6)^2 + (0-0.7)^2)) = 0.34.$$

Since $E_2 < E_1$ then the second network is less wrong on the example x than the first network.

Backpropagation of error through the network

To find minimum of the functional (59) $F(\omega)$ with $\omega = (\omega_{ji}^{(1)}, \omega_{kj}^{(2)})$, recall it below:

$$F(\omega) = F(\omega_{ji}^{(1)}, \omega_{kj}^{(2)}) = \frac{1}{2}\|t_i - y_i\|^2 = \frac{1}{2}\sum_{i=1}^{m}(t_i - y_i)^2, \tag{38}$$

we need to solve the minimization problem

$$\min_{\omega} F(\omega) \tag{39}$$

and find a stationary point of (38) with respect to ω such that

$$F'(\omega)(\bar{\omega}) = 0, \tag{40}$$

where $F'(\omega)$ is the Fréchet derivative such that

$$F'(\omega)(\bar{\omega}) = F'_{\omega_{ji}^{(1)}}(\omega)(\bar{\omega}_{ji}^{(1)}) + F'_{\omega_{kj}^{(2)}}(\omega)(\bar{\omega}_{kj}^{(2)}). \tag{41}$$

Backpropagation of error through the network

Recall now that y_i in the functional (38) is defined as

$$y_i = f(\sum_j \omega_{ji}^{(1)} x_j) = f(\sum_j \omega_{ji}^{(1)} \underbrace{f(\sum_k \omega_{kj}^{(2)} x_k)}_{x_j}). \tag{42}$$

Thus, if the transfer function f in (42) is sigmoid, then

$$\begin{aligned} F'_{\omega_{ji}^{(1)}}(\omega)(\bar{\omega}_{ji}^{(1)}) &= (t_i - y_i) \cdot y'_{i\,(\omega_{ji}^{(1)})}(\bar{\omega}_{ji}^{(1)}) \\ &= (t_i - y_i) \cdot x_j \cdot f(\sum_j \omega_{ji}^{(1)} x_j)(1 - f(\sum_j \omega_{ji}^{(1)} x_j))(\bar{\omega}_{ji}^{(1)}) \\ &= (t_i - y_i) \cdot x_j \cdot y_i(1 - y_i))(\bar{\omega}_{ji}^{(1)}), \end{aligned} \tag{43}$$

Backpropagation of error through the network

Here we have used that for the sigmoid function $f'(\Sigma) = f(\Sigma)(1 - f(\Sigma))$ since

$$
\begin{aligned}
f'(\Sigma) &= \left(\frac{1}{1+e^{-\Sigma}}\right)' = \frac{1+e^{-\Sigma}-1}{(1+e^{-\Sigma})^2} \\
&= f(\Sigma)\left[\frac{(1+e^{-\Sigma})-1}{1+e^{-\Sigma}}\right] = f(\Sigma)\left[\frac{(1+e^{-\Sigma})}{1+e^{-\Sigma}} - \frac{1}{1+e^{-\Sigma}}\right] \\
&= f(\Sigma)(1-f(\Sigma)).
\end{aligned} \tag{44}
$$

Backpropagation of error through the network

Again, since

$$y_i = f(\sum_j \omega_{ji}^{(1)} x_j) = f(\sum_j \omega_{ji}^{(1)} \underbrace{f(\sum_k \omega_{kj}^{(2)} x_k)}_{x_j}). \tag{45}$$

for the sigmoid transfer function f we also get

$$\begin{aligned} F'_{\omega_{kj}^{(2)}}(\omega)(\bar{\omega}_{kj}^{(2)}) &= (t_i - y_i) \cdot y'_{i\,(\omega_{kj}^{(2)})}(\bar{\omega}_{kj}^{(2)}) \\ &= \left[\underbrace{h_j(1-h_j)}_{f'(h_j)} \cdot \left[\sum_i \underbrace{y_i(1-y_i)}_{f'(y_i)}(t_i - y_i)\omega_{ji}^{(1)}\right] \cdot x_k\right](\bar{\omega}_{kj}^{(2)}), \end{aligned} \tag{46}$$

since for the sigmoid function f we have:
$f'(h_j) = f(h_j)(1 - f(h_j)), f'(y_i) = f(y_i)(1 - f(y_i))$ (prove this). Hint:
$h_j = f(\sum_k \omega_{kj}^{(2)} x_k), y_i = f(\sum_j \omega_{ji}^{(1)} x_j)$.

