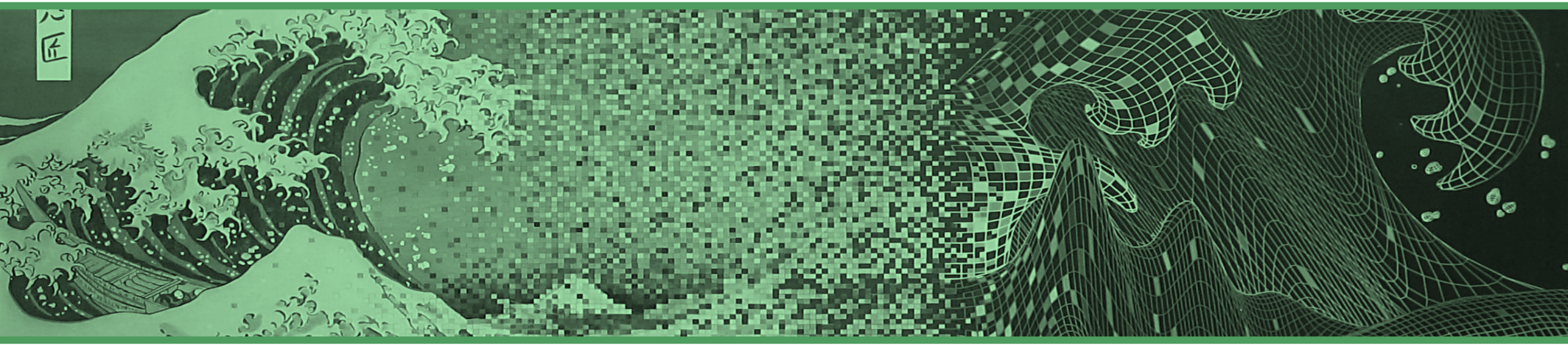
TÉCNICO LISBOA

# Linear and Bayesian regression

## Regression approaches for predictive tasks



Aprendizagem 2024

Miguel Couceiro miguel.j.couceiro@tecnico.ulisboa.pt
Andreas Wischert andreas.wichert@tecnico.ulisboa.pt

# Outline

- **Regression as a task** *versus* **an approach**
- **Linear regression**
  - multiple linear regression
  - Moore-Penrose and pseudoinverse solutions
- **Non-linear regression**
  - principles
  - linear basis functions
- **Bayesian regression**
  - regularization

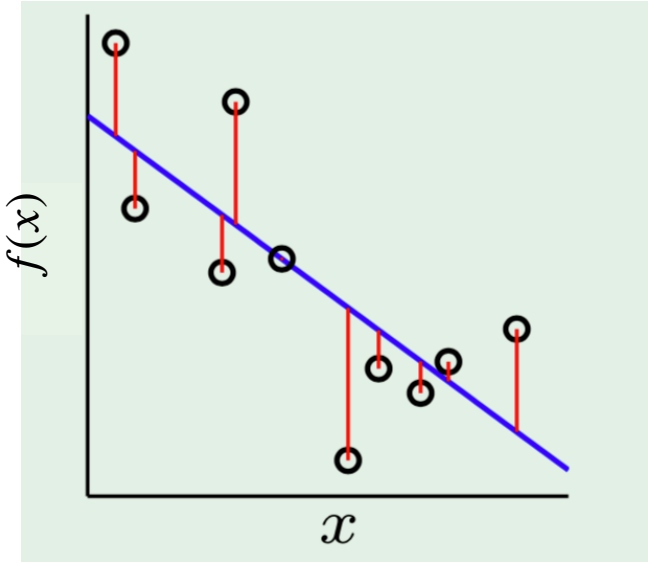
# Regression as a task *versus* approach

- Until here, **regression** defined as a descriptive or predictive **learning task**, $f: X \rightarrow \mathbb{R}$
  - e.g. decision tree regressor, kNN regressor

  - *descriptive setting*: given a set of observations, describe the relation between a set of (explanatory) attributes and a target real-valued attribute
  - *predictive setting*: given a set of observations with a real-valued outcome, learn a mapping to estimate the outcome of new observations

- However, **regression** is also used in the field to denote a class of **learning approaches**
  - *linear regression* for regression tasks
  - *logistic regression* for classification tasks

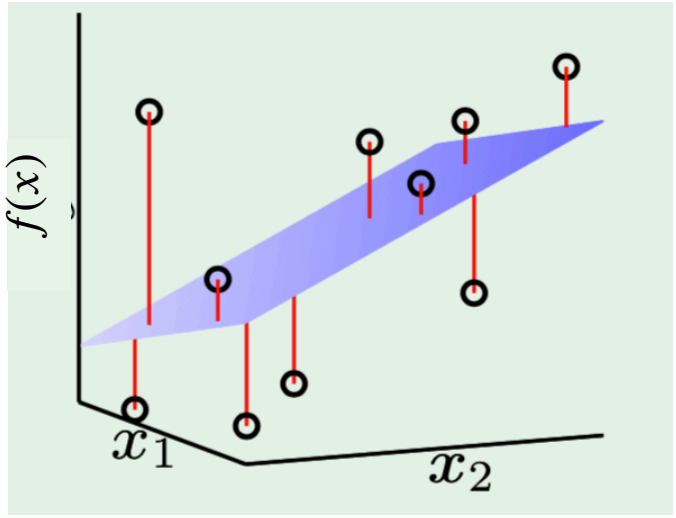
# Regression of a line




  - $w_0$ being the intercept and $w_1$ being the slope of the line
    - if $w_0 = 0$, the line goes through the origin
- How good is my regression? function of the residuals (*red lines*)
  - use testing (training) observations to assess the testing (training) errors
- Univariate data! How to go multivariate?

# Linear Regression

- Simple linear model (**hyperplane**) is the linear combination:




$$f(\mathbf{x}) = w_0 + \sum_{j=1}^{m} w_j \bullet x_j = w_0 + \langle \mathbf{w} \,|\, \mathbf{x} \rangle$$

  - **parameters** $w_j$ control the behavior of the system
    - intercept $w_0$ is often call the bias parameter of the affine transformation
      - i.e. the function is biased towards $w_0$ in the absence of any input!
    - $\mathbf{w} = (w_1, \ldots, w_m)$ are often referred as predictors

# Outline

- Regression as a task *versus* an approach
- **Linear regression**
  - **multiple linear regression**
  - **Moore-Penrose and pseudoinverse solutions**
- Non-linear regression
  - principles
  - linear basis functions
- Bayesian regression
  - regularization

# Mean squared error

- If we assume a new variable $y_0$ so that observation $\mathbf{x}$ takes $x_0 = 1$, we can simplify the formula

$$f(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M} w_j \bullet x_j = \langle \mathbf{w} | \mathbf{x} \rangle$$

- The training set consists of n observations: $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_n)^T$
- Together with 8numeric) targets: $\mathbf{z} = (z_1, z_2, \ldots, z_i, \ldots, z_n)^T$

- The **mean square error** over all training observations:

$$E(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} (f(\mathbf{x}_i, \mathbf{w}) - z_i)^2 = \frac{1}{n} \|\hat{\mathbf{z}} - \mathbf{z}\|^2$$

# Sum of squares error

- When optimizing the hyperplane… we may want to compare different hyperplanes
  - either dividing the squared errors by $n$ or not will yield same answer
  - sum of squares function over all $n$ training observations:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}\left(f\left(\mathbf{x}_i, \mathbf{w}\right) - z_i\right)^2 = \frac{1}{2} \bullet \left\|\hat{\mathbf{z}} - \mathbf{z}\right\|^2 = \frac{1}{2} \bullet \left\|\mathbf{z} - \hat{\mathbf{z}}\right\|^2$$

  - scaled by ½ Euclidean between predictions and target

$$E(\mathbf{w}) = \frac{1}{2} \bullet \left\| \begin{pmatrix} z_1 \\ \vdots \\ z_i \\ \vdots \\ z_n \end{pmatrix} - \begin{pmatrix} \hat{z}_1 \\ \vdots \\ \hat{z}_i \\ \vdots \\ \hat{z}_n \end{pmatrix} \right\|^2$$

