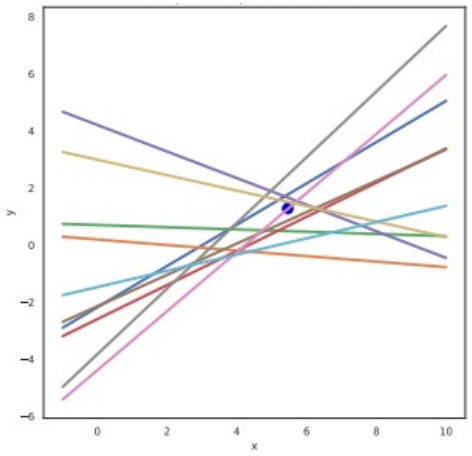
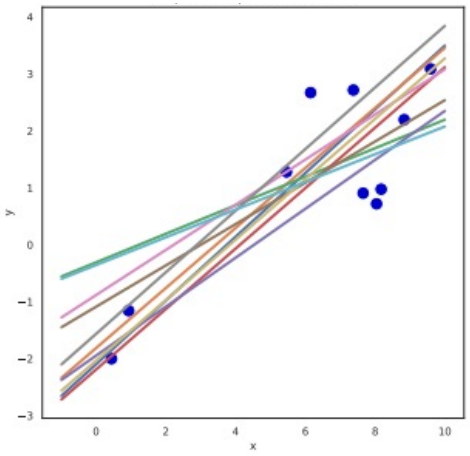
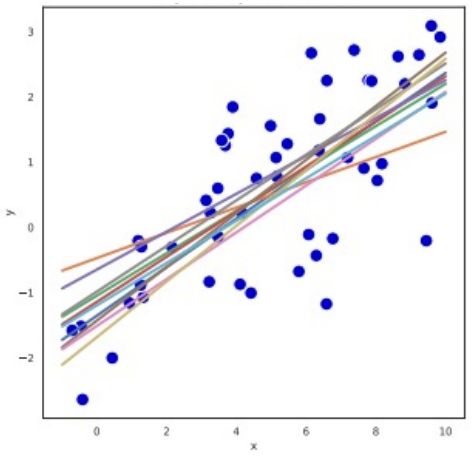
Machine Learning

