Machine Learning and Pervasive Computing

Stephan Sigg

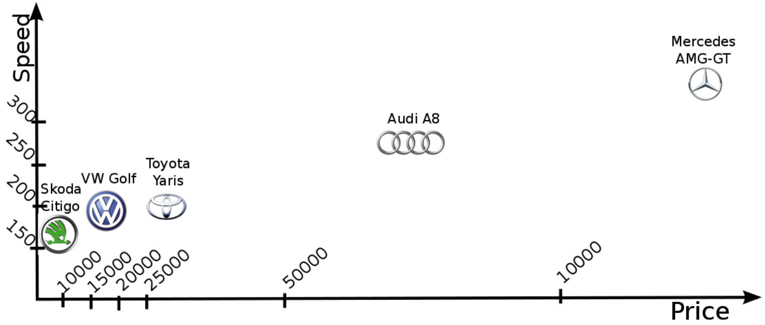
Georg-August-University Goettingen, Computer Networks

05.11.2014

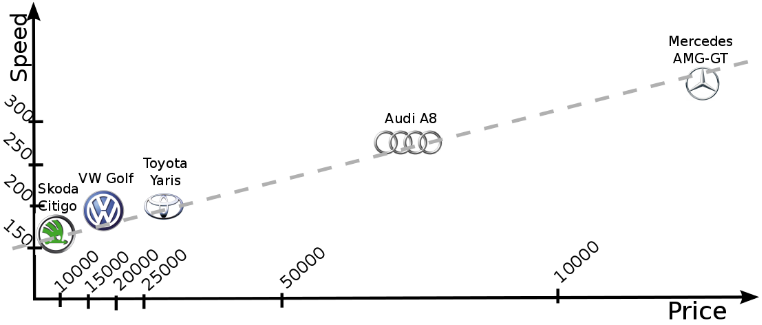
Overview and Structure

Outline

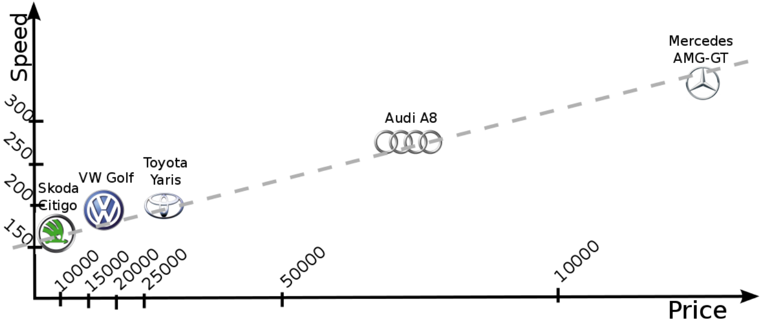
Linear regression



Linear regression

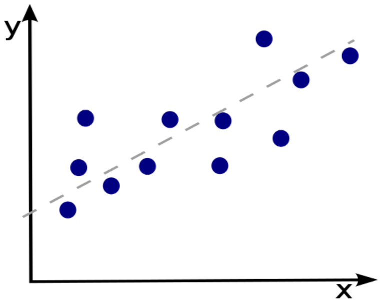


Linear regression



$$h(x) = w_0 + w_1 x$$

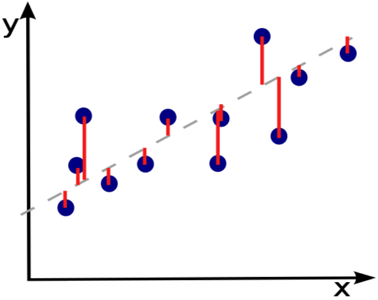
Linear regression



$$h(x) = w_0 + w_1 x$$

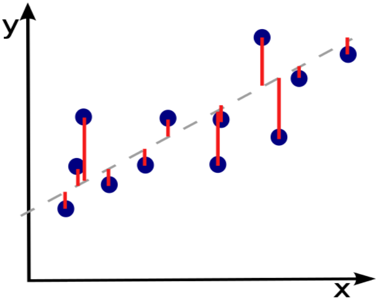
How to choose the parameter w_0 and w_1?