# Design matrix and linear mapping

- The data is also with $xj0$=1 is called *designed matrix*

$$X = \begin{pmatrix} \mathbf{x}_1{}^T \\ \mathbf{x}_2{}^T \\ \vdots \\ \mathbf{x}_n{}^T \end{pmatrix} = \begin{pmatrix} x11 & x12 & \cdots & x1m \\ x21 & x22 & \cdots & x2m \\ \vdots & \vdots & \cdots & \vdots \\ xn1 & xn2 & & xnm \end{pmatrix}$$

  - … we can specify the target linear mapping as

$$\begin{pmatrix} \hat{z}_1 \\ \hat{z}_2 \\ \vdots \\ \hat{z}_n \end{pmatrix} = \begin{pmatrix} 1 & x11 & x12 & \cdots & x1m \\ 1 & x21 & x22 & \cdots & x2m \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & xn1 & xn2 & & xnm \end{pmatrix} \bullet \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_j \\ \vdots \\ w_m \end{pmatrix}$$

$$\boldsymbol{z} = X \bullet \mathbf{w} = (\mathbf{w}^T \bullet X^T)^T$$

# Minimizing errors

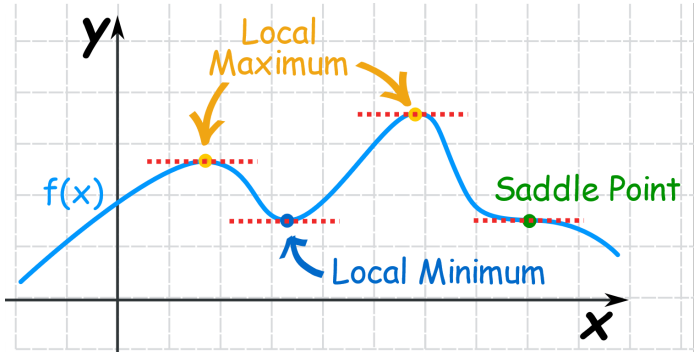
- To find the best hyperplane, we need to minimize the error:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(z_i - f(\mathbf{x}_i, \mathbf{w}))^2 = \frac{1}{2} \bullet \left\|\mathbf{z} - \hat{\mathbf{z}}\right\|^2$$

**NB:** using the introduced notation, we can rewrite it as:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(z_i - \mathbf{w}^T \bullet \mathbf{x}_i)^2 = \frac{1}{2} \bullet \left\|\mathbf{z} - X \bullet \mathbf{w}\right\|^2 = \frac{1}{2} \bullet (\mathbf{z} - X \bullet \mathbf{w})^T(\mathbf{z} - X \bullet \mathbf{w})$$

- How to minimize the error?
  - find minimum of the error function to see where solution lies
  - how? points where slope is zero: zeros of the derivative
    - in vector calculus, derivative is termed gradient



# Least squares estimation (1/2)

- To minimize the observed errors, we set the gradient of E(w) to zero
  - gradient operator: $\nabla = \left[\frac{\partial}{\partial w_1}, \frac{\partial}{\partial w_2}, \ldots, \frac{\partial}{\partial w_m}\right]^T$
  - applied to $E(\mathbf{w})$…

$$\nabla E(\mathbf{w}) = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_m}\right]^T = \nabla\left(\frac{1}{2} \bullet (\mathbf{z} - X \bullet \mathbf{w})^T (\mathbf{z} - X \bullet \mathbf{w})\right) = 0$$

- Gradient rules…

$$\nabla(a \cdot f(\mathbf{w}) + b \cdot g(\mathbf{w})) = a \cdot \nabla f(\mathbf{w}) + b \cdot \nabla g(\mathbf{w}), \quad a, b \in \mathbb{R}$$

$$\text{and for } A = X^T \cdot X \text{ symmetric}$$

$$\nabla\left((\mathbf{w}^T \cdot A \cdot \mathbf{w}\right) = 2 \cdot A \cdot \mathbf{w}$$

$$\nabla_w\left(\mathbf{z}^T \mathbf{w}\right) = \mathbf{z}$$

# Least squares estimation (2/2)

$$\nabla E(\mathbf{w}) = \nabla\left(\frac{1}{2} \bullet (\mathbf{z} - X \bullet \mathbf{w})^T (\mathbf{z} - X \bullet \mathbf{w})\right) = 0$$

$$\nabla\left(\mathbf{z}^t \bullet \mathbf{z} - 2\mathbf{z}^T \bullet X \bullet \mathbf{w} + \mathbf{w}^T \bullet X^T \bullet X \bullet \mathbf{w}\right) = 0$$

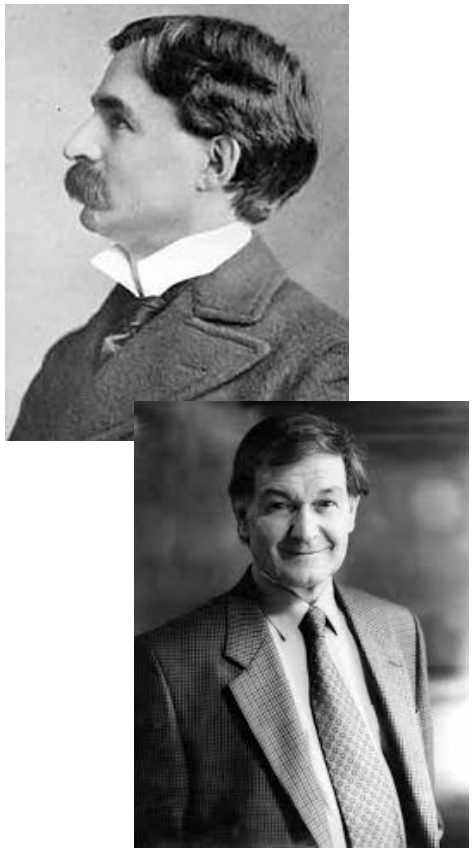
$$\nabla\left(\mathbf{z}^t \bullet \mathbf{z}\right) - 2\nabla\left(\mathbf{z}^T \bullet X \bullet \mathbf{w}\right) + \nabla\left(\mathbf{w}^T \bullet X^T \bullet X \bullet \mathbf{w}\right) = 0$$

$$-2X^T\mathbf{z} + 2X^T \bullet X \bullet \mathbf{w} = 0$$

$$X^T\mathbf{z} = X^T \bullet X \bullet \mathbf{w}$$

$$\left(X^T \bullet X\right)^{-1} \bullet X^T \bullet \mathbf{z} = \mathbf{w}$$

The matrix $X^+ = (X^T X)^{-1} X^T$ is called the *Moore-Penrose inverse* of $X$ (or the pseudo-inverse of $X$)



# Multivariate linear regression

$$\hat{z} = f(\mathbf{x}) = w_0 + w_1 x_1 + .. + w_m x_m$$

- **Train**: estimate parameters

$$\text{data } X = \begin{pmatrix} 1 & x11 & x12 & \cdots & x1m \\ 1 & x21 & x22 & \cdots & x2m \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & xn1 & xn2 & \cdots & xnm \end{pmatrix} \quad \text{outcome } z = \begin{pmatrix} z_1 \\ \vdots \\ z_n \end{pmatrix}$$

$$\mathbf{w} = \left(X^T \bullet X\right)^{-1} \bullet X^T \bullet \mathbf{z} \text{ where } \mathbf{w} = \begin{pmatrix} w_0 \\ \vdots \\ w_m \end{pmatrix}$$

- **Test**: given a specific observation $\mathbf{x}_{new} = [x_{new\ 1}, \ldots, x_{new\ m}]$ compute:

$$\hat{z} = f\left(\mathbf{x}_{new}\right) = w_0 + w_1 x_{new\ 1} + .. + w_m x_{new\ m}$$

# Multivariate linear regression: example

| | | | |
|---|---|---|---|
| 1 | 41.9 | 29.1 | 251.3 |
| 2 | 43.4 | 29.3 | 251.3 |
| 3 | 43.9 | 29.5 | 248.3 |
| 4 | 44.5 | 29.7 | 267.5 |
| 5 | 47.3 | 29.9 | 273.0 |
| 6 | 47.5 | 30.3 | 276.5 |
| 7 | 47.9 | 30.5 | 270.3 |
| 8 | 50.2 | 30.7 | 274.9 |
| 9 | 52.8 | 30.8 | 285.0 |
| 10 | 53.2 | 30.9 | 290.0 |
| 11 | 56.7 | 31.5 | 297.0 |
| 12 | 57.0 | 31.7 | 302.5 |
| 13 | 63.5 | 31.9 | 304.5 |
| 14 | 65.3 | 32.0 | 309.3 |
| 15 | 71.1 | 32.1 | 321.7 |
| 16 | 77.0 | 32.5 | 330.7 |
| 17 | 77.8 | 32.9 | 349.0 |