Backpropagation of error through the network

Usually, $F'_{\omega_{ji}^{(1)}}(\omega)/x_j$, $F'_{\omega_{kj}^{(2)}}(\omega)/x_k$ in (43), (46) are called responsibilities of output layer neurons and hidden-layer neurons $\delta_i^{(1)}, \delta_i^{(2)}$, respectively, and they are defined as

$$\begin{aligned} \delta_i^{(1)} &= (t_i - y_i)y_i(1 - y_i), \\ \delta_j^{(2)} &= h_j(1 - h_j) \cdot \sum_i \delta_i^{(1)} \omega_{ji}^{(1)}. \end{aligned} \tag{47}$$

By knowing responsibilities (47), weights can be updates using usual gradient update formulas:

$$\begin{aligned} \omega_{ji}^{(1)} &= \omega_{ji}^{(1)} + \eta \delta_i^{(1)} x_j, \\ \omega_{kj}^{(2)} &= \omega_{kj}^{(2)} + \eta \delta_j^{(2)} x_k. \end{aligned} \tag{48}$$

Here, η is the step size in the gradient update of weights and we use value of learning rate for it such that $\eta \in (0, 1)$.

Algorithm A1: backpropagation of error through the network with one hidden layer

- Step 0. Initialize weights.
- Step 1. Take example x in the input layer and perform forward propagation.
- Step 2. Let $y = (y_1, ..., y_m)$ be the output layer and let $t = (t_1, ..., t_m)$ be the target vector.
- Step 3. For every output neuron $y_i, i = 1, ..., m$ calculate its responsibility δ_i^1 as

$$\delta_i^{(1)} = (t_i - y_i) y_i (1 - y_i). \tag{49}$$

- Step 4. For every hidden neuron compute responsibility $\delta_j^{(2)}$ for the network's error as

$$\delta_j^{(2)} = h_j(1 - h_j) \cdot \sum_i \delta_i^{(1)} (\omega_{ji})^1, \tag{50}$$

where $\delta_i^{(1)}$ are computed using (60).

- Step 5. Update weights with learning rate $\eta \in (0, 1)$ as

$$\begin{aligned} \omega_{ji}^{(1)} &= \omega_{ji}^{(1)} + \eta(\delta_i^{(1)}) x_j, \\ \omega_{kj}^{(2)} &= \omega_{kj}^{(2)} + \eta(\delta_j^{(2)}) x_k. \end{aligned} \tag{51}$$

Algorithm A2: backpropagation of error through the network with l hidden layers

- Step 0. Initialize weights and take $l = 1$.
- Step 1. Take example x^l in the input layer and perform forward propagation.
- Step 2. Let $y^l = (y_1^l, ..., y_m^l)$ be the output layer and let $t^l = (t_1^l, ..., t_m^l)$ be the target vector.
- Step 3. For every output neuron $y_i^l, i = 1, ..., m$ calculate its responsibility $(\delta_i^{(1)})^l$ as

$$(\delta_i^{(1)})^l = (t_i^l - y_i^l) y_i^l (1 - y_i^l). \tag{52}$$

- Step 4. For every hidden neuron compute responsibility $(\delta_j^{(2)})^l$ for the network's error as

$$(\delta_j^{(2)})^l = h_j^l (1 - h_j^l) \cdot \sum_i (\delta_i^{(1)})^l (\omega_{ji}^{(1)})^l, \tag{53}$$

where $(\delta_i^{(1)})^l$ are computed using (60).

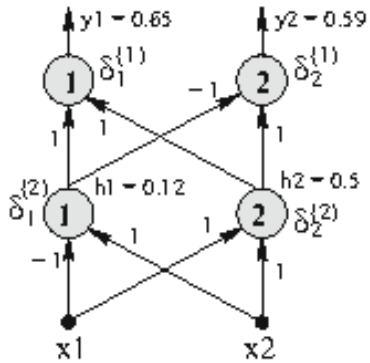
- Step 5. Update weights with learning rate $\eta^l \in (0, 1)$ as

$$\begin{aligned} (\omega_{ji}^{(1)})^{l+1} &= (\omega_{ji}^{(1)})^l + \eta^l (\delta_i^{(1)})^l x_j^l, \\ (\omega_{kj}^{(2)})^{l+1} &= (\omega_{kj}^{(2)})^l + \eta^l (\delta_j^{(2)})^l x_k^l. \end{aligned} \tag{54}$$

- Step 6. If the mean square error less than tolerance, or $\|(\omega_{ji}^{(1)})^{l+1} - (\omega_{ji}^{(1)})^l\| < \epsilon_1$ and $\|(\omega_{kj}^{(2)})^{l+1} - (\omega_{kj}^{(2)})^l\| < \epsilon_2$ stop, otherwise go to the next layer $l = l + 1$, assign $x^l = x^{l+1}$ and return to the step 1. Here, ϵ_1, ϵ_2 are tolerances chosen by the user.