Linear regression



$$h(x) = w_0 + w_1 x$$

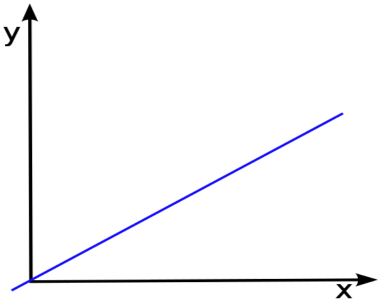
Cost function to estimate the quality of the current solution (Gradient descent).



$$h(x) = w_0 + w_1 x$$

$$\text{minimize } E[w_0, w_1] = \frac{1}{2n} \sum_{i=1}^{n} \left(h(x_i) - y_i\right)^2$$

Gradient descent cost function – intuition

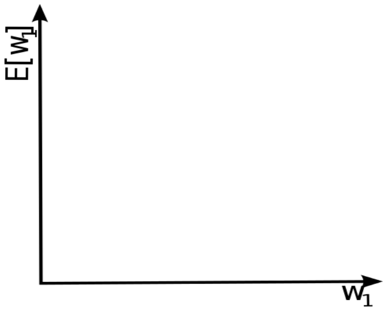


$$
\begin{aligned}
h(x) &= w_0 + w_1 x \\
\text{minimize } E[w_0, w_1] &= \frac{1}{2n}\sum_{i=1}^{n}\left(h(x_i) - y_i\right)^2
\end{aligned}
$$

For fixed w_1 this is a function of x

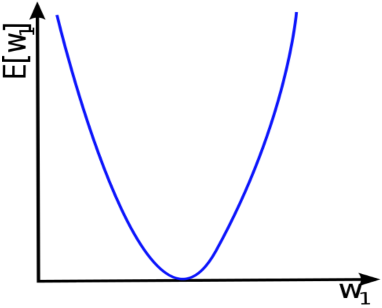
(additive constant w_0 ignored in this figure)

Gradient descent cost function – intuition



$$h(x) = w_0 + w_1 x$$

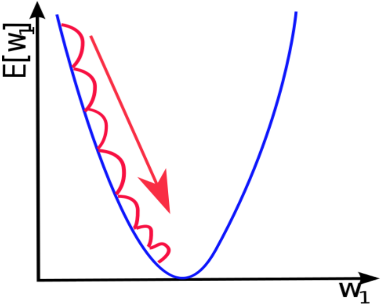
$$\text{minimize } E[w_0, w_1] = \frac{1}{2n} \sum_{i=1}^{n} \left(h(x_i) - y_i\right)^2$$



$$\begin{aligned} h(x) &= w_0 + w_1 x \\ \text{minimize } E[w_0, w_1] &= \frac{1}{2n} \sum_{i=1}^{n} (h(x_i) - y_i)^2 \end{aligned}$$

(additive constant w_0 ignored in this figure)

Gradient descent cost function – Gradient descent



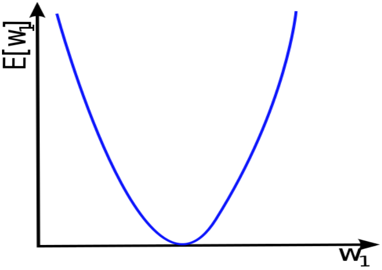
$$\text{minimize } E[w_0, w_1] \quad = \frac{1}{2n} \sum_{i=1}^{n} \left(h(x_i) - y_i\right)^2$$

$$\text{E.g.:} w_1 = w_1 - \delta \cdot \frac{\partial}{\partial w_1} E[w_0, w_1]$$

Iterative approximation of w_1

Outline

Least squares estimation



Given an error function

$$E[w_0, w_1] = \sum_{i=1}^{n} (y_i - (w_0 + w_1 x))^2$$

we can minimize the error by requiring

$$\frac{\partial E}{\partial w_0} = 0, \frac{\partial E}{\partial w_1} = 0$$

Least squares estimation

Given an error function

$$E[w_0, w_1] = \sum_{i=1}^{n} (y_i - (w_0 + w_1 x))^2$$

we can minimize the error by requiring

$$\frac{\partial E}{\partial w_0} = 0, \frac{\partial E}{\partial w_1} = 0$$

Differentiation yields

$$\frac{\partial E}{\partial w_0} = \sum_{i=1}^{n} 2\,(y_i - (w_1 x_i + w_0)) \cdot 1$$

$$\frac{\partial E}{\partial w_1} = \sum_{i=1}^{n} 2\,(y_i - (w_1 x_i + w_0)) \cdot (-x_i)$$