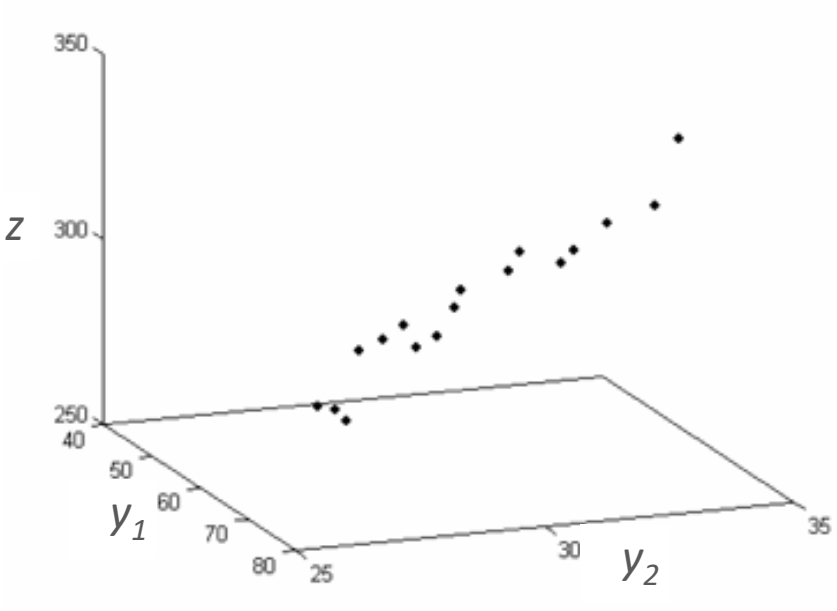
$$X = \begin{bmatrix} 1 & 41.9 & 29.1 \\ 1 & 43.4 & 29.3 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ 1 & 77.8 & 32.9 \end{bmatrix} \quad \mathbf{z} = \begin{bmatrix} 251.3 \\ 251.3 \\ \cdot \\ \cdot \\ \cdot \\ 349.0 \end{bmatrix}$$

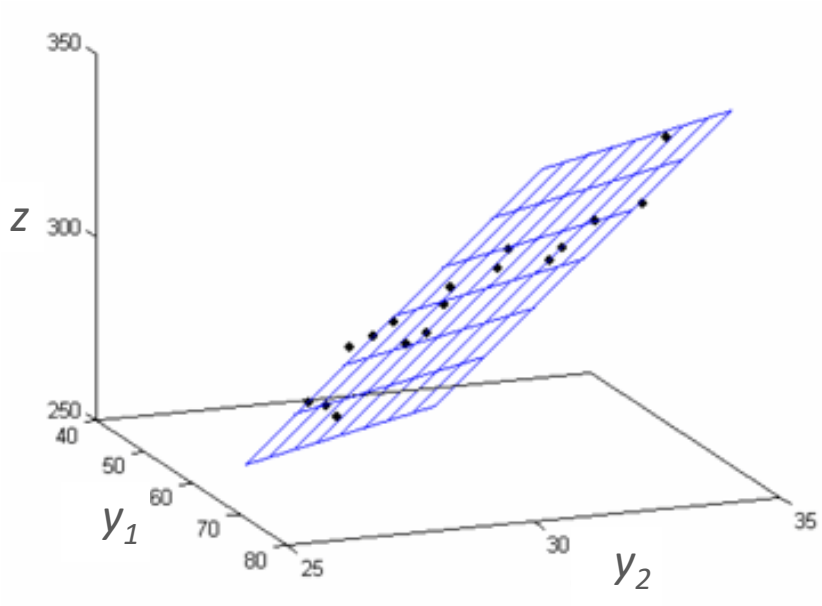
$$\mathbf{w} = \left(X^T \bullet X\right)^{-1} \bullet X^T \bullet \boldsymbol{z}$$

$$= \begin{bmatrix} 17 & 941 & 525.3 \\ 941 & 54270 & 29286 \\ 525.3 & 29286 & 16254 \end{bmatrix}^{-1} \bullet X^T \bullet \boldsymbol{z}$$

$$= \begin{bmatrix} -153.51 \\ 1.24 \\ 12.08 \end{bmatrix}$$

# Multivariate linear regression




$$\hat{z} = -153.51 + 1.24x_1 + 12.08x_2$$

Result of testing observation $\mathbf{x}_{new} = (25,50)$?

# Moore-Penrose and pseudoinverse

- If the matrix $X^T X$ is not invertible, it is said to be *singular* or *degenerate.* When?
  - some features are linear combinations of the others
  - $n < m$ (high-dimensionality, i.e. more variables than observations)

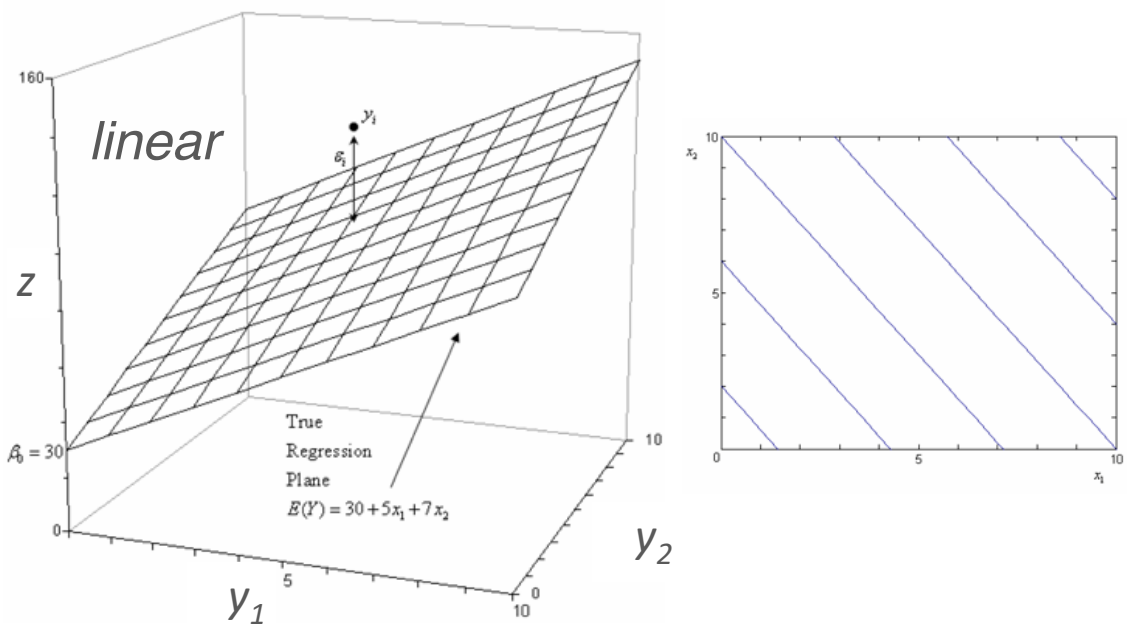
- But the **pseudo-inverse** $A^+$ of A is always defined, i.e.:
  1. $AA^+$ is the identity on A, and $A^+A$ is the identity on $A^+$
  2. $AA^+$ is *Hermitian* (iff $\mathbf{w} \bullet AA^+\mathbf{v} = AA^+\mathbf{w} \bullet \mathbf{v}$) and so is $A^+A$

- Then, how to deal with not invertible cases? **SVD decomposition** (details on a later class)
  - If $X$ is a $n \times m$ matrix, then the SVD decomposition of $X$ is $X = U \cdot D \cdot V^T$
    - $U$ is an $n \times n$ orthogonal matrix (whose columns are left-singular and whose first $r \leq \min\{n, m\}$ form a basis of the column space of $X$ and the last $n - r$ form a basis of the null space of the conjugate of $X$)
    - $D$ is a diagonal $n \times m$ matrix with non-negative real numbers on the diagonal (singular values, I.e., square roots of Eigen-values)
    - $V$ is an $m \times m$ orthogonal matrix (whose columns are right-singular and whose first $r$ form a basis of the row space of $X$ and the last $m - r$ form a basis of the null space of $X$)
  - In this case, the pseudo-inverse can be obtained by: $X^+ = VD^+U^T$, where $D^+$ can be obtained by taking the inverse of every nonzero element of the diagonal and transposing the resulting matrix

