Mathematical Background: Regression

Mathematical Background: Linear Regression

Training neural networks is closely related to regression.

Given:
- A dataset $((x_1, y_1), \ldots, (x_n, y_n))$ of n data tuples and
- a hypothesis about the functional relationship, e.g. $y = g(x) = a + bx$.

Approach: Minimize the sum of squared errors, that is,

$$F(a, b) = \sum_{i=1}^{n} (g(x_i) - y_i)^2 = \sum_{i=1}^{n} (a + bx_i - y_i)^2.$$

Necessary conditions for a minimum
(a.k.a. Fermat's theorem, after Pierre de Fermat, 1601–1665):

$$\frac{\partial F}{\partial a} = \sum_{i=1}^{n} 2(a + bx_i - y_i) = 0 \quad \text{and}$$

$$\frac{\partial F}{\partial b} = \sum_{i=1}^{n} 2(a + bx_i - y_i)x_i = 0$$

Mathematical Background: Linear Regression

Result of necessary conditions: System of so-called **normal equations**, that is,

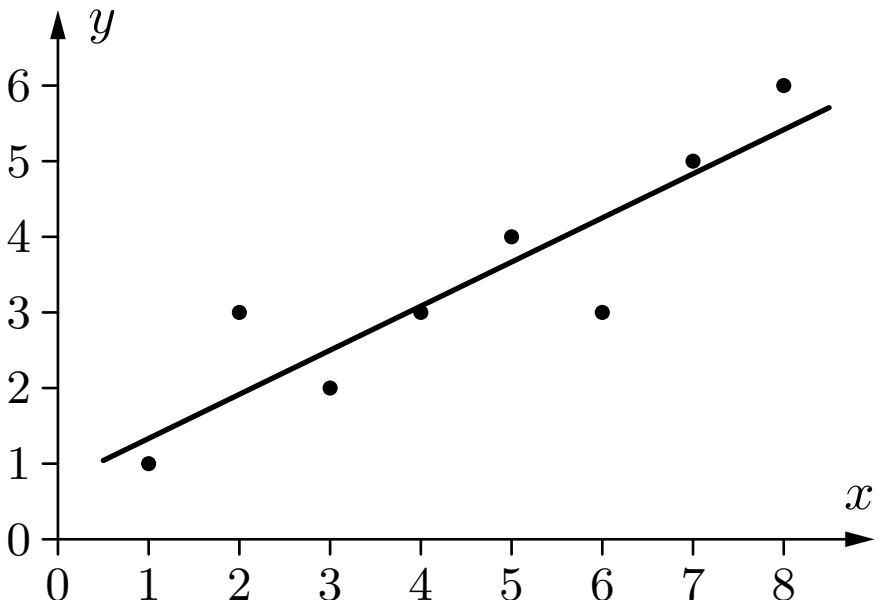
$$\begin{aligned} na + \left(\sum_{i=1}^{n} x_i\right) b &= \sum_{i=1}^{n} y_i, \\ \left(\sum_{i=1}^{n} x_i\right) a + \left(\sum_{i=1}^{n} x_i^2\right) b &= \sum_{i=1}^{n} x_i y_i. \end{aligned}$$

- Two linear equations for two unknowns a and b.
- System can be solved with standard methods from linear algebra.
- Solution is unique unless all x-values are identical.
- The resulting line is called a **regression line**.

Linear Regression: Example

x	1	2	3	4	5	6	7	8
y	1	3	2	3	4	3	5	6

$$y = \frac{3}{4} + \frac{7}{12}x.$$



Mathematical Background: Polynomial Regression

Generalization to polynomials

$$y = p(x) = a_0 + a_1 x + \ldots + a_m x^m$$

Approach: Minimize the sum of squared errors, that is,

$$F(a_0, a_1, \ldots, a_m) \;=\; \sum_{i=1}^{n} (p(x_i) - y_i)^2 \;=\; \sum_{i=1}^{n} (a_0 + a_1 x_i + \ldots + a_m x_i^m - y_i)^2$$

Necessary conditions for a minimum: All partial derivatives vanish, that is,

$$\frac{\partial F}{\partial a_0} = 0, \quad \frac{\partial F}{\partial a_1} = 0, \quad \ldots \quad , \frac{\partial F}{\partial a_m} = 0.$$

