Linear Least Squares: Applications EE-312

Prof. Pierre Vandergheynst

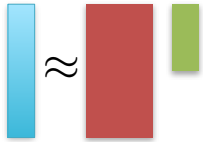
A General View

Solve: $y = Ax,\ A \in \mathbb{R}^{m \times n}$

Determined (m=n):

Over-Determined (m>n): $\arg\min_{x \in \mathbb{R}^n} \|Ax - y\|_2$

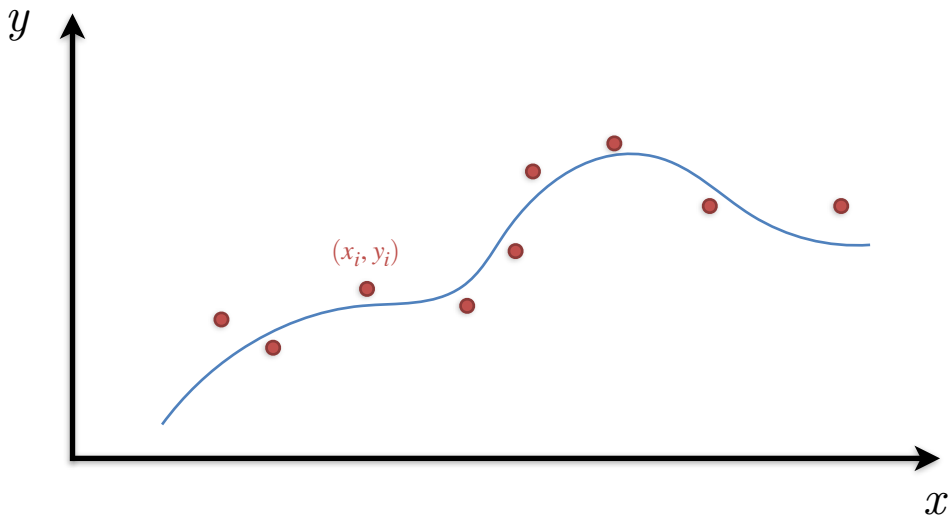
$$x = \left(A^T A\right)^{-1} A^T y = A^+ y$$



Under-Determined (m<n): $\arg\min_{x \in \mathbb{R}^n} \{\|x\|_2 \text{ such that } Ax = y\}$

$$x = A^T \left(AA^T\right)^{-1} y = A^+ y$$

Fitting Data

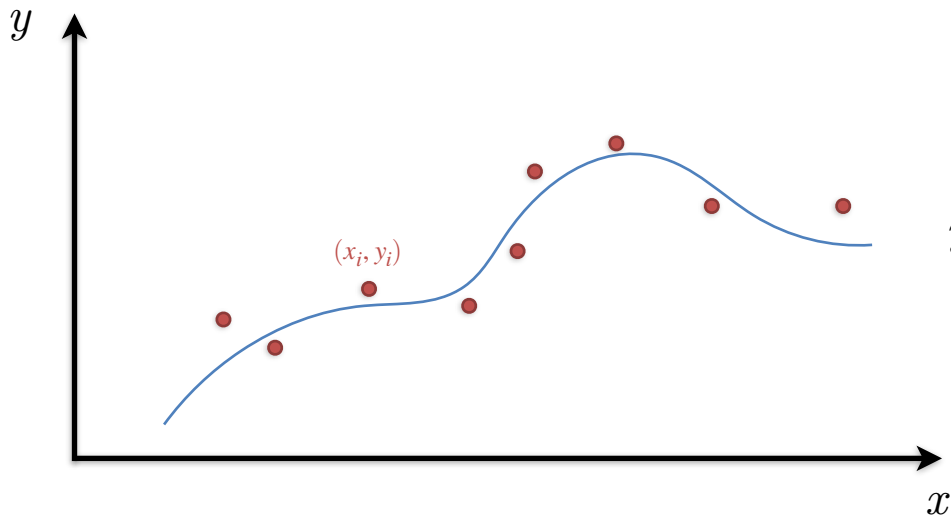


$y_i,\ i = 1, \ldots, m$ measurements taken at corresponding x_i

Hypothesis: $y = f(x; \beta) = \sum_{j=1}^{n} \beta_j \varphi_j(x)$

A linear model (in the unknown parameters β_j)