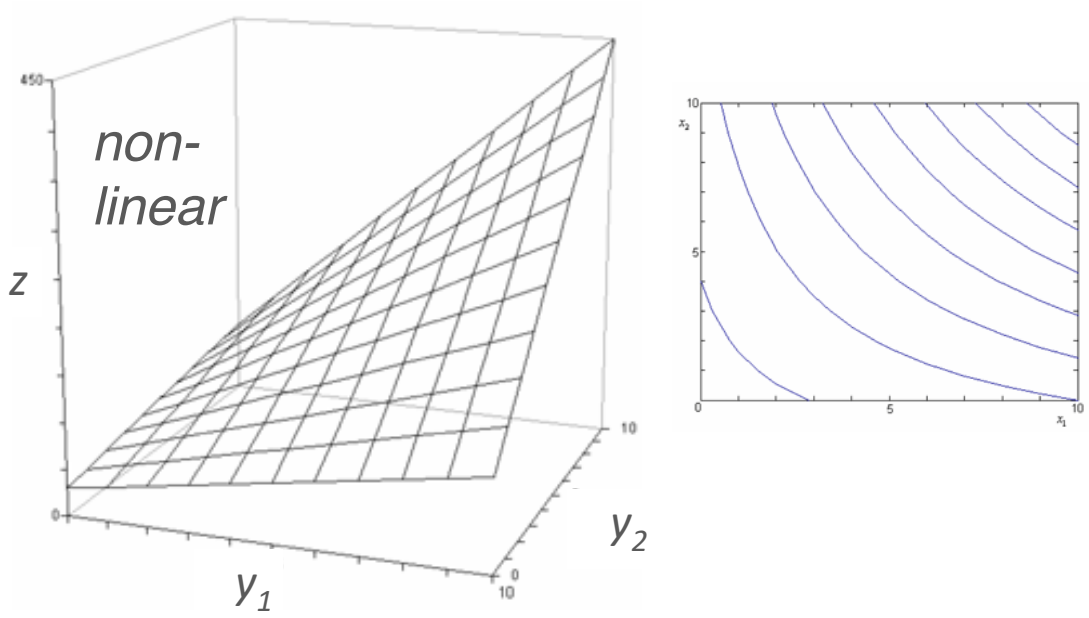
# Outline

- Regression as a task *versus* an approach
- Linear regression
  - multiple linear regression
  - Moore-Penrose and pseudoinverse solutions
- **Non-linear regression**
  - **principles**
  - **linear basis functions**
- Bayesian regression
  - regularization

# Non-linear regression



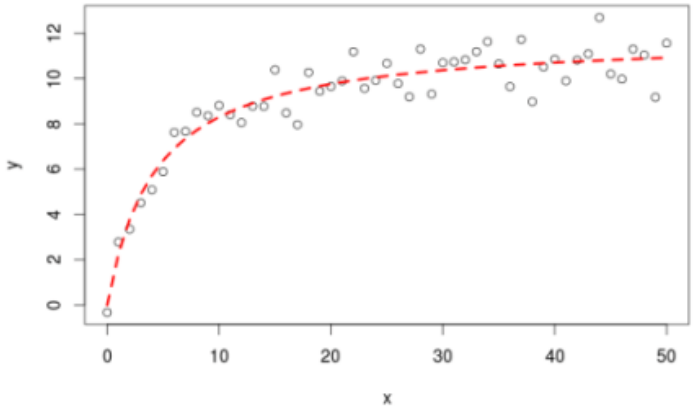

$$z = 30 + 5x_1 + 7x_2$$

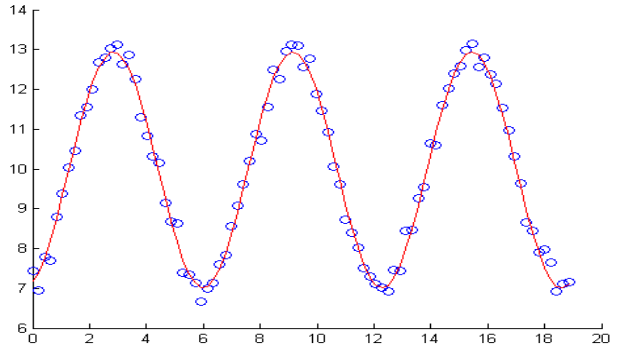



$$z = 500 + 5x_1 + 7x_2 - 3x_1^2 - 5x_2^2 + 3x_1x_2$$

# Non-linear regression

- Data may not be well explained by a linear regression




- **How-to?**
  1. Move data into a linear domain by a suitable **data transformation** (*kernel*)

     e.g. $z = ae^{bx} \Leftrightarrow \ln(z) = \ln(a) + bx$

  2. Nonlinear data can be **split up into segments** and linear regression performed per segment

# Linear basis function models

- Central idea of most non-linear regression models: use **linear regression with non-linear features**
  - non-linear regression is the linear combination of fixed nonlinear functions

$$f(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M} w_j \bullet \phi_j(\mathbf{x})$$

with $\phi_0(\mathbf{x}) = 1$

$$f(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M} w_j \bullet \phi_j(\mathbf{x}) = \langle \mathbf{w} \,|\, \Phi(\mathbf{x}) \rangle = \mathbf{w}^T \Phi(\mathbf{x})$$

  - **example**: $\Phi(\mathbf{x}) = \Phi\left(x_1, x_2, x_3\right) = (x_1 x_3, \sqrt{x_2 + x_3})$, where $\phi_2(\mathbf{x}) = \sqrt{x_2 + x_3}$
  - **example**: when basis function is $\phi_j(\mathbf{x}) = x_j$, we are in the presence of a classic linear regression

# Linear basis function models

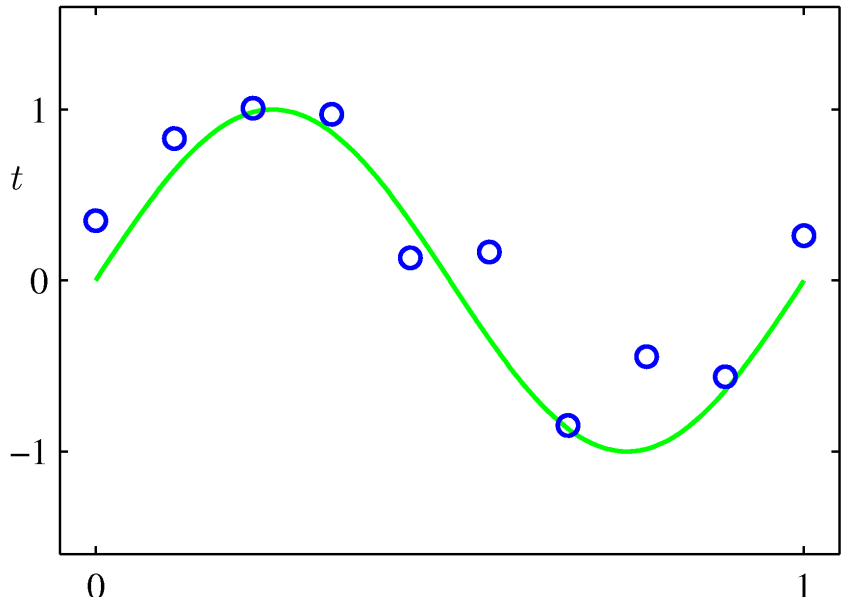
- Let $\Phi_{i,j} = \phi_j(\mathbf{x}_i)$

$$\begin{pmatrix} \hat{z}_1 \\ \hat{z}_2 \\ \vdots \\ \hat{z}_n \end{pmatrix} = \begin{pmatrix} 1 & \phi 11 & \phi 12 & \cdots & \phi 1M \\ 1 & \phi 21 & \phi 22 & \cdots & \phi 2M \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \phi n1 & \phi n2 & & \phi nM \end{pmatrix} \cdot \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_j \\ \vdots \\ w_M \end{pmatrix}$$