Linear Regression

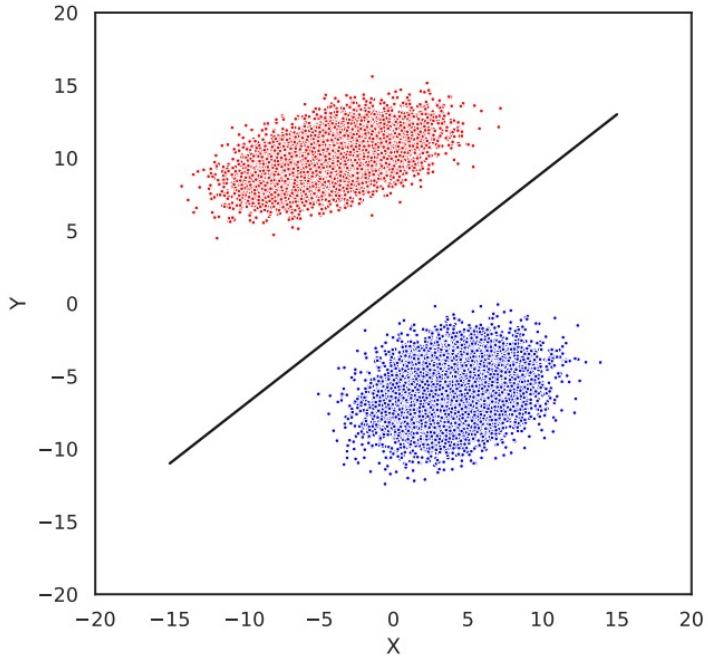
Jian Liu



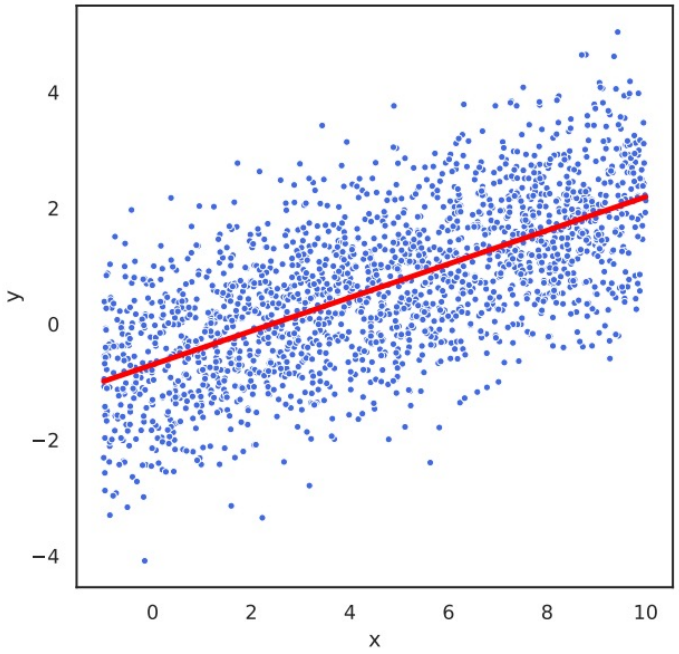




Classification vs Regression



(a) Classification



(b) Regression

Linear Regression

- Assumes a linear functional relationship between the predictor and target variables: $y = f(x, \omega) + \epsilon$
- A learned regression model can be used to *predict* y for new values of x
- Linear and non-linear *basis functions* can be used to construct a linear regression function
- So why is it called linear regression?

  Ex: $y = \omega_0 + \omega_1 x + \omega_2 x^2 + \omega_3 x^3 + \epsilon$

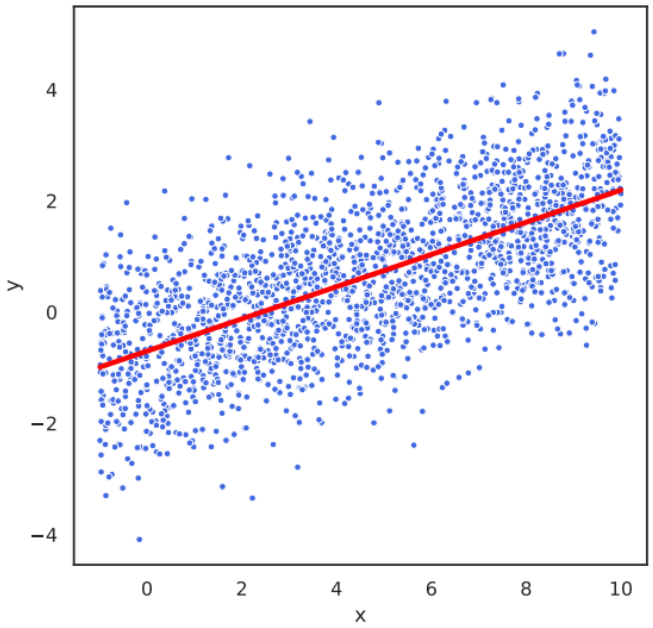


Figure: Linear regression: polynomial of degree 1

Linear Regression

- Let's say we have training data comprising $\{x_i, y_i\}_{i=1..N}$ pairs
- x_i: BMI & y_i: cardiac outputs (ejection fraction)
- We want to fit a polynomial of degree 1 to the data such that

$$\hat{y}_i = \omega_0 + \omega_1 x_i \tag{1}$$

- Where, $\hat{y}_i$ is the *predicted* value for y_i
- ω_0: intercept and ω_1: slope
- So how do we estimate ω_0, ω_1?

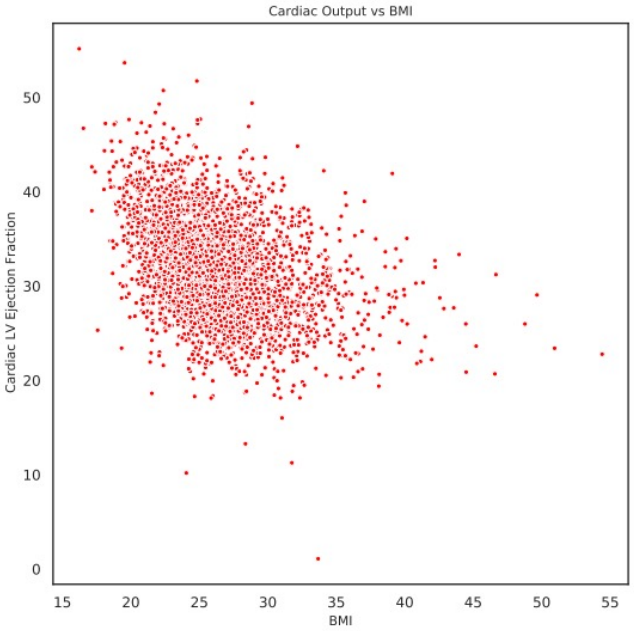


Figure: Cardiac output vs BMI

Ordinary Least Squares (OLS)

- First define the residual for a single data point i as:

$$r_i = (y_i - (\omega_0 + \omega_1 x_i)) = (y_i - \hat{y}_i) \quad (2)$$

- Key idea in OLS - find ω_0, ω_1 that minimises the sum of squared residuals:

$$R(\boldsymbol{\omega}) = \sum_{i=1}^{N} r_i^2 = \sum_{i=1}^{N} (y_i - \hat{y}_i(x_i, \boldsymbol{\omega}))^2 \quad (3)$$

- So how do we minimise $R(\boldsymbol{\omega})$?