Example of backpropagation of error through the network

Source: M. Kurbat, *An Introduction to machine learning*, Springer, 2017.



- Assume that after forward propagation with sigmoid transfer function we have

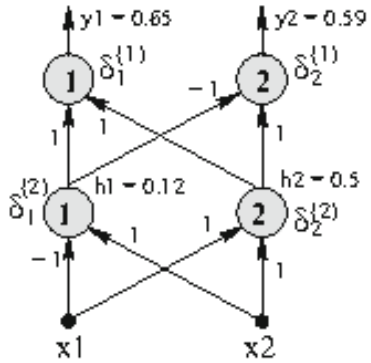
$$h_1 = f(x_1^{(2)}) = 0.12,\ h_2 = f(x_2^{(2)}) = 0.5,$$
$$y_1 = f(x_1^{(1)}) = 0.65,\ y_2 = f(x_2^{(1)}) = 0.59.$$

- Let the target vector be $t(x) = (1, 0)$ for the output vector $y = (0.65, 0.59)$.
- Compute responsibility for the output neurons:

$$\sigma_1^{(1)} = y_1 * (1 - y_1)(t_1 - y_1) = 0.65(1 - 0.65)(1 - 0.65) = 0.0796,$$
$$\sigma_2^{(1)} = y_2 * (1 - y_2)(t_2 - y_2) = 0.59(1 - 0.59)(0 - 0.59) = -0.1427$$

Example of backpropagation of error through the network

Source: M. Kurbat, *An Introduction to machine learning*, Springer, 2017.



- Compute the weighted sum for every hidden neuron

$$\delta_1 = \sigma_1^{(1)} w_{11}^{(1)} + \sigma_2^{(1)} w_{12}^{(1)} = 0.0796 * 1 + (-0.1427) * (-1) = 0.2223,$$
$$\delta_2 = \sigma_1^{(1)} w_{21}^{(1)} + \sigma_2^{(1)} w_{22}^{(1)} = 0.0796 * 1 + (-0.1427) * 1 = -0.0631.$$

- Compute responsibility for the hidden neurons for above computed δ_1, δ_2:

$$\sigma_1^{(2)} = h_1(1-h_1)\delta_1 = -0.0235,\ \sigma_2^{(2)} = h_2(1-h_2)\delta_2 = 0.0158.$$

Example of backpropagation of error through the network

- Compute new weights $\omega_{ji}^{(1)}$ for output layer with learning rate $\eta = 0.1$ as:

$$\omega_{11}^{(1)} = \omega_{11}^{(1)} + \eta\sigma_1^{(1)} h_1 = 1 + 0.1 * 0.0796 * 0.12 = 1.00096,$$
$$\omega_{21}^{(1)} = \omega_{21}^{(1)} + \eta\sigma_1^{(1)} h_2 = 1 + 0.1 * 0.0796 * 0.5 = 1.00398,$$
$$\omega_{12}^{(1)} = \omega_{12}^{(1)} + \eta\sigma_2^{(1)} h_1 = -1 + 0.1 * (-0.1427) * 0.12 = -1.0017,$$
$$\omega_{22}^{(1)} = \omega_{22}^{(1)} + \eta\sigma_2^{(1)} h_2 = 1 + 0.1 * (-0.1427) * 0.5 = 0.9929.$$

- Compute new weights $\omega_{kj}^{(2)}$ for hidden layer with learning rate $\eta = 0.1$ as:

$$\omega_{11}^{(2)} = \omega_{11}^{(2)} + \eta\sigma_1^{(2)} x_1 = -1 + 0.1 * (-0.0235) * 1 = -1.0024,$$
$$\omega_{21}^{(2)} = \omega_{21}^{(2)} + \eta\sigma_1^{(2)} x_2 = 1 + 0.1 * (-0.0235) * 1 = 1.0024,$$
$$\omega_{12}^{(2)} = \omega_{12}^{(2)} + \eta\sigma_2^{(2)} x_1 = 1 + 0.1 * 0.0158 * 1 = 1.0016,$$
$$\omega_{22}^{(2)} = \omega_{22}^{(2)} + \eta\sigma_2^{(2)} x_2 = 1 + 0.1 * 0.0158 * (-1) = 0.9984.$$

- Using computed weights for hidden and output layers, one can test a neural network for a new example.