**Here:** $\Phi^+$is the Moore-Penrose or pseudo-inverse of $\Phi$, i.e., $\Phi^+ = (\Phi^T\Phi)^{-1}\Phi^T$

# Linear basis function models

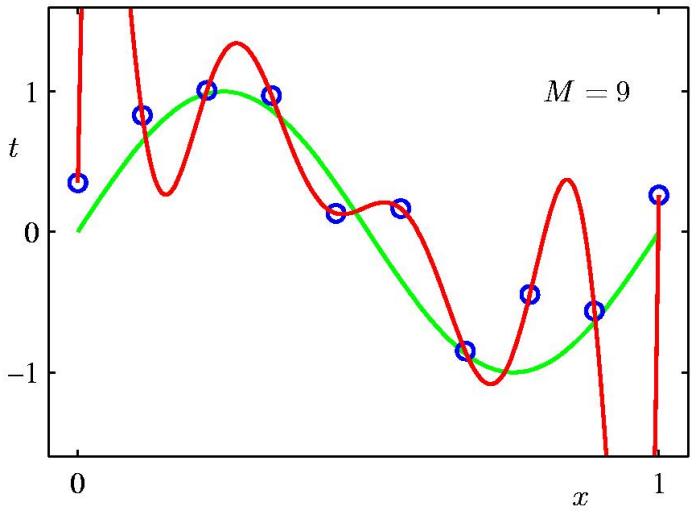
- **Another example:** *polynomial curve fitting* (univariate data)
  - $\phi_j(x) = x^j$




$$f(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

# Linear basis function models

- $m$ (dimensionality on the original space) and $M$ (dimensionality of the transformed space) may differ
  - $m > M$ going into lower-dimensional spaces, e.g. $\Phi(x_1, x_2, x_3) = (x_1 x_3, \sqrt{x_2 + x_3})$
  - $m < M$ going into higher-dimensional space
  - example: approximate univariate data with 9-degree polynomial
    - $m = 1, M = 9$
    - 10 parameters to estimate $\mathbf{w} = [w_0 \dots w_9]^T$



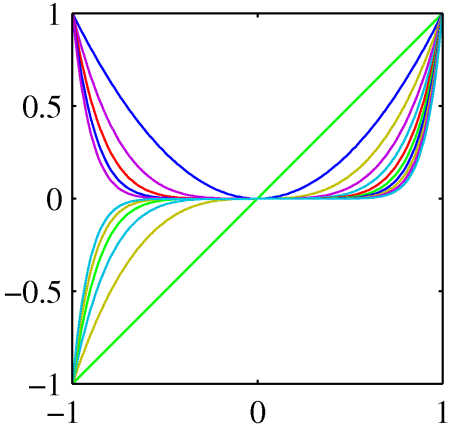
# Basis function models

$$f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \Phi(\mathbf{x}) = \sum_{j=1}^{M} w_j \phi_j(\mathbf{x})$$

- ***Polynomial basis*** functions: $\phi_j(\mathbf{x}) = \mathbf{x}^j$

  Example: $\phi_2 \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \\ 9 \end{pmatrix}$

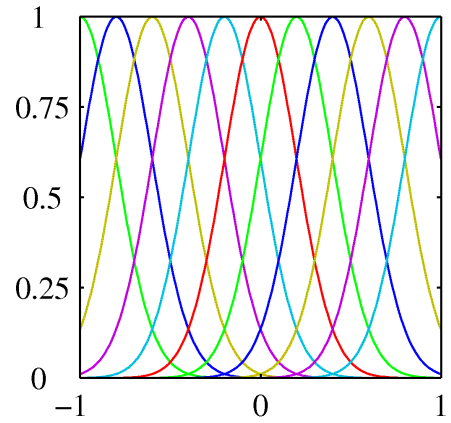
  Global: small change affect all basis functions



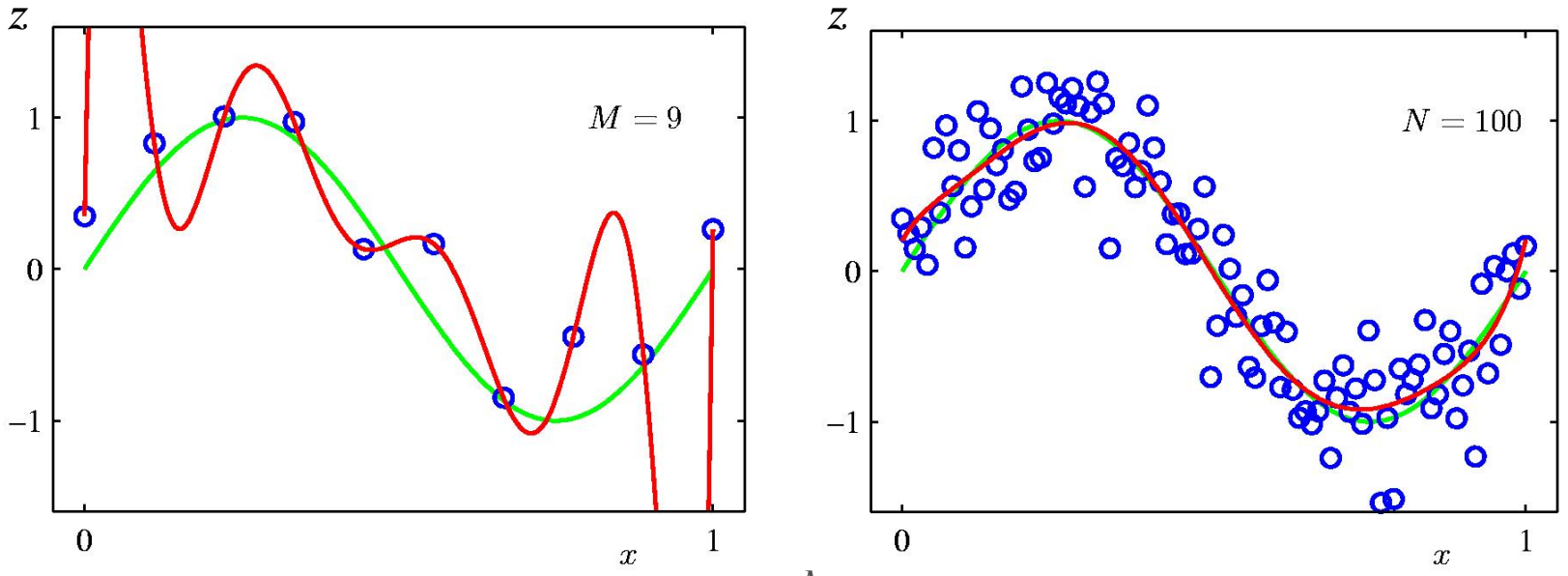

- ***Gaussian basis*** functions: $\phi_j(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{2\sigma^2}\right)$

  Example: given $\mathbf{c}_2 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$, $\sigma = 1$, then $\phi_2 \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \exp(-2)$

  Local: small change only affect nearby basis functions



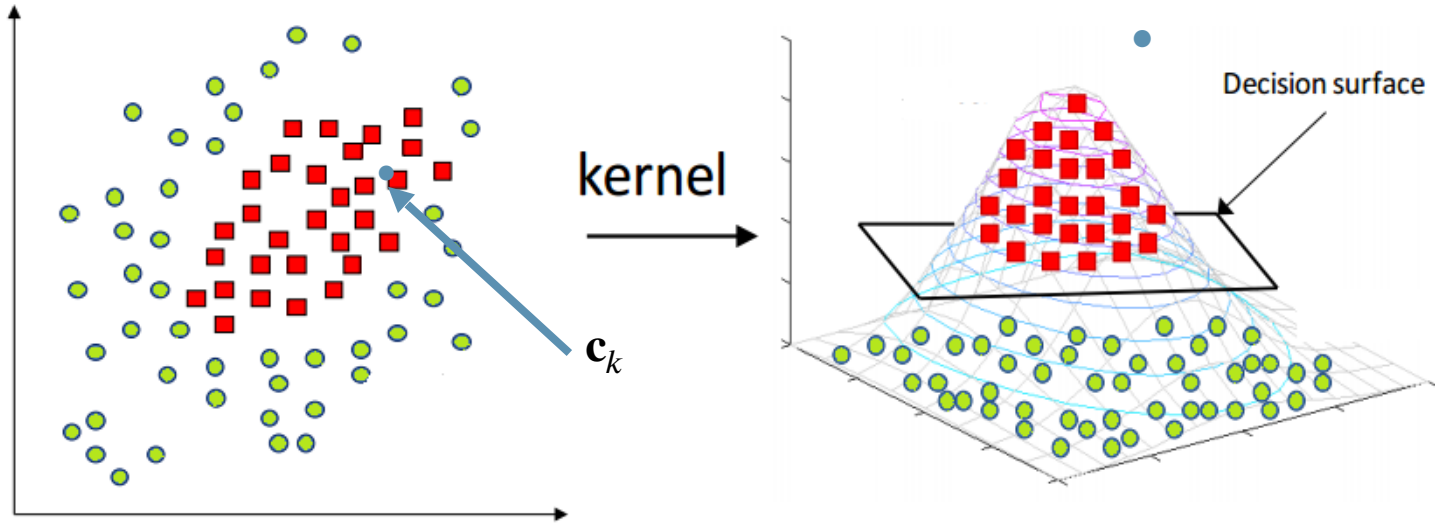
# Polynomial basis functions




- Consider the univariate example where $f(x, \mathbf{w}) = \sum_{j=1}^{M} w_j \phi_j(x)$ and $\phi_j(x) = x^j$
- With many $M$ features, our function becomes too expressive
  - can lead to overfitting (left image): low training error yet high testing error
- How to solve this?
  - lower capacity of the polynomial function (decrease $M$)
  - increase sample size whenever possible
  - assess performance on testing/validation set