Least squares estimation

$$\frac{\partial E}{\partial w_0} = \sum_{i=1}^{n} 2\left(y_i - (w_1 x_i + w_0)\right) \cdot 1$$

$$\frac{\partial E}{\partial w_1} = \sum_{i=1}^{n} 2\left(y_i - (w_1 x_i + w_0)\right) \cdot (-x_i)$$

Setting

$$\frac{\partial E}{\partial w_0} = \frac{\partial E}{\partial w_1} = 0$$

will lead to

$$\sum_{i=1}^{n} (y_i - (w_1 x_i + w_0)) \cdot x_i = 0$$

$$\sum_{i=1}^{n} (y_i - (w_1 x_i + w_0)) = 0$$

Least squares estimation

$$\begin{aligned}\sum_{i=1}^{n}(y_i - (w_1 x_i + w_0)) \cdot x_i &= 0 \\ \sum_{i=1}^{n}(y_i - (w_1 x_i + w_0)) &= 0\end{aligned}$$

rewrite as

$$\begin{aligned}\left(\sum_{i=1}^{n} x_i^2\right) w_1 + \left(\sum_{i=1}^{n} x_i\right) w_0 &= \textstyle\sum_{i=1}^{n} x_i y_i \\ \left(\sum_{i=1}^{n} x_i\right) w_1 + \left(\sum_{i=1}^{n} 1\right) w_0 &= \textstyle\sum_{i=1}^{n} y_i\end{aligned}$$

Least squares estimation

$$\left(\sum_{i=1}^{n} x_i^2\right) w_1 + \left(\sum_{i=1}^{n} x_i\right) w_0 = \sum_{i=1}^{n} x_i y_i$$

$$\left(\sum_{i=1}^{n} x_i\right) w_1 + \left(\sum_{i=1}^{n} 1\right) w_0 = \sum_{i=1}^{n} y_i$$

Consequently, values of w_0 and w_1 that minimize the error satisfy

$$\begin{pmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \end{pmatrix} \begin{pmatrix} w_1 \\ w_0 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{pmatrix}$$

Least squares estimation

$$\begin{pmatrix} \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n 1 \end{pmatrix} \begin{pmatrix} w_1 \\ w_0 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n x_i y_i \\ \sum_{i=1}^n y_i \end{pmatrix}$$

for an invertible matrix this implies

$$\begin{pmatrix} w_1 \\ w_0 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n 1 \end{pmatrix}^{-1} \begin{pmatrix} \sum_{i=1}^n x_i y_i \\ \sum_{i=1}^n y_i \end{pmatrix}$$

By solving this linear equation system, optimal values of w_0 and w_1 can be determined.

Least squares estimation

$$\begin{pmatrix} \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n 1 \end{pmatrix} \begin{pmatrix} w_1 \\ w_0 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n x_i y_i \\ \sum_{i=1}^n y_i \end{pmatrix}$$

for an invertible matrix this implies

$$\begin{pmatrix} w_1 \\ w_0 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n 1 \end{pmatrix}^{-1} \begin{pmatrix} \sum_{i=1}^n x_i y_i \\ \sum_{i=1}^n y_i \end{pmatrix}$$

By solving this linear equation system, optimal values of w_0 and w_1 can be determined.

However, for least squares to be applicable, it is necessary that the matrix is invertible.

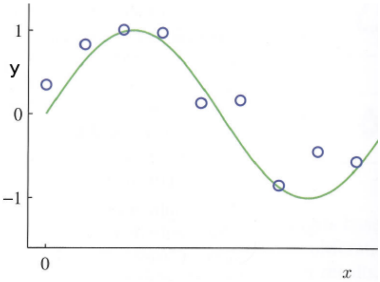
Outline

Polynomial regression (Polynomial curve fitting)

Example

A curve shall be approximated by a machine learning approach

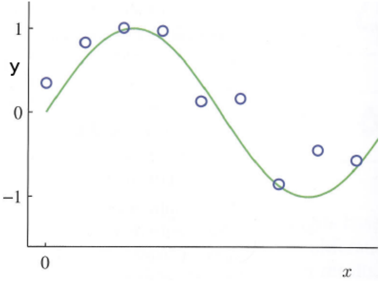
Sample points are created for the function $\sin(2\pi x) + \mathcal{N}$ where $\mathcal{N}$ is a random noise value