Fitting Data



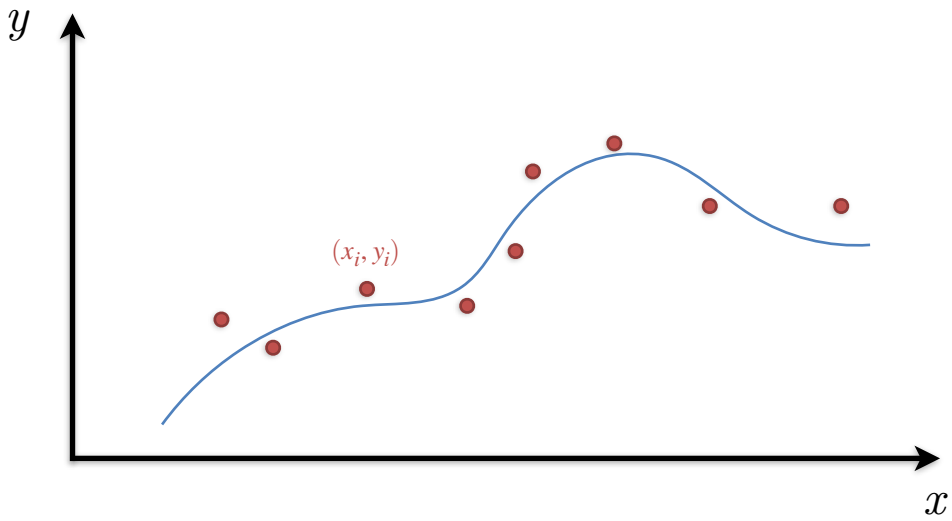
$$y = A\beta + \delta$$

It is unlikely we achieve $y_i = f(x_i;\beta)\,\forall i \Rightarrow y_i = f(x_i;\beta) + \delta_i$

Small error, noise

Set $\quad y = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} \in \mathbb{R}^m \quad A = \begin{pmatrix} \varphi_1(x_1) & \dots & \varphi_n(x_1) \\ \vdots & & \vdots \\ \varphi_1(x_m) & \dots & \varphi_n(x_m) \end{pmatrix} \in \mathbb{R}^{m\times n} \quad \beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_n \end{pmatrix} \in \mathbb{R}^n \quad \delta = \begin{pmatrix} \delta_1 \\ \vdots \\ \delta_m \end{pmatrix} \in \mathbb{R}^m$

Fitting Data



$$y = Ax + \delta$$

LS will minimize the Euclidean norm of error: $\arg\min_{x \in \mathbb{R}^n} \|Ax - y\|_2$

Assuming A is a tall, full-rank matrix (i.e rank$(A) = n$):

$$x = \left(A^T A\right)^{-1} A^T y$$

Example: Polynomial Regression

Problem: fit a polynomial of degree $< n$,

$$p(t) = \beta_0 + \beta_1 t + \ldots + \beta_{n-1} t^{n-1}$$

to data (t_i, y_i), $i = 1 \ldots m$

Basis functions: $\varphi_j(t) = t^{j-1}$

Matrix form: $A_{ij} = \varphi_j(t_i) = {t_i}^{j-1}$

$$A = \begin{bmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^{n-1} \\ 1 & t_2 & t_2^2 & \cdots & t_2^{n-1} \\ & \vdots & & & \vdots \\ 1 & t_m & t_m^2 & \cdots & t_m^{n-1} \end{bmatrix}$$

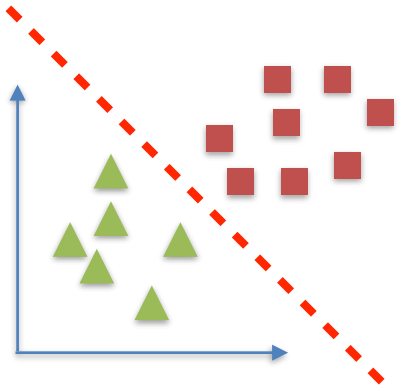
Vandermonde matrix

$t_k \neq t_\ell$ for $k \neq \ell$ and $m > n \Rightarrow A$ has full rank n

From regression to classification

Linearly separable data (2 classes):

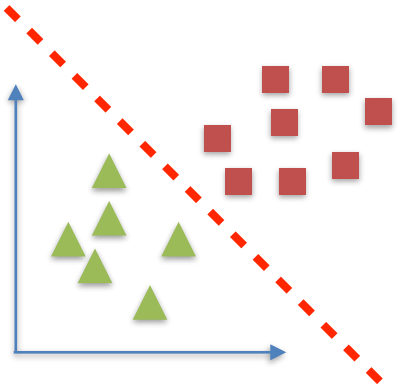
There exists a hyperplane that separates data space in two regions, each containing only one class.