# Gaussian basis function models

- Given a set of reference points in the $m$-dimensional space $\{\mathbf{c}_1, \ldots, \mathbf{c}_m\}$
  - $\phi_k(\mathbf{x}, \mathbf{c}_k) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_k\|^2}{2 \bullet \sigma^2}\right)$ is called a Gaussian function




  - we can sample $M$ points $\mathbf{c}_k$ and set $\sigma$ below their average distance
  - result: we are mapping our $m$-space data onto a $M$-spaced data with different distributions

# Outline

- Regression as a task *versus* an approach
- Linear regression
  - multiple linear regression
  - Moore-Penrose and pseudoinverse solutions
- Non-linear regression
  - principles
  - linear basis functions
- **Bayesian regression**
  - **regularization**

# Bayesian regression

- The introduced Bayesian stance for classification can be also extend to learn a regression model
  - we replace the evaluation of class hypotheses by the parameters to be estimated

$$p(\mathbf{w} \mid D) = \frac{p(D \mid \mathbf{w}) \bullet p(\mathbf{w})}{p(D)}$$

  - **Maximum likelihood** (ML): parameters $\mathbf{w}$ that maximize the likelihood function $p(D \mid \mathbf{w})$
    - how probable the observed data set is for different hyperplanes $\mathbf{w}$
    - evaluated on the observed data $D$
  - **Maximum a posterior** (MAP): parameters $\mathbf{w}$ that maximize the posterior $p(\mathbf{w} \mid D)$
    - likelihood multiplied by the prior $p(\mathbf{w})$ knowing weights $w_j$ are expected to be $N(0,\sigma_j^2)$

$$\textit{posterior} \propto \textit{likelihood} \times \textit{prior}$$

  - $p(D)$ is just a normalisation constant which ensures that $p(\mathbf{w} \mid D)$ is a valid probability density

# Bayesian regression

- In frequentist paradigms $\mathbf{w}$ is considered as a **fixed parameter**
  - determined by some estimator and errors are observed by considering the dataset $D$
- Bayesian viewpoint: there is only a dataset $D$ and **uncertainty** is represented by the distribution $\mathbf{w}$
  - maximum likelihood (ML) is

$$\mathbf{w}_{ML} = \arg\max_{\mathbf{w}} p(D|\mathbf{w})$$

  - since **log** is a monotonically increasing function

$$\mathbf{w}_{ML} = \arg\max_{\mathbf{w}} \log(p(D|\mathbf{w}))$$

  - maximizing a posteriori (MAP) is

$$\mathbf{w}_{MAP} = \arg\max_{\mathbf{w}} \log(p(\mathbf{w}|D))$$

# Bayesian regression

- We know that likelihood function is $p(z_i \mid \mathbf{w}, \mathbf{x}_i)$ : $\mathbf{w}$ in relation with $\mathbf{x}_i$ generates the data $z_i$
- We want to compute the posterior distribution $p(\mathbf{w} \mid z_i, \mathbf{x}_i)$
- From $p(\mathbf{w}, z_i) = p(\mathbf{w} \mid z_i)p(z_i) = p(z_i \mid \mathbf{w})p(\mathbf{w})$ and $p(\mathbf{w}, z_i \mid \mathbf{x}_i) = p(\mathbf{w} \mid z_i, \mathbf{x}_i)p(z_i) = p(z_i \mid \mathbf{w}, \mathbf{x}_i)p(\mathbf{w})$, we get:

$$p(\mathbf{w} \mid z_i, \mathbf{x}_i) = \frac{p(z_i \mid \mathbf{w}, \mathbf{x}_i)p(\mathbf{w})}{p(z_i)}$$

**In other words:** the posterior fulfills $p\left(\mathbf{w} \mid \mathbf{z}, X\right) \propto p(\mathbf{z} \mid \mathbf{w}, X)p(\mathbf{w})$

- To get a more concrete view, we usually use a Gaussian setting…

# Gaussian Environment

- The parameter vector $\mathbf{w}$ is fixed but unknown
- The $n$ examples $\mathbf{x}_i$ are drown independently from the same distribution
  - generally assumed to be independent and identically distributed (iid)
- The process that generates observations (target values) can be assumed to be Gaussian distributed
  - the error $\varepsilon$ in the linear regression $z = \hat{z} + \varepsilon = \mathbf{w} \bullet \mathbf{x} + \varepsilon$ is described by a Gaussian density $\varepsilon \sim N(0,\sigma)$

- The **likelihood** for a specific example is:

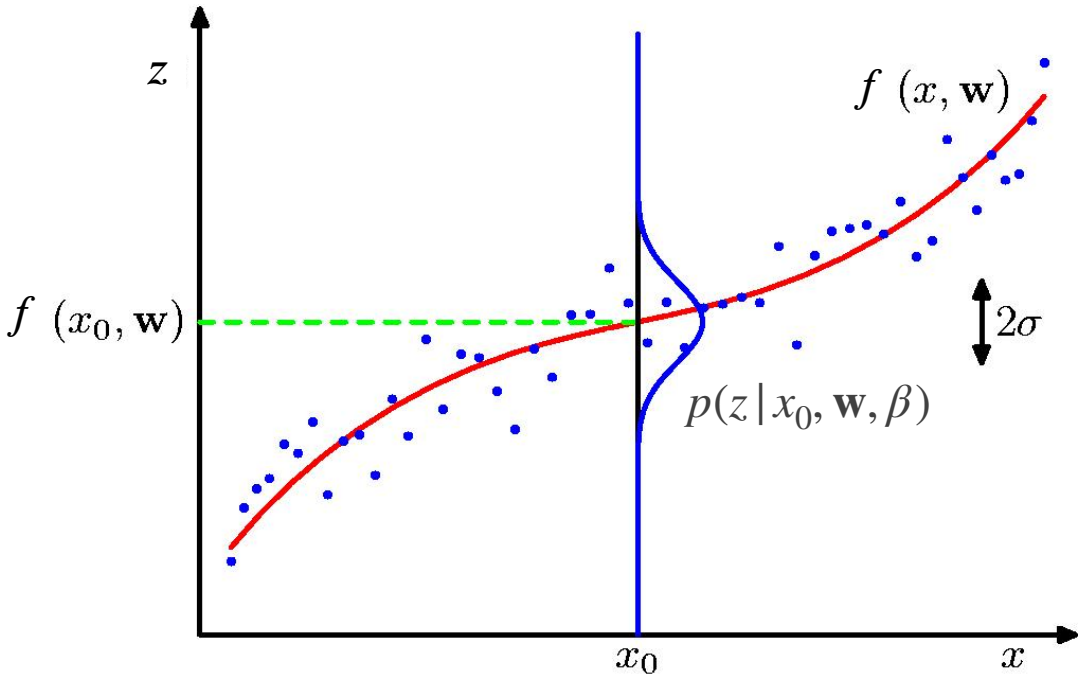
$$p(z_i \mid \mathbf{w}, \mathbf{x}_i, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\Big( -\frac{1}{2\sigma^2}(z_i - \mathbf{w}^T \cdot \mathbf{x}_i)^2 \Big)$$