Mathematical Background: Polynomial Regression

System of normal equations for polynomials

$$
\begin{array}{rcccccccc}
n a_0 & + & \left(\sum_{i=1}^n x_i\right) a_1 & + & \ldots & + & \left(\sum_{i=1}^n x_i^m\right) a_m & = & \sum_{i=1}^n y_i \\
\left(\sum_{i=1}^n x_i\right) a_0 & + & \left(\sum_{i=1}^n x_i^2\right) a_1 & + & \ldots & + & \left(\sum_{i=1}^n x_i^{m+1}\right) a_m & = & \sum_{i=1}^n x_i y_i \\
\vdots & & & & & & & & \vdots \\
\left(\sum_{i=1}^n x_i^m\right) a_0 & + & \left(\sum_{i=1}^n x_i^{m+1}\right) a_1 & + & \ldots & + & \left(\sum_{i=1}^n x_i^{2m}\right) a_m & = & \sum_{i=1}^n x_i^m y_i,
\end{array}
$$

- $m + 1$ linear equations for $m + 1$ unknowns $a_0, \ldots, a_m$.
- System can be solved with standard methods from linear algebra.
- Solution is unique unless the points lie exactly on a polynomial of lower degree.

Generalization to more than one argument

$$z = f(x, y) = a + bx + cy$$

Approach: Minimize the sum of squared errors, that is,

$$F(a, b, c) = \sum_{i=1}^{n} (f(x_i, y_i) - z_i)^2 = \sum_{i=1}^{n} (a + bx_i + cy_i - z_i)^2$$

Necessary conditions for a minimum: All partial derivatives vanish, that is,

$$\begin{aligned}
\frac{\partial F}{\partial a} &= \sum_{i=1}^{n} 2(a + bx_i + cy_i - z_i) &&= 0, \\
\frac{\partial F}{\partial b} &= \sum_{i=1}^{n} 2(a + bx_i + cy_i - z_i)x_i &&= 0, \\
\frac{\partial F}{\partial c} &= \sum_{i=1}^{n} 2(a + bx_i + cy_i - z_i)y_i &&= 0.
\end{aligned}$$

Mathematical Background: Multilinear Regression

System of normal equations for several arguments

$$
\begin{aligned}
na + \left(\sum_{i=1}^{n} x_i\right) b + \left(\sum_{i=1}^{n} y_i\right) c &= \sum_{i=1}^{n} z_i \\
\left(\sum_{i=1}^{n} x_i\right) a + \left(\sum_{i=1}^{n} x_i^2\right) b + \left(\sum_{i=1}^{n} x_i y_i\right) c &= \sum_{i=1}^{n} z_i x_i \\
\left(\sum_{i=1}^{n} y_i\right) a + \left(\sum_{i=1}^{n} x_i y_i\right) b + \left(\sum_{i=1}^{n} y_i^2\right) c &= \sum_{i=1}^{n} z_i y_i
\end{aligned}
$$

- 3 linear equations for 3 unknowns a, b, and c.
- System can be solved with standard methods from linear algebra.
- Solution is unique unless all data points lie on a straight line.

Multilinear Regression

General multilinear case:

$$y = f(x_1, \ldots, x_m) = a_0 + \sum_{k=1}^{m} a_k x_k$$

Approach: Minimize the sum of squared errors, that is,

$$F(\vec{a}) = (\mathbf{X}\vec{a} - \vec{y})^\top (\mathbf{X}\vec{a} - \vec{y}),$$

where

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & \ldots & x_{m1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1n} & \ldots & x_{mn} \end{pmatrix}, \qquad \vec{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \qquad \text{and} \qquad \vec{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix}$$

Necessary conditions for a minimum:

$$\vec{\nabla}_{\vec{a}}\, F(\vec{a}) = \vec{\nabla}_{\vec{a}}\, (\mathbf{X}\vec{a} - \vec{y})^\top (\mathbf{X}\vec{a} - \vec{y}) = \vec{0}$$

Multilinear Regression

- $\vec{\nabla}_{\vec{a}} F(\vec{a})$ may easily be computed by remembering that the differential operator

$$\vec{\nabla}_{\vec{a}} = \left(\frac{\partial}{\partial a_0}, \ldots, \frac{\partial}{\partial a_m} \right)$$

  behaves formally like a vector that is "multiplied" to the sum of squared errors.

- Alternatively, one may write out the differentiation componentwise.