Polynomial curve fitting

We will try to fit the data points into a polynomial function:

$$h(x, \vec{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$



Polynomial curve fitting

We will try to fit the data points into a polynomial function:

$$h(x, \overrightarrow{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

This can be obtained by minimising an error function that measures the misfit between $h(x, \overrightarrow{w})$ and the training data set:

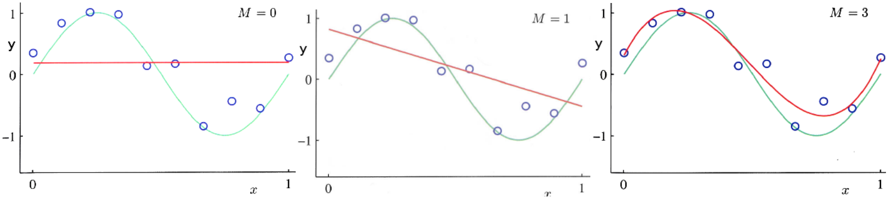
$$E(\overrightarrow{w}) = \frac{1}{2} \sum_{i=1}^{N} \left[h(x_i, \overrightarrow{w}) - y_i \right]^2$$

$E(\overrightarrow{w})$ is non-negative and zero if and only if all points are covered by the function

Polynomial curve fitting

One problem is the right choice of the dimension M

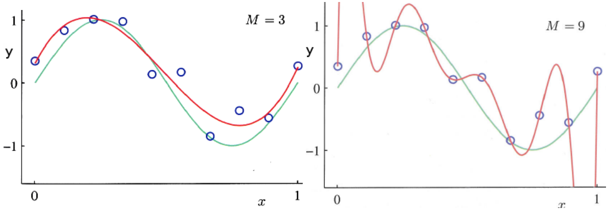
When M is too small, the approximation accuracy might be bad



Polynomial curve fitting

However, when M becomes too big, the resulting polynomial will cross all points exactly

When M reaches the count of samples in the training data set, it is always possible to create a polynomial of order M that contains all values in the data set exactly.



Polynomial curve fitting

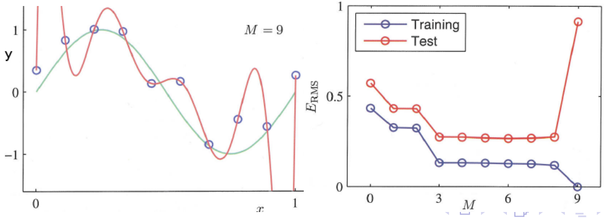
This event is called overfitting

The polynomial is now trained too well to the training data

It will therefore perform badly on another sample of test data for the same phenomenon

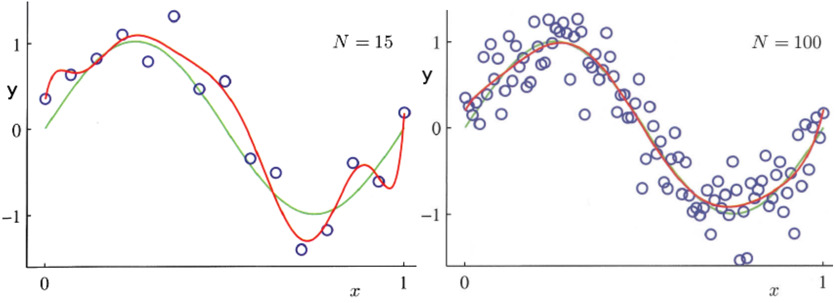
We visualise it by the Root of the Mean Square (RMS) of $E(\overrightarrow{w})$

$$E_{RMS} = \sqrt{\frac{2E(\overrightarrow{w})}{N}}$$



Polynomial curve fitting

With increasing number of data points, the problem of overfitting becomes less severe for a given value of M



Polynomial curve fitting

One solution to cope with overfitting is regularisation

A penalty term is added to the error function

This term discourages the coefficients of $\overrightarrow{w}$ from reaching large values

