Artificial Intelligence

Neural Networks

Lesson 6: Regression

Vincenzo Piuri

Università degli Studi di Milano

Contents

- Linear regression
- Polynomial regression
- Multilinear regression
- Logistic regression
- Two-class problems

Linear Regression (1)

Training neural networks is closely related to regression

- Given
  - a dataset $((x_1, y_1), \ldots, (x_n, y_n))$, of n data tuples
  - a hypothesis about the functional relationship e.g. $y = g(x) = a + bx$
- Approach
  - Minimize the sum of squared errors

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

Linear Regression (2)

- Necessary conditions for minimum (Fermat's theorem)

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

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

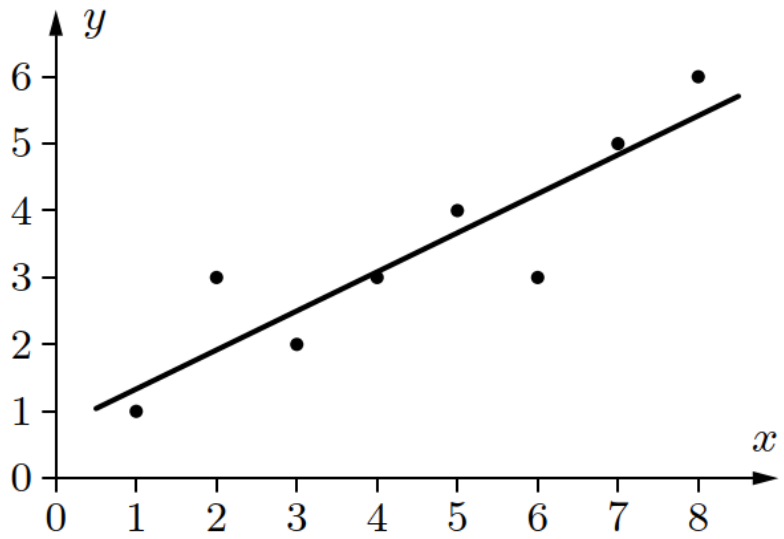
  - System can be solved with standard methods from linear algebra.
  - Solution is unique unless all x-values are identical.

Linear Regression (3)

- The resulting line is called a regression line

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

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



Polynomial Regression (1)

- Generalization to polynomials:

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

- Approach
  - Minimize the sum of squared errors

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

Polynomial Regression (2)

- Necessary conditions for a minimum:
  all partial derivates vanish

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

  – System can be solved with standard methods from linear algebra.

  – Solution is unique unless the points lie exactly on a polynomial of lower degree.

Multilinear Regression (1)

- Generalization to more than one argument

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

- Approach
  - Minimize the sum of squared errors

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

Multilinear Regression (2)

- Necessary conditions for a minimum: All partial derivatives vanish

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

  - System can be solved with standard methods from linear algebra.
  - Solution is unique unless all data points lie on a straight line.

Multilinear Regression (3)

- General multilinear case:

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

- Approach
  - Minimize the sum of squared errors

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

  - where

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

Multilinear Regression (4)

- Necessary conditions for a minimum

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

  - System of normal equations

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

  - Has a solution unless $X^T X$ is singular

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

Multilinear Regression (5)

An extension to **multi-polynomial regression** is straightforward: add the desired products of powers to the matrix X.

Logistic Regression (1)

- Generalization to non-polynomial functions

$$y = ax^b$$

- Approach
  - Find transformation to linear/polynomial case
    $\ln y = \ln a + b \ln x$
- Logistic function

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

  - Apply Logit-Transformation

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

Logistic Regression (2)

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

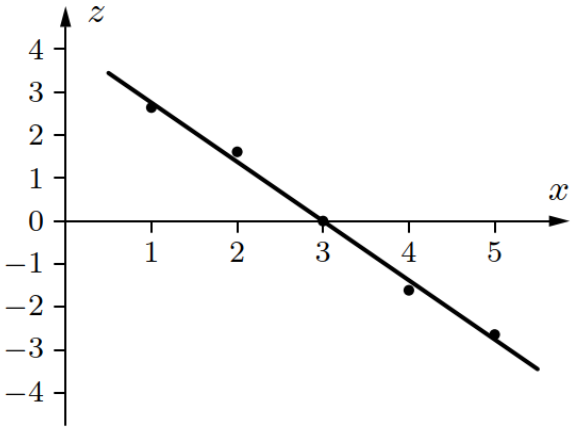
$$z = \ln\left(\frac{Y-y}{y}\right), \qquad Y = 6.$$