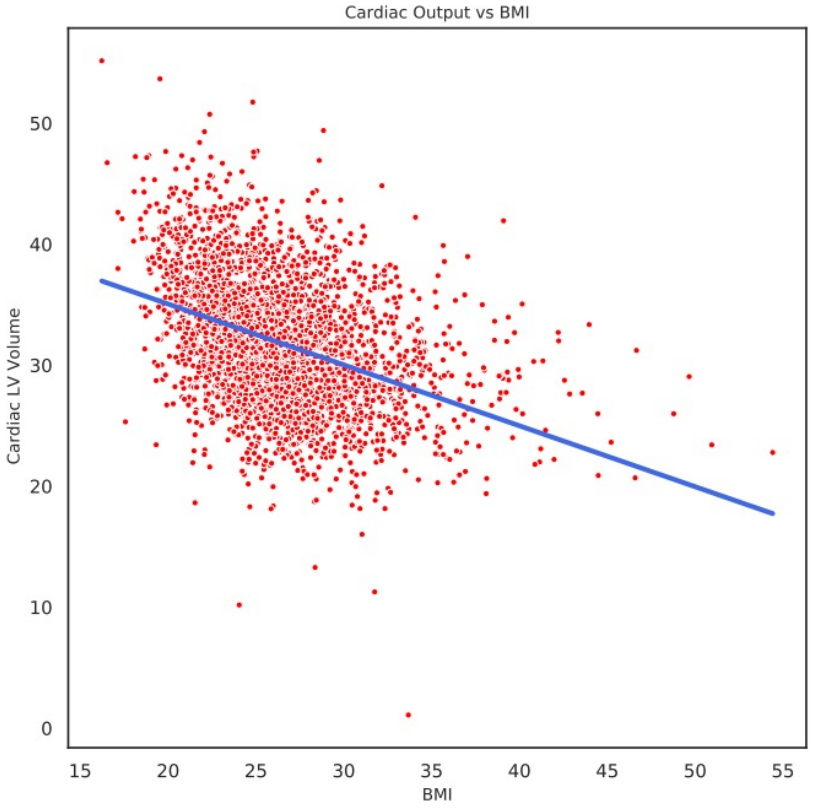


Figure: Cardiac output vs BMI

Ordinary Least Squares (OLS)

- Setting derivates of R w.r.t $\boldsymbol{\omega}$ to 0:

$$\frac{\partial R(\omega_0, \omega_1)}{\partial \omega_0} = 0 \tag{4a}$$

$$\frac{\partial R(\omega_0, \omega_1)}{\partial \omega_1} = 0 \tag{4b}$$

- This results in a linear system of two equations with two unknowns (ω_0, ω_1):

$$\sum_{i=1}^{N} x_i y_i = \omega_0 \sum_{i=1}^{N} x_i + \omega_1 \sum_{i=1}^{N} x_i^2 \tag{5a}$$

$$\sum_{i=1}^{N} y_i = \omega_0 N + \omega_1 \sum_{i=1}^{N} x_i \tag{5b}$$

Ordinary Least Squares (OLS)

- Expressing the system of equations in matrix form:

$$\begin{pmatrix} \sum_i^N x_i y_i \\ \sum_i^N y_i \end{pmatrix} = \begin{pmatrix} \sum_i^N x_i & \sum_i^N x_i^2 \\ N & \sum_i^N x_i \end{pmatrix} \begin{pmatrix} \omega_0 \\ \omega_1 \end{pmatrix} \tag{6a}$$

$$\begin{pmatrix} \omega_0 \\ \omega_1 \end{pmatrix} = \begin{pmatrix} \sum_i^N x_i & \sum_i^N x_i^2 \\ N & \sum_i^N x_i \end{pmatrix}^{-1} \begin{pmatrix} \sum_i^N x_i y_i \\ \sum_i^N y_i \end{pmatrix} \tag{6b}$$

- Equation 6b is the OLS solution for ω_0, ω_1 for a degree$=1$ polynomial regression function

OLS: Design Matrix

- So how can we generalise this to arbitrary degree polynomials?
- Lets redefine our regression function as:

$$\hat{y}_i(x_i, \boldsymbol{\omega}) = \sum_{j=0}^{M} \omega_j \phi_j(x_i) = \boldsymbol{\omega}^T \boldsymbol{\phi}(x_i) \tag{7}$$