Separating hyperplane or
Linear Discriminant Function

From regression to classification

Linearly separable data (2 classes):



Separating hyperplane or
Linear Discriminant Function

$y(x) = w^T x + w_0,$ data point $x \in \mathbb{R}^n$

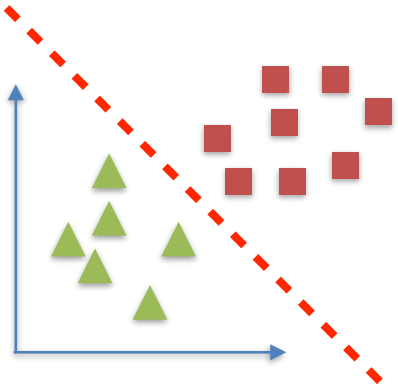
weight vector $w \in \mathbb{R}^n$

bias $w_0 \in \mathbb{R}$

Classification rule: $\text{sign}\big(y(x)\big)$

From regression to classification

Linearly separable data (2 classes):



Loss: Penalty occurring when mis-predicting $\text{sign}(y)$

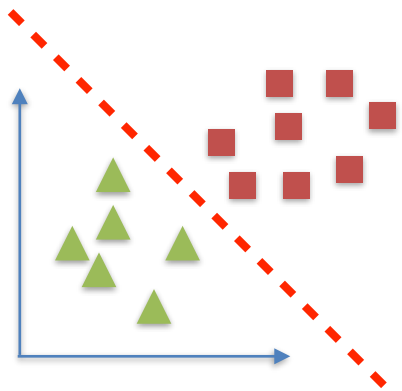
For a data point x with label $t \in \{0,1\}$

$$L_{\text{zero-one}}(y(x), t) = \begin{cases} 0 & \text{if } \text{sign}(y(x)) = t \\ 1 & \text{if } \text{sign}(y(x)) \neq t \end{cases}$$

Learning: Using labeled training examples, find the optimal weight vector that minimises the loss

From regression to classification

Linearly separable data (2 classes):



A Least Squares formulation

"Extension" trick: $\tilde{w} = (w_0, w) \in \mathbb{R}^{n+1}$

$\tilde{x} = (1, x) \in \mathbb{R}^{n+1}$

$y(x) = \tilde{w}^T \tilde{x}$

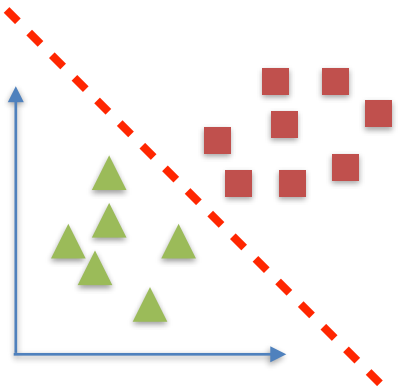
Loss: forget the binary nature of the prediction and use squared error

$$L_{\text{squared}}(y(x), t) = \big(\text{sign}(y(x)) - t\big)^2$$

$$L_{\text{squared}}(y(x), t) = \big(y(x) - t\big)^2$$

From regression to classification

Linearly separable data (2 classes):



A Least Squares formulation

"Extension" trick: $\tilde{w} = (w_0, w) \in \mathbb{R}^{n+1}$

$\tilde{x} = (1, x) \in \mathbb{R}^{n+1}$

$y(x) = \tilde{w}^T \tilde{x}$

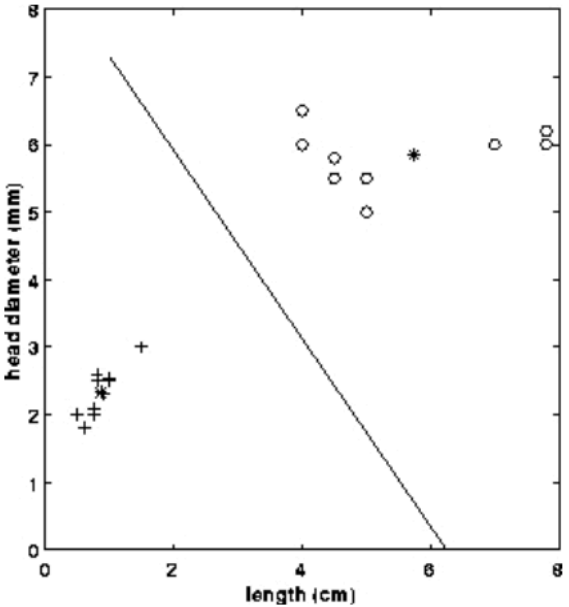
m training samples data matrix $\tilde{X} \in \mathbb{R}^{(n+1)\times m}$

$$L_{\text{train}} = \sum_{i=1}^{m} (y_i - \tilde{w}^T \tilde{x}_i)$$

$$L_{\text{train}} = \|\tilde{X}^T \tilde{w} - y\|_2^2$$

From regression to classification

Linearly separable data (2 classes):