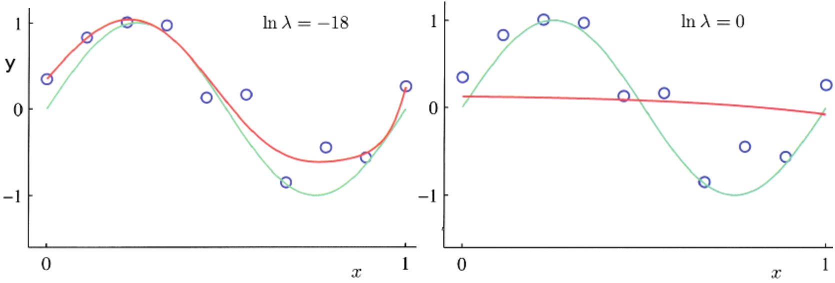
$$\overline{E}(\overrightarrow{w}) = \frac{1}{2}\sum_{i=1}^{N}\left[h(x_i, \overrightarrow{w}) - y_i\right]^2 + \frac{\lambda}{2}||\overrightarrow{w}||^2$$

with

$$||\overrightarrow{w}||^2 = \overrightarrow{w}^T\overrightarrow{w} = w_0^2 + w_1^2 + \cdots + w_M^2$$

Polynomial curve fitting

Depending on the value of λ, overfitting is controlled



$$\overline{E}(\vec{w}) = \frac{1}{2}\sum_{i=1}^{N}\left[h(x_i,\vec{w}) - y_i\right]^2 + \frac{\lambda}{2}||\vec{w}||^2$$

Outline

Multivariable linear regression

In multivariable linear regression problems we assume that multiple regression variables (features) apply.

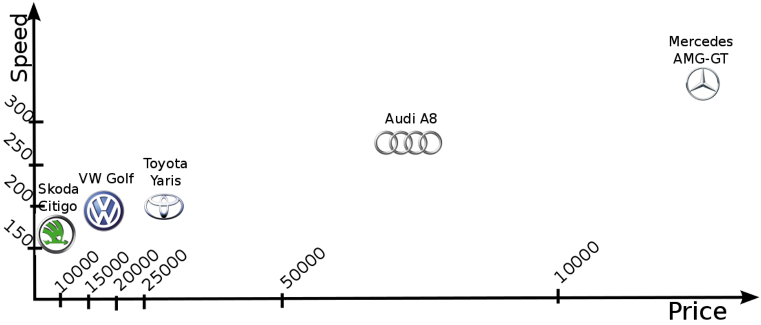
Multivariable linear regression

In multivariable linear regression problems we assume that multiple regression variables (features) apply.

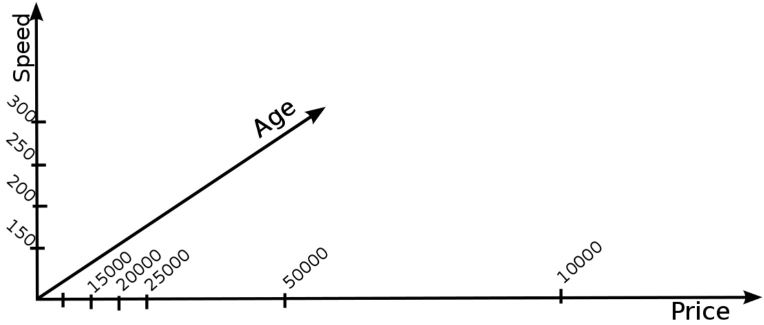
$$h(x_{j1}, \ldots, x_{jm}) = \sum_{i=0}^{m} w_i x_{ji}$$

$$\text{minimize } E[W] = \frac{1}{2n} \sum_{j=1}^{n} (h(x_{j1}, \ldots, x_{jm}) - y_j)^2$$