- For a degree=2 polynomial we have: $\hat{y}_i(x_i, \boldsymbol{\omega}) = \omega_0 + \omega_1 x_i + \omega_2 x_i^2$

$$\boldsymbol{\omega} = \begin{pmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \end{pmatrix}, \qquad \boldsymbol{\phi}(x_i) = \begin{pmatrix} \phi_0(x_i) \\ \phi_1(x_i) \\ \phi_2(x_i) \end{pmatrix} = \begin{pmatrix} 1 \\ x_i \\ x_i^2 \end{pmatrix} \tag{8}$$

OLS: Design Matrix

- Similarly for polynomials of arbitrary degree we have:

$$\hat{y}_i(x_i, \boldsymbol{\omega}) = \sum_{j=0}^{M} \omega_j \phi_j(x_i) = \boldsymbol{\omega}^T \boldsymbol{\phi}(x_i) \tag{9a}$$

$$\boldsymbol{\omega} = \begin{pmatrix} \omega_0 \\ \omega_1 \\ \vdots \\ \omega_M \end{pmatrix}, \qquad \boldsymbol{\phi} = \begin{pmatrix} \phi_0(x_i) \\ \phi_1(x_i) \\ \vdots \\ \phi_M(x_i) \end{pmatrix} \tag{9b}$$

- As before $\boldsymbol{\omega}$ is the weights vector; $\boldsymbol{\phi}$ is the *basis functions* vector
- Each $\phi_j(x_i)$ is a basis function ; $\phi_0(x_i) = 1$ is a dummy basis function for ω_0 (intercept)
- This form is generalisable to other types of basis functions as well

OLS: Design Matrix

- Fitting an M-degree polynomial requires estimation of $M+1$ parameters in $\boldsymbol{\omega}$
- Using $\boldsymbol{\omega}, \phi$ we can define $R(\boldsymbol{\omega})$ as: $R(\boldsymbol{\omega}) = \sum_{i=1}^{N}(y_i - \boldsymbol{\omega}^T\phi(x_i))^2$
- As before we can estimate $\boldsymbol{\omega}_{\text{OLS}}$ by minimising $R(\boldsymbol{\omega})$ w.r.t $\boldsymbol{\omega}$; $\frac{\partial R(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} = 0$

$$\frac{\partial R(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} = \sum_{i=1}^{N}(y_i - \boldsymbol{\omega}^T\phi(\boldsymbol{x}_i))\phi(\boldsymbol{x}_i)^T = 0, \tag{10a}$$

$$\sum_{i=1}^{N} y_i\phi^T(x_i) = \boldsymbol{\omega}^T(\sum_{i=1}^{N}\phi(x_i)\phi^T(x_i)) \tag{10b}$$

- Equation 10b can be expressed in matrix-vector form by defining the *design matrix* $\boldsymbol{\Phi}$
- $\boldsymbol{\Phi}$ is an $N \times (M+1)$ matrix where each x_i has an associated basis function vector $\phi(x_i)$

OLS: Estimating regression weights

- By defining design matrix $\mathbf{\Phi} = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_M(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_M(x_2) \\ \vdots & \vdots & & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_M(x_N) \end{pmatrix}$
- ω_{OLS} is given by solving: $\mathbf{\Phi}^T\mathbf{y} = \mathbf{\Phi}^T\mathbf{\Phi}\omega$ **the normal equation**

$$\omega_{OLS} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{y} \tag{11}$$

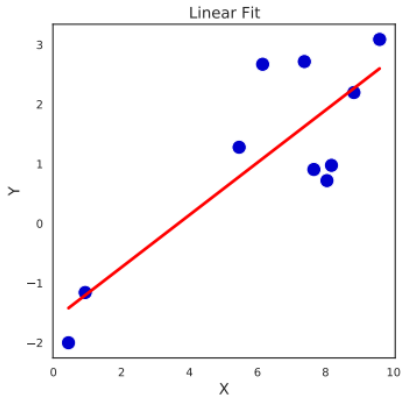
- $(\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T \equiv \mathbf{\Phi}^\dagger$ is also known as the *Moore-Penrose pseudoinverse* of $\mathbf{\Phi}$

The solution can be obtained by this set of linear equations (the normal equation).

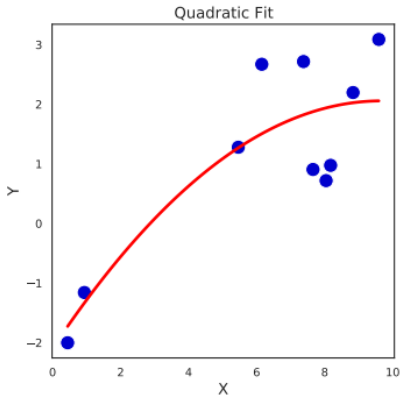
However, numerical inversion of matrices can be troublesome, especially if the matrix is large.

Regression Examples