A Least Squares formulation

"Extension" trick: $\tilde{w} = (w_0, w) \in \mathbb{R}^{n+1}$

$\tilde{x} = (1, x) \in \mathbb{R}^{n+1}$

$y(x) = \tilde{w}^T \tilde{x}$

m training samples data matrix $\tilde{X} \in \mathbb{R}^{(n+1)\times m}$

$$L_{\text{train}} = \sum_{i=1}^{m} (y_i - \tilde{w}^T \tilde{x}_i)$$

$$L_{\text{train}} = \|\tilde{X}^T \tilde{w} - y\|_2^2$$

System Identification

Basic problem: you observe the input $u(t)$ and the corresponding output $y(t)$ of an unknown linear system H. Our goal is to identify a good model for the system's action: $y(t) = H\,u(t)$, where H is linear

Example: moving average model

$$y(t) = h_0 u(t) + h_1 u(t-1) + \ldots + h_n u(t-n), \text{ where } h_0, \ldots h_n \in \mathbb{R}$$

System Identification

$$y(t) = h_0 u(t) + h_1 u(t-1) + \ldots + h_n u(t-n), \text{ where } h_0, \ldots h_n \in \mathbb{R}$$

Matrix form:

$$\begin{pmatrix} y(n) \\ y(n+1) \\ \vdots \\ y(n+m) \end{pmatrix} = \begin{pmatrix} u(n) & u(n-1) & \ldots & u(0) \\ u(n+1) & u(n) & \ldots & u(1) \\ \vdots & \vdots & & \vdots \\ u(n+m) & u(n+m-1) & \ldots & u(m) \end{pmatrix} \begin{pmatrix} h_0 \\ h_1 \\ \vdots \\ h_n \end{pmatrix}$$

System Identification

Example:

Prediction / Forecasting / Modelling

Goal: given n past observations x of a time series, predict the next sample

Example: the autoregressive AR(n) model

$$x(N) = c + h_1 x(N-1) + h_2 x(N-2) + \ldots + h_n x(N-n) + \delta_N$$

Example:

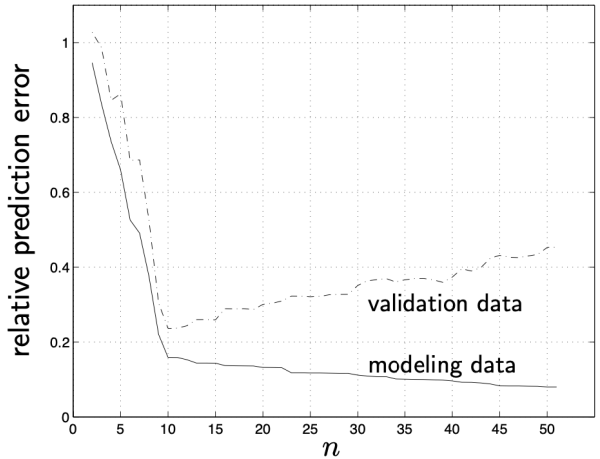
Note: Model order selection

Think of linear regression with polynomials

The larger the order of the model, the lower the error on the data used for training

BUT prediction/modeling power on unseen data becomes worse

Solution: Cross-Validation - evaluate performance of model order
on data not used for training



Streaming Measurements and Recursive Least Squares

If we view the Least Squares in row form:

$$\text{minimize} \left\{ \|Ax - y\|_2^2 = \sum_{i=1}^{m} (a_i^T x - y_i)^2 \right\}$$

m linear observations of the unknown x

Each row performs a measurement $a_i^T x \simeq y_i$

Find the x that best explains the measurements

And the solution is written (in terms of rows) as:

$$x = \left(\sum_{i=1}^{m} a_i a_i^T \right)^{-1} \sum_{i=1}^{m} y_i a_i$$

sum of measurements vectors a_i,

weighted by observations y_i

Streaming Measurements and Recursive Least Squares

Now observations come in stream, so we keep adding information about x

$$x(m) = \left(\sum_{i=1}^{m} a_i a_i^T\right)^{-1} \sum_{i=1}^{m} y_i a_i$$

Our goal is to recursively compute the solution as measurements come

$$x(m) = P(m)^{-1} q(m)$$

Easy !

$$P(m+1) = P(m) + a_{m+1} a_{m+1}^T \qquad q(m+1) = q(m) + y_{m+1} a_{m+1}$$

Challenge: compute the inverse of this rank-1 update efficiently

Streaming Measurements and Recursive Least Squares

$$x(m) = P(m)^{-1} q(m)$$

$$P(m+1) = P(m) + a_{m+1} a_{m+1}^T \qquad q(m+1) = q(m) + y_{m+1} a_{m+1}$$

Remark: as $P(m)$ becomes invertible, it will remain so when we add rows

We can compute $P(m+1)^{-1}$ using the Sherman-Morrison formula:

$$(P + aa^T)^{-1} = P^{-1} - \frac{1}{1 + a^T P^{-1} a} (P^{-1} a)(P^{-1} a)^T$$

P non-singular and symmetric

Cost $= \mathcal{O}(n^2)$ instead of $\mathcal{O}(n^3)$