- And since the samples are independent, the ``global'' likelihood is thus the product of individual likelihoods:

$$p(\mathbf{z} \mid \mathbf{w}, D, \sigma^2) = \Pi_{i=1}^{n} p(z_i \mid \mathbf{x}_i, \mathbf{w}, \sigma^2) = \frac{1}{\left(\sqrt{2\pi}\sigma\right)^n} \exp\Big( -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (z_i - \mathbf{w}^T \cdot \mathbf{x}_i)^2 \Big)$$

# Curve fitting

- Curve fitting indicates how probable the observed data $D$ is for different $\mathbf{w}$




  - precision, just an inverted variance, is often used by convention

# Prior and Posterior estimation

- To estimate the **posterior**: need to estimate the *prior* and the *likelihood*
- **Prior** estimation essentially depends on hyperplane expectations
- The $D+1$ iid components of $\mathbf{w}$ are described by a Gaussian distribution of 0 mean and variance $\sigma_w^2$.
- The prior is thus:

$$p(\mathbf{w}\,|\,\sigma_w^2) = \Pi_{i=0}^{D} p(w_i\,|\,\sigma_w^2) = \Pi_{i=0}^{D} \mathcal{N}(\mathbf{w}\,|\,0,\sigma_w^2) = \frac{1}{\left(\sqrt{2\pi}\sigma_w\right)^{D+1}} \Pi_{j=0}^{D} \left(\exp(-\frac{w_j^2}{2\sigma_w^2})\right)$$

$$= \frac{1}{\left(\sqrt{2\pi}\sigma_w\right)^{D+1}} \exp(-\frac{1}{2\sigma_w^2} \Sigma_{j=0}^{D} w_j^2) = \frac{1}{\left(\sqrt{2\pi}\sigma_w\right)^{D+1}} \exp(-\frac{1}{2\sigma_w^2} \mathbf{w}^T \cdot \mathbf{w}) = \frac{1}{\left(\sqrt{2\pi}\sigma_w\right)^{D+1}} \exp(-\frac{1}{2\sigma_w^2} \|\mathbf{w}\|^2)$$

- The **likelihood** was obtained by: $p(\mathbf{z}\,|\,\mathbf{w}, D, \sigma^2) = \Pi_{i=1}^{n} p(z_i\,|\,\mathbf{x}_i, \mathbf{w}, \sigma^2) = \frac{1}{\left(\sqrt{2\pi}\sigma\right)^{n}} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (z_i - \mathbf{w}^T \cdot \mathbf{x}_i)^2\right)$

# Posterior estimation

- The Posterior is then: $p(\mathbf{w} \mid X, \mathbf{z}, \sigma^2) \propto p(\mathbf{z} \mid X, \mathbf{w}, \sigma^2) \bullet p(\mathbf{w} \mid \sigma^2)$
- Simplifying we get $p(\mathbf{w} \mid X, \mathbf{z}, \sigma^2) \propto \exp\Big( -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (z_i - \mathbf{w}^T \cdot \mathbf{x}_i)^2 - \frac{1}{2\sigma_w^2} \|\mathbf{w}\|^2 \Big)$
- Letting $\lambda = \sigma^2 / \sigma_w^2$ and having the posterior, we can find a weight tuple $\mathbf{w}$ that maximizes it :

$$\mathbf{w}_{MAP} = arg \max_{\mathbf{w}} (p(\mathbf{w} \mid X, \mathbf{z}, \lambda))$$

- This is the same as maximizing the log posterior

$$\mathbf{w}_{MAP} = arg \max_{\mathbf{w}} \log(p(\mathbf{w} \mid X, \mathbf{z}, \lambda)) = arg \max_{\mathbf{w}} \Big( -\frac{1}{2} \sum_{i=1}^{n} (z_i - \mathbf{w}^T \cdot \mathbf{x}_i)^2 - \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \Big)$$

# Relation between regularized least squares & MAP

- The **ordinary least-squares estimator** is the weight vector that minimizes the quadratic loss

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(z_i - \mathbf{w}^T \cdot \mathbf{x}_i)^2$$

- To overcome overfitting we can add a **regularization term** (constraining the model's freedom)

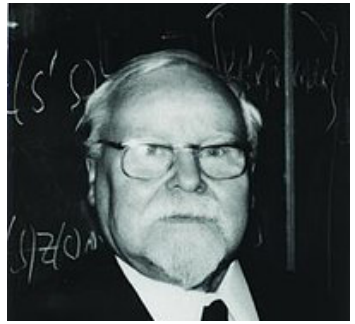
$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(z_i - \mathbf{w}^T \cdot \mathbf{x}_i)^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

**NB:** Usually we simplify $\|\mathbf{w}\| = \|\mathbf{w}\|_2$ (Euclidean norm)

**Hence** the **regularized least-squares estimator** is in fact identical to our MAP estimate!

**NB2:** This term forces the weights $\mathbf{w}$ top be as small as possible

# Regularized least squares



- **Ridge regression model:** regressor with the introduced quadratic regularization term, also called **Tikhonov regularization**
- Minimizing the quadratic function is equivalent to maximizing $\mathbf{w}_{MAP}$

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(z_i - \mathbf{w}^T \cdot \mathbf{x}_i)^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

  - we set the gradient of $E(\mathbf{w})$ to zero with the gradient operator

$$\nabla E(\mathbf{w}) = \nabla\left(\frac{1}{2} \bullet (\mathbf{z} - X \bullet \mathbf{w})^T(\mathbf{z} - X \bullet \mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}\right) = 0$$

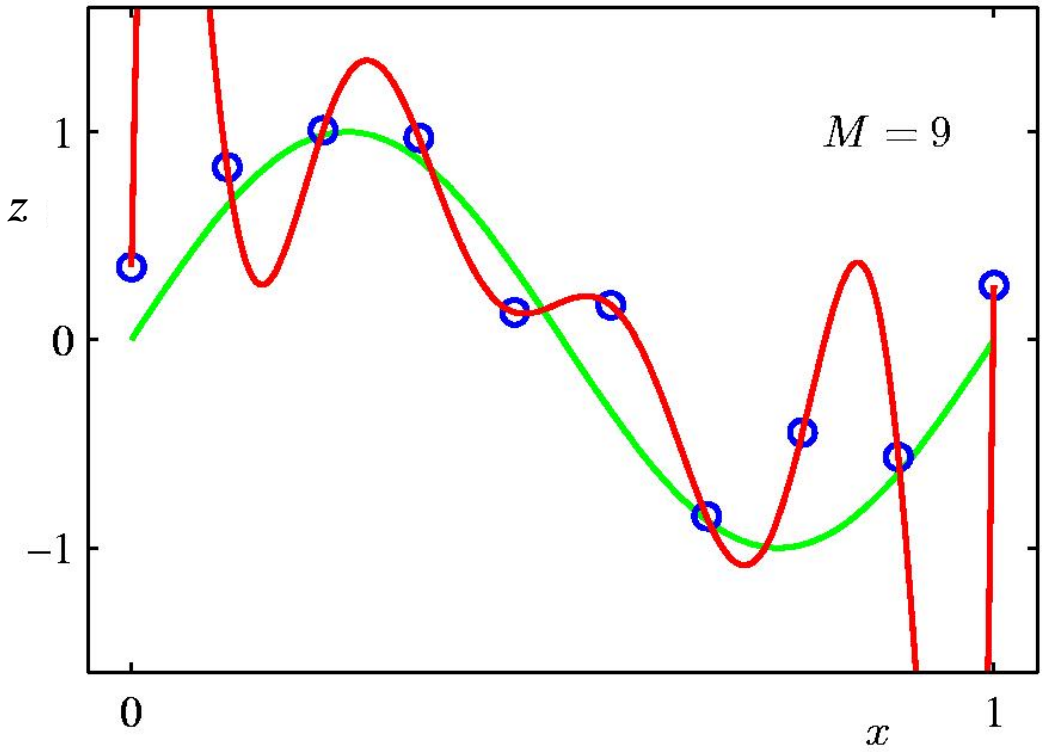
$$-2X^T\mathbf{z} + 2X^T \bullet X \bullet \mathbf{w} + 2\lambda \bullet \mathbf{w} = 0$$