Perceptron non-regularized neural network

- Step 0. Initialize weights ω_i to small random numbers.
- Step 1. If $\sum_{i=0}^{n} \omega_i x_i > 0$ we will say that the example is positive and $h(x) = 1$.
- Step 2. If $\sum_{i=0}^{n} \omega_i x_i < 0$ we will say the the example is negative and $h(x) = 0$.
- Step 3. Update every weight ω_i using algorithm of backpropagation of error through the network (perform steps 3-5 of A1 or A2)
- Step 4. If $c(\mathbf{x}) = h(\mathbf{x})$ for all learning examples - stop. Otherwise return to step 1.

Here, $\eta \in (0, 1]$ is called the learning rate.

Non-regularized and regularized neural network

Our goal is to find optimal weights $\omega_{ji}^{(1)}$ and $\omega_{kj}^{(2)}$ in forward propagation

$$y_i = f(\Sigma_j \omega_{ji}^{(1)} x_j) = f(\Sigma_j \omega_{ji}^{(1)} \underbrace{f(\Sigma_k \omega_{kj}^{(2)} x_k)}_{x_j}). \tag{55}$$

To do this we introduce functional

$$F(\omega_{ji}^{(1)}, \omega_{kj}^{(2)}) = \frac{1}{2}\|t_i - y_i\|^2 = \frac{1}{2}\sum_{i=1}^{m}(t_i - y_i)^2. \tag{56}$$

Here, $t = t(x)$ is the target vector which depends on the concrete example x. In the domain with m classes the target vector $t = (t_1(x), ..., t_m(x))$ consists of m binary numbers such that

$$t_i(x) = \begin{cases} 1, & \text{example } x \text{ belongs to } i\text{-th class,} \\ 0, & \text{otherwise.} \end{cases} \tag{57}$$

- Non-regularized neural network

$$F(w) = \frac{1}{2}\|t_i - y_i(w)\|^2 = \frac{1}{2}\sum_{i=1}^{m}(t_i - y_i(w))^2. \quad (58)$$

- Regularized neural network

$$F(w) = \frac{1}{2}\|t_i - y_i(w)\|^2 + \frac{1}{2}\gamma\|w\|^2 = \frac{1}{2}\sum_{i=1}^{m}(t_i - y_i(w))^2 + \frac{1}{2}\gamma\sum_{j=1}^{M}|w_j|^2 \quad (59)$$

Here, γ is reg.parameter, $\|w\|^2 = w^T w = w_1^2 + ... + w_M^2$, M is number of weights.

Algorithm: backpropagation of error through the regularized network with one hidden layer

- Step 0. Initialize weights.
- Step 1. Take example x in the input layer and perform forward propagation.
- Step 2. Let $y = (y_1, ..., y_m)$ be the output layer and let $t = (t_1, ..., t_m)$ be the target vector.
- Step 3. For every output neuron $y_i, i = 1, ..., m$ calculate its responsibility δ_i^1 as

$$\delta_i^{(1)} = (t_i - y_i) y_i (1 - y_i). \tag{60}$$

- Step 4. For every hidden neuron compute responsibility $\delta_j^{(2)}$ for the network's error as

$$\delta_j^{(2)} = h_j(1 - h_j) \cdot \sum_i \delta_i^{(1)} (\omega_{ji})^1, \tag{61}$$

where $\delta_i^{(1)}$ are computed using (60).

- Step 5. Update weights with learning rate $\eta \in (0, 1)$ and regularization parameters $\gamma_1, \gamma_2 \in (0, 1)$ as

$$\begin{aligned} \omega_{ji}^{(1)} &= \omega_{ji}^{(1)} + \eta(\delta_i^{(1)}) x_j + \gamma_1 \omega_{ji}^{(1)}, \\ \omega_{kj}^{(2)} &= \omega_{kj}^{(2)} + \eta(\delta_j^{(2)}) x_k + \gamma_2 \omega_{kj}^{(1)}. \end{aligned} \tag{62}$$