With the former method we obtain for the derivative:

$$\begin{aligned}
\vec{\nabla}_{\vec{a}} F(\vec{a}) &= \vec{\nabla}_{\vec{a}} \left((\mathbf{X}\vec{a} - \vec{y})^\top (\mathbf{X}\vec{a} - \vec{y}) \right) \\
&= \left(\vec{\nabla}_{\vec{a}} \left(\mathbf{X}\vec{a} - \vec{y} \right) \right)^\top (\mathbf{X}\vec{a} - \vec{y}) + ((\mathbf{X}\vec{a} - \vec{y})^\top \left(\vec{\nabla}_{\vec{a}} \left(\mathbf{X}\vec{a} - \vec{y} \right) \right))^\top \\
&= \left(\vec{\nabla}_{\vec{a}} \left(\mathbf{X}\vec{a} - \vec{y} \right) \right)^\top (\mathbf{X}\vec{a} - \vec{y}) + \left(\vec{\nabla}_{\vec{a}} \left(\mathbf{X}\vec{a} - \vec{y} \right) \right)^\top (\mathbf{X}\vec{a} - \vec{y}) \\
&= 2\mathbf{X}^\top (\mathbf{X}\vec{a} - \vec{y}) \\
&= 2\mathbf{X}^\top \mathbf{X}\vec{a} - 2\mathbf{X}^\top \vec{y} \quad = \quad \vec{0}
\end{aligned}$$

Multilinear Regression

Necessary condition for a minimum therefore:

$$\begin{aligned}\vec{\nabla}_{\vec{a}} F(\vec{a}) &= \vec{\nabla}_{\vec{a}} (\mathbf{X}\vec{a} - \vec{y})^\top (\mathbf{X}\vec{a} - \vec{y}) \\ &= 2\mathbf{X}^\top \mathbf{X}\vec{a} - 2\mathbf{X}^\top \vec{y} \stackrel{!}{=} \vec{0}\end{aligned}$$

As a consequence we obtain the system of **normal equations**:

$$\mathbf{X}^\top \mathbf{X}\vec{a} = \mathbf{X}^\top \vec{y}$$

This system has a solution unless $\mathbf{X}^\top \mathbf{X}$ is singular. If it is regular, we have

$$\vec{a} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \vec{y}.$$

$(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$ is called the (Moore-Penrose-)**Pseudoinverse** of the matrix $\mathbf{X}$.

With the matrix-vector representation of the regression problem
an extension to **multipolynomial regression** is straighforward:
Simply add the desired products of powers to the matrix $\mathbf{X}$.

Mathematical Background: Logistic Regression

Generalization to non-polynomial functions

$$\text{Simple example:} \qquad y = ax^b$$

Idea: Find transformation to linear/polynomial case.

$$\text{Transformation for the above example:} \qquad \ln y = \ln a + b \cdot \ln x.$$

Special case: **logistic function**

$$y = \frac{Y}{1 + e^{a+bx}} \qquad \Leftrightarrow \qquad \frac{1}{y} = \frac{1 + e^{a+bx}}{Y} \qquad \Leftrightarrow \qquad \frac{Y - y}{y} = e^{a+bx}.$$

Result: Apply so-called **Logit-Transformation**

$$\ln\left(\frac{Y - y}{y}\right) = a + bx.$$

Logistic Regression: Example

x	1	2	3	4	5
y	0.4	1.0	3.0	5.0	5.6

Transform the data with

$$z = \ln\left(\frac{Y - y}{y}\right), \qquad Y = 6.$$

The transformed data points are

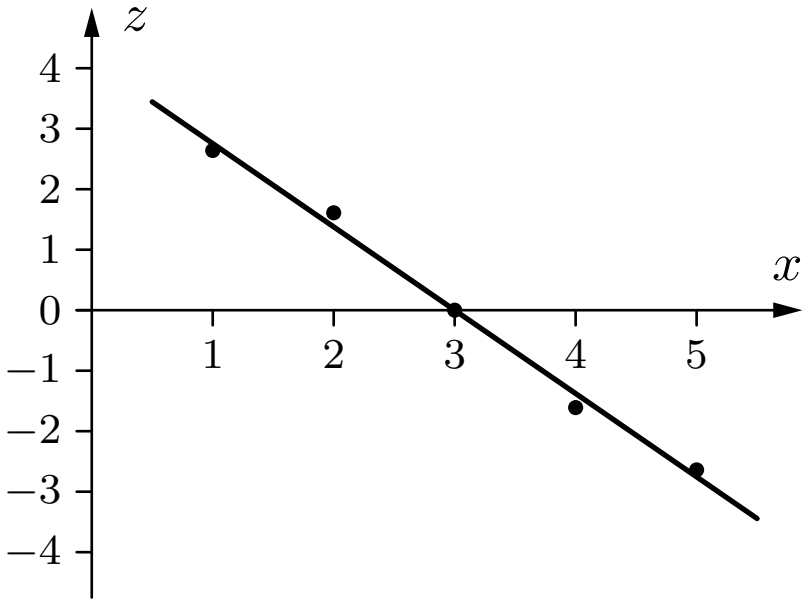
x	1	2	3	4	5
z	2.64	1.61	0.00	-1.61	-2.64