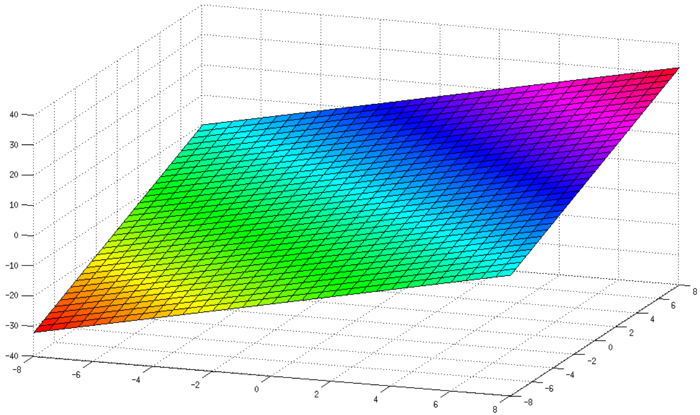
Multivariable linear regression



Multivariable linear regression



Multivariable linear regression



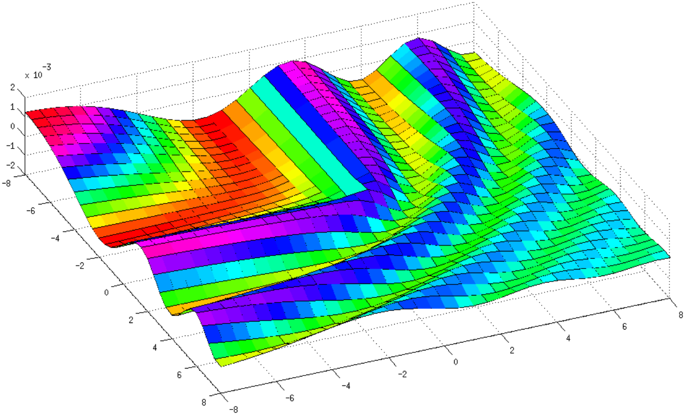
$$
\begin{aligned}
h(x_{j1}, \ldots, x_{jm}) &= \sum_{i=0}^{m} w_i x_{ji} \\
\text{minimize } E[W] &= \frac{1}{2n} \sum_{j=1}^{n} \left(h(x_{j1}, \ldots, x_{jm}) - y_j \right)^2
\end{aligned}
$$

$$
\begin{aligned}
h(x_{j1}, \ldots, x_{jm}) &= \sum_{i=0}^{m} w_i x_{ji} \\
\text{minimize } E[W] &= \frac{1}{2n} \sum_{j=1}^{n} \left(h(x_{j1}, \ldots, x_{jm}) - y_j \right)^2
\end{aligned}
$$

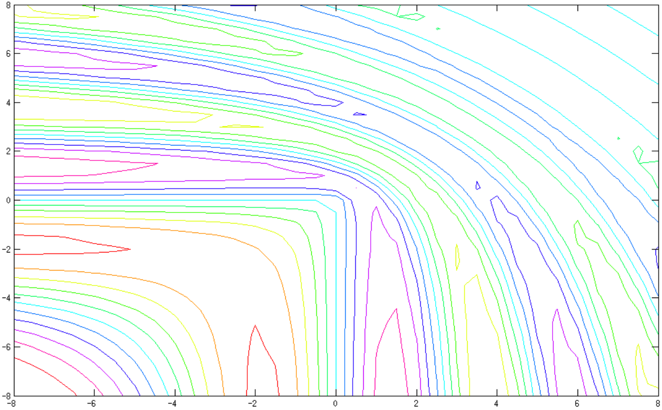
$$
\text{E.g.:} w_i = w_i - \delta \cdot \frac{\partial}{\partial w_i} E[W]
$$

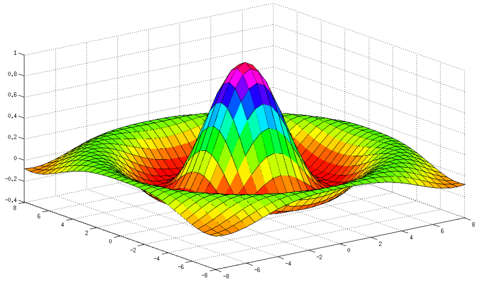
w_i are optimised together over several iterations

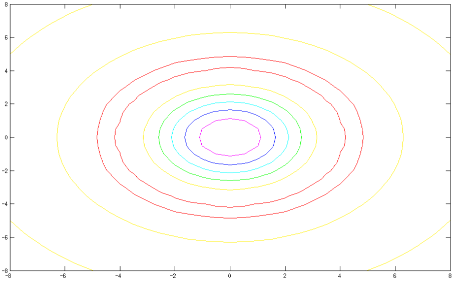
Local optima



Local optima – contour plot




1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
8 6 4 2 0 -2 -4 -6 -8
-8 -6 -4 -2 0 2 4 6 8


8
6
4
2
0
-2
-4
-6
-8
-8 -6 -4 -2 0 2 4 6 8

Outline

Multivariate linear regression

Multivariate linear regression describes a regression problem with multiple classes.