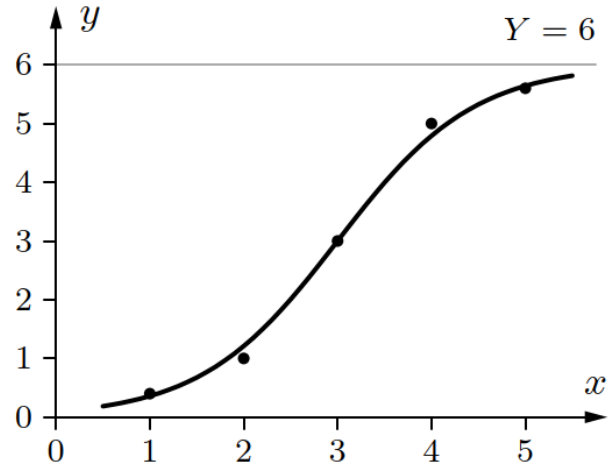
x	1	2	3	4	5
z	2.64	1.61	0.00	-1.61	-2.64

- Resulting regression line and therefore the desired function are

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

Logistic Regression (3)



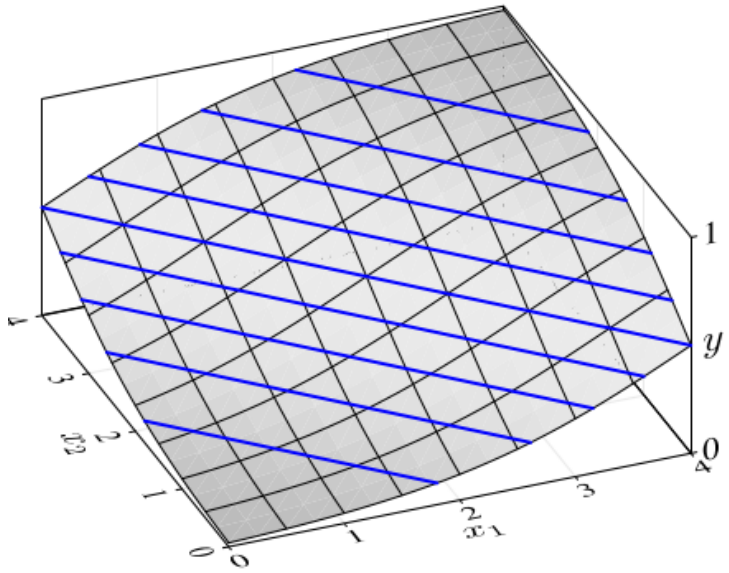


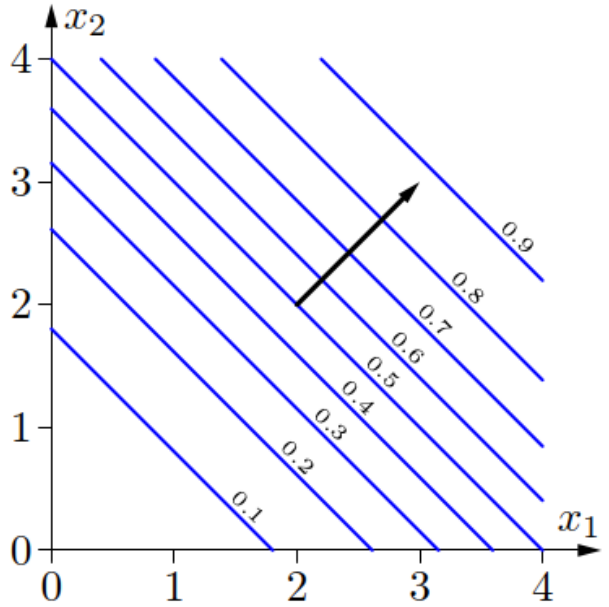
- The logistic regression function can be computed by a single neuron

Logistic Regression (4)

Logistic function for two arguments x_1 and x_2:

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





Two-class Problems (1)

Two-class problems

- Let C be a class of attributes, $\mathrm{dom}(C) = \{c_1, c_2\}$, and $\vec{X}$ an m-dimension 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 $X = \{\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}\}$ each of which belongs to one of two classes c_1 and c_2
- Desired
  - A simple description of the function $p(\vec{x})$

Two-class Problems (2)

- Approach
  - Describe p by a logistic function

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

  - Apply logit transformation to $p(x)$

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

  - The values $p(\vec{x})$ may be obtained by kernel estimation

Two-class Problems (3)

"Influence function" (**kernel**) describes how strongly a data point influences the probability estimate for neighboring points.

- Common kernel: *Gaussian function*

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

Two-class Problems (4)

- Kernel estimate of probability density given a data set

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

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