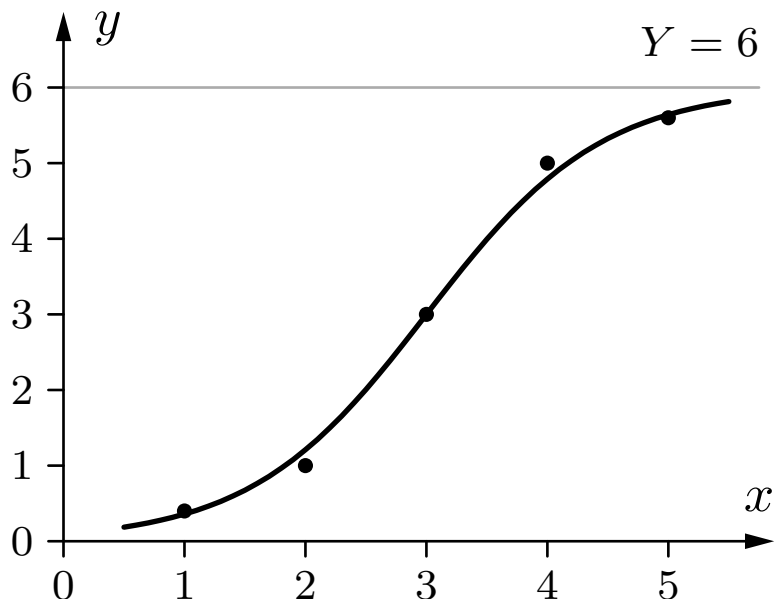
The resulting regression line and therefore the desired function are

$$z \approx -1.3775x + 4.133 \qquad \text{and} \qquad y \approx \frac{6}{1 + e^{-1.3775x+4.133}}.$$

Attention: Note that the error is minimized only in the transformed space! Therefore the function in the original space may not be optimal!

Logistic Regression: Example





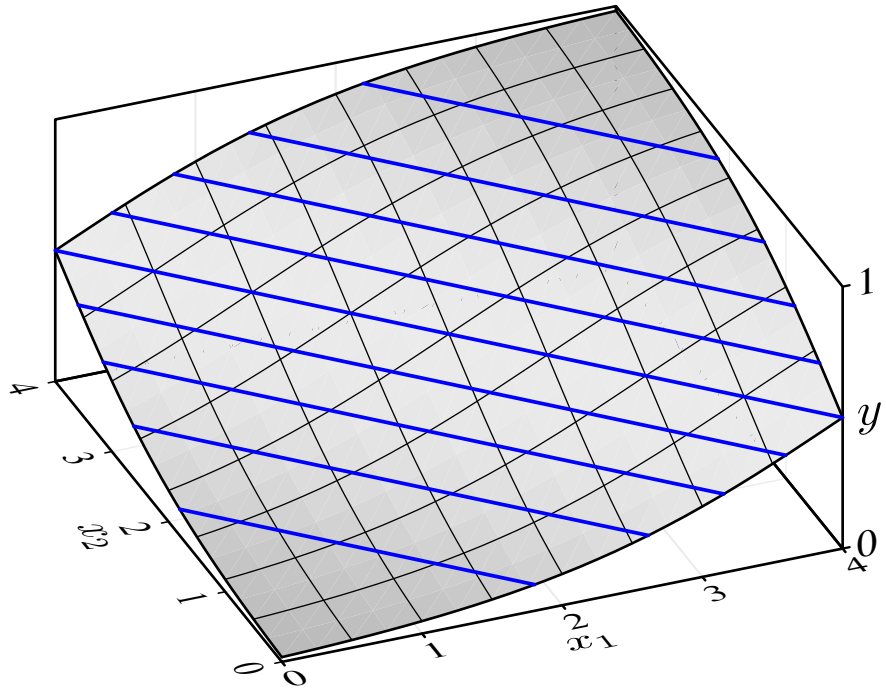
The logistic regression function can be computed by a single neuron with