$$X^T\mathbf{z} = (X^T \bullet X + \lambda \bullet I) \bullet \mathbf{w}$$

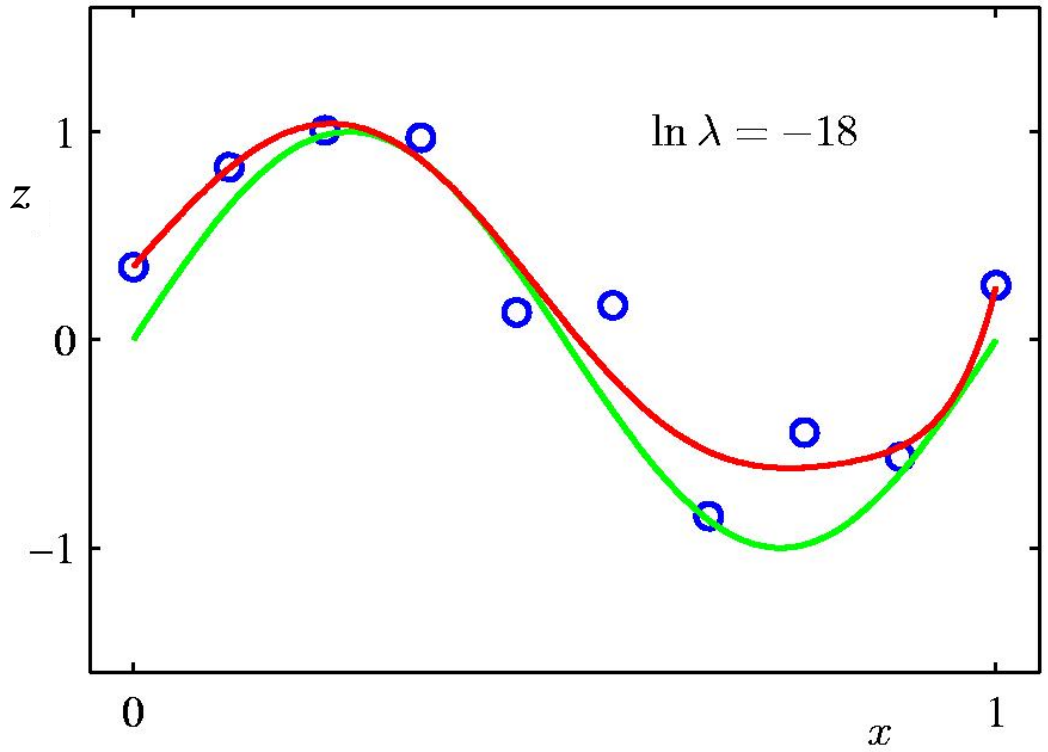
$$\left(X^T \bullet X + \lambda \bullet I\right)^{-1} \bullet X^T \bullet \mathbf{z} = \mathbf{w}$$

# Polynomial fitting and regularization

Without regularization




With regularization (Ridge)



# Lasso regularization

- An alternative to the quadratic regularization term is a linear term:
  - *Lasso* (least absolute shrinkage and selection operator) term

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(z_i - \mathbf{w}^T \cdot \mathbf{x}_i)^2 + \frac{\lambda}{2}\|\mathbf{w}\|_1$$

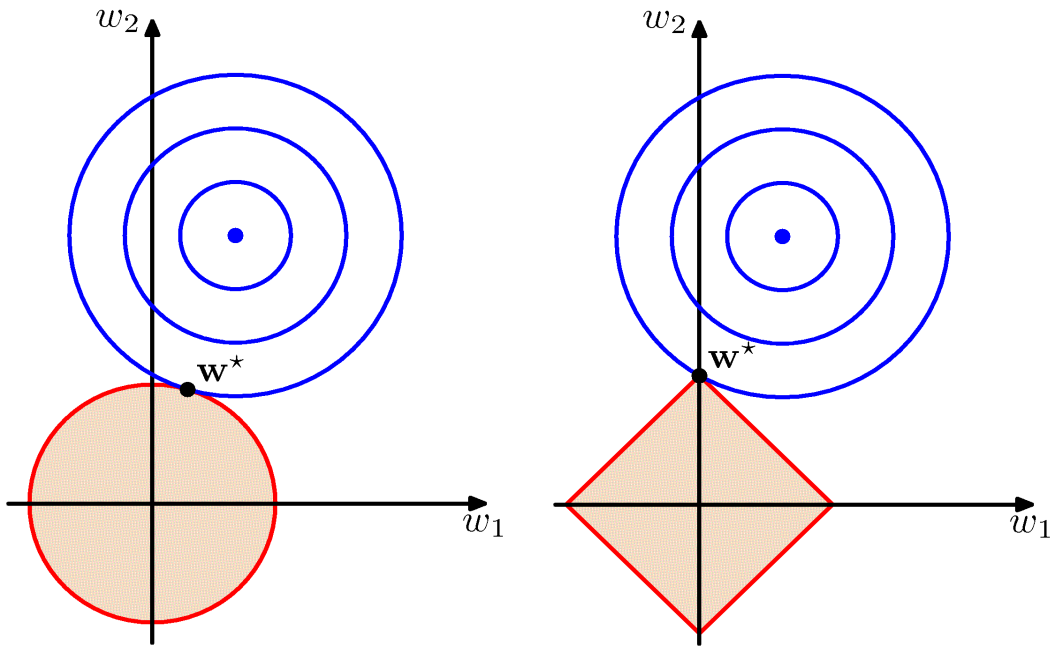
  - similarly, for large $\lambda$, some $w_j$ are driven to zero, leading to a sparse model
  - Bayesian interpretation: MAP estimates assume prior distribution is Laplacian (instead of Gaussian)

$$p(\mathbf{w}\,|\,b) = \left(\frac{1}{2b}\right)^{D+1}\Pi_{i=0}^{D}\exp\left(\frac{-|\,w_i\,|}{b}\right) = \left(\frac{1}{2b}\right)^{D+1}\exp\left(\frac{-\|\mathbf{w}\|_1}{b}\right)$$

Where the scale parameter $b > 0$ is usually referred to as the **diversity parameter.**

# Regularized least squares

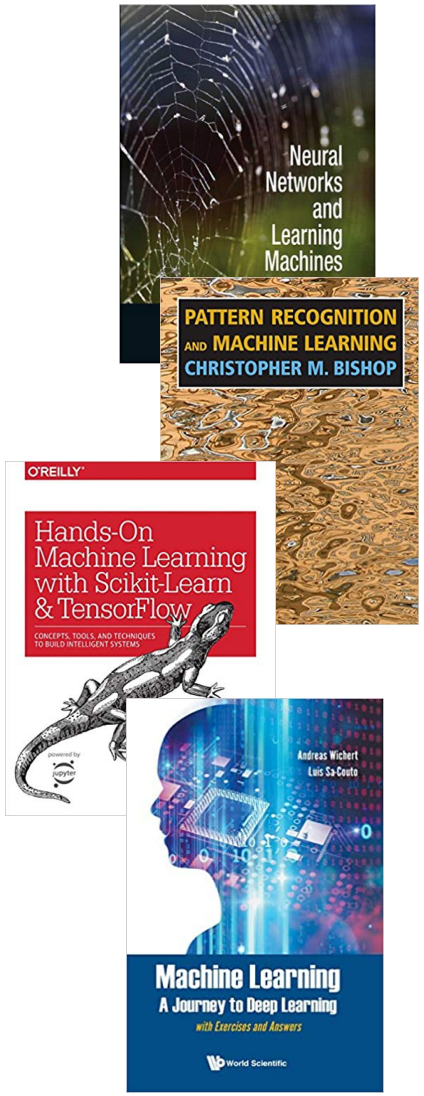
- Lasso tends to generate sparser solutions than a quadratic regularizer
- Why regularization is relevant to avoid overfitting risks?
  – particularly relevant in high dimensional data spaces



# Outline

- **Regression as a task** *versus* **an approach**
- **Linear regression**
  - multiple linear regression
  - Moore-Penrose and pseudoinverse solutions
- **Non-linear regression**
  - principles
  - linear basis functions
- **Bayesian regression**
  - regularization

# Literature




S. Haykin, Neural Networks and Learning Machine (3rd Ed), Pearson 2008

– Chapter 2

C. Bishop, Pattern Recognition and Machine Learning, Springer 2006

– Sections *1.1, 1.2.3, 1.2.4, 1.2.5, 1.2.6,* Chapter *3* till Section *3.3.3*

A. Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow, O'Reilly, 2017

– Chapter 4

A. Wichert, L. Sá-Couto, Machine Learning: A Journey to Deep Learning, World Scientific, 2021

– Chapter 4

# Thank You


miguel.j.couceiro@tecnico.ulisboa.pt
andreas.wichert@tecnico.ulisboa.pt