Example e.g. from accelerometer data

- **Activities** walking, standing, climbin/descending stairs, ...
- **Sentiment** emotional states
- **Transportation mode** office, riding tram, driving ...
- **Location** Home, office, ...

Multivariate linear regression

Regression model is extended to multiple responses with respect to one class: $Y_j = y_{j1}, \ldots, y_{jl}$

$$\begin{aligned} y_{j1} &= w_{01} + w_{11}x_{j1} + \cdots + w_{n1}x_{jn} \\ y_{j2} &= w_{02} + w_{12}x_{j1} + \cdots + w_{n2}x_{jn} \\ \vdots & \qquad\qquad \vdots \\ y_{jl} &= w_{0l} + w_{1l}x_{j1} + \cdots + w_{nl}x_{jn} \end{aligned}$$

Multivariate linear regression

Regression model is extended to multiple responses with respect to one class: $Y_j = y_{j1}, \ldots, y_{jl}$

$$\begin{aligned} y_{j1} &= w_{01} + w_{11}x_{j1} + \cdots + w_{n1}x_{jn} \\ y_{j2} &= w_{02} + w_{12}x_{j1} + \cdots + w_{n2}x_{jn} \\ \vdots & \qquad\qquad \vdots \\ y_{jl} &= w_{0l} + w_{1l}x_{j1} + \cdots + w_{nl}x_{jn} \end{aligned}$$

It is then possible to estimate the regression coefficients associated with y_{ji} using only the i-th row of the matrix.

Multivariate linear regression

It is then possible to estimate the regression coefficients associated with y_{ji} using only the i-th row of the matrix.

$$W_i = \left(X^T X\right)^{-1} X^T Y_{(i)}$$

and collecting all univariate estimates into a matrix

$$W = \left(X^T X\right)^{-1} X^T Y$$

$Y_{(i)}$ is the vector of n measurements of the i-th variable
X^T denotes the transpose of X and X^{-1} its inverse

Multivariate linear regression

The least squares estimator for W minimizes the sums of squares elements on the diagonal of the residual sum of squares and crossproducts matrix $(Y - XW)^T(Y - XW)$

Outline

Logistic regression

Nominal classes

Classes might be nominal in real-world problems

Logistic regression

Nominal classes

Classes might be nominal in real-world problems

Weather	Sunny, rainy
Medical	positive diagnosis, negative diagnosis
Localisation	indoor, outdoor

Logistic regression

Nominal classes

Classes might be nominal in real-world problems

Weather	Sunny, rainy
Medical	positive diagnosis, negative diagnosis
Localisation	indoor, outdoor

In such case, classification is binary: $y \in \{0, 1\}$

Logistic regression

Nominal classes

Classes might be nominal in real-world problems

Weather	Sunny, rainy
Medical	positive diagnosis, negative diagnosis
Localisation	indoor, outdoor

In such case, classification is binary: $y \in \{0, 1\}$

Linear regression: $h(x)$ can be smaller than 0 or greater than 1

Logistic regression

Nominal classes

Classes might be nominal in real-world problems

Weather	Sunny, rainy
Medical	positive diagnosis, negative diagnosis
Localisation	indoor, outdoor

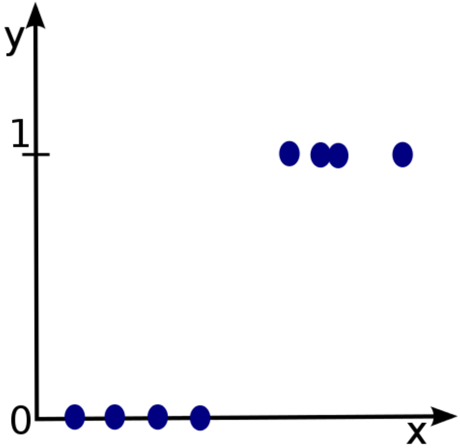
In such case, classification is binary: $y \in \{0, 1\}$