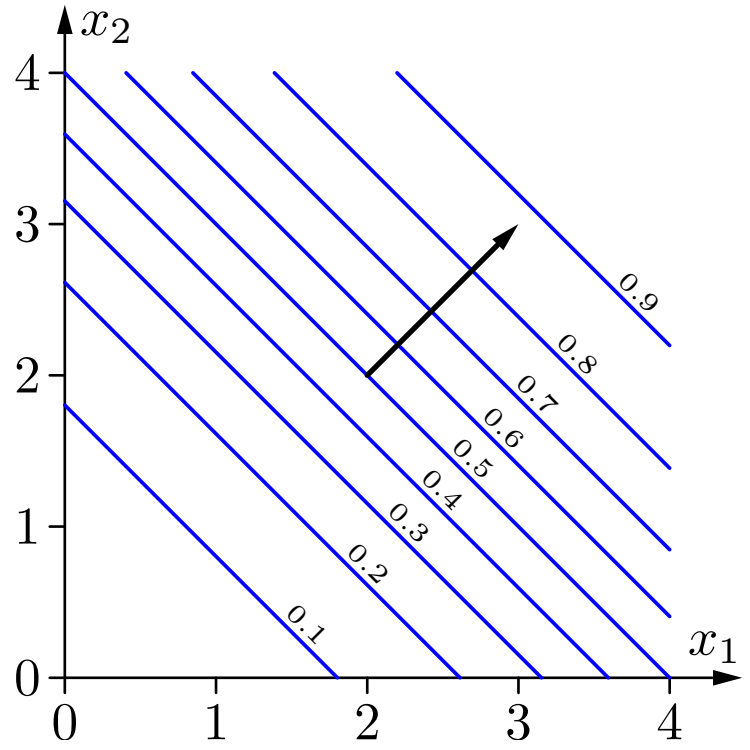
- network input function $f_{\text{net}}(x) \equiv wx$ with $w \approx -1.3775$,
- activation function $f_{\text{act}}(\text{net}, \theta) \equiv \left(1 + e^{-(\text{net} - \theta)}\right)^{-1}$ with $\theta \approx 4.133$ and
- output function $f_{\text{out}}(\text{act}) \equiv 6\,\text{act}$.

Logistic Function: Two-dimensional Example

Example logistic function for two arguments x_1 and x_2:

$$y = \frac{1}{1 + \exp(4 - x_1 - x_2)} = \frac{1}{1 + \exp\Big(4 - (1,1)^\top (x_1, x_2)\Big)}$$





The blue lines have show where the logistic function has a certain value in $\{0.1, \ldots, 0.9\}$.

Logistic Regression: Two Class Problems

- Let C be a class attrib., $\mathrm{dom}(C) = \{c_1, c_2\}$, and $\vec{X}$ an m-dim. random vector.
  Let $P(C = c_1 \mid \vec{X} = \vec{x}) = p(\vec{x})$ and $P(C = c_2 \mid \vec{X} = \vec{x}) = 1 - p(\vec{x})$.
- **Given:** A set of data points $\mathbf{X} = \{\vec{x}_1, \ldots, \vec{x}_n\}$ (realizations of $\vec{X}$), each of which belongs to one of the two classes c_1 and c_2.
- **Desired:** A simple description of the function $p(\vec{x})$.
- **Approach:** Describe p by a logistic function:

$$p(\vec{x}) \quad = \quad \frac{1}{1 + e^{a_0 + \vec{a}\vec{x}}} \quad = \quad \frac{1}{1 + \exp\left(a_0 + \sum_{i=1}^{m} a_i x_i\right)}$$

  Apply logit transformation to $p(x)$:

$$\ln\left(\frac{1 - p(\vec{x})}{p(\vec{x})}\right) \quad = \quad a_0 + \vec{a}\vec{x} \quad = \quad a_0 + \sum_{i=1}^{m} a_i x_i$$

  The values $p(\vec{x}_i)$ may be obtained by kernel estimation (see next slide).

Logistic Regression: Kernel Estimation

- **Idea:** Define an "influence function" (kernel), which describes how strongly a data point influences the probability estimate for neighboring points.
- Common choice for the kernel function: **Gaussian function**

$$K(\vec{x}, \vec{y}) \quad = \quad \frac{1}{(2\pi\sigma^2)^{\frac{m}{2}}} \exp\left(-\frac{(\vec{x}-\vec{y})^\top(\vec{x}-\vec{y})}{2\sigma^2}\right)$$

- Kernel estimate of probability density given a data set $\mathbf{X} = \{\vec{x}_1, \ldots, \vec{x}_n\}$:

$$\hat{f}(\vec{x}) \quad = \quad \frac{1}{n}\sum_{i=1}^{n} K(\vec{x}, \vec{x}_i).$$

- Kernel estimation applied to a two class problem:

$$\hat{p}(\vec{x}) \quad = \quad \frac{\sum_{i=1}^{n} c(\vec{x}_i) K(\vec{x}, \vec{x}_i)}{\sum_{i=1}^{n} K(\vec{x}, \vec{x}_i)}.$$

  It is $c(\vec{x}_i) = 1$ if x_i belongs to class c_1 and $c(\vec{x}_i) = 0$ otherwise.