Linear regression: $h(x)$ can be smaller than 0 or greater than 1

Logistic regression: $0 \leq h(x) \leq 1$

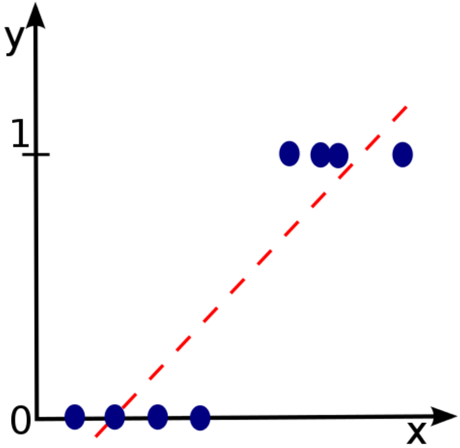
Logistic regression

Nominal classes



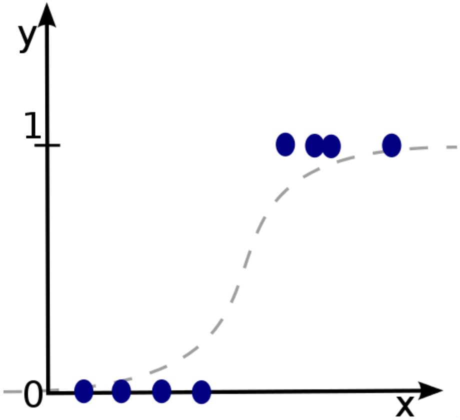
Logistic regression

Nominal classes



Logistic regression

Cost function



Logistic regression

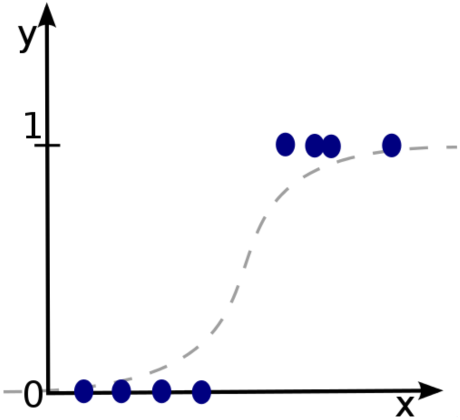
Cost function

Linear regression

$$h(x) = W^T x$$

Logistic regression

$$h(x) = \frac{1}{1+e^{W^T x}}$$



Logistic regression

Cost function

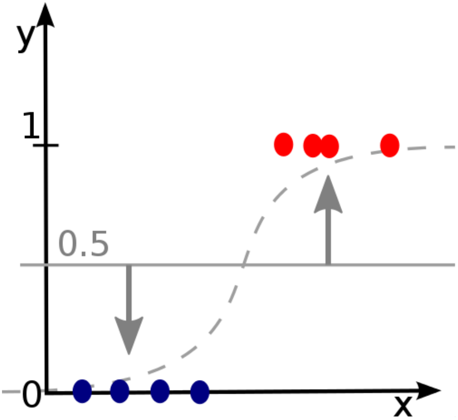
Linear regression

$$h(x) = W^T x$$

Logistic regression

$$h(x) = \frac{1}{1+e^{W^T x}}$$

$$y = \begin{cases} 1 & \text{if } h(x) \geq 0.5 \\ 0 & \text{else} \end{cases}$$

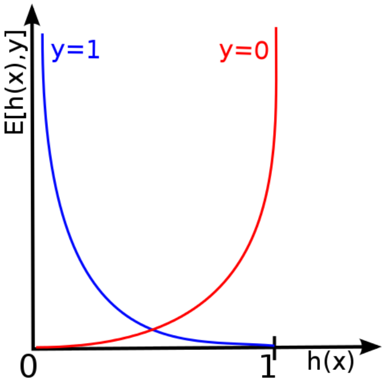


Logistic regression

Cost function

$$y = \begin{cases} 1 & \text{if } h(x) \geq 0.5 \\ 0 & \text{else} \end{cases}$$

$$E[h(x), y] = \begin{cases} -\log(h(x)) & \text{if } y = 1 \\ -\log(1 - h(x)) & \text{else} \end{cases}$$



Outline

Questions?

Stephan Sigg
stephan.sigg@cs.uni-goettingen.de

Literature

- C.M. Bishop: Pattern recognition and machine learning, Springer, 2007.
- R.O. Duda, P.E. Hart, D.G. Stork: Pattern Classification, Wiley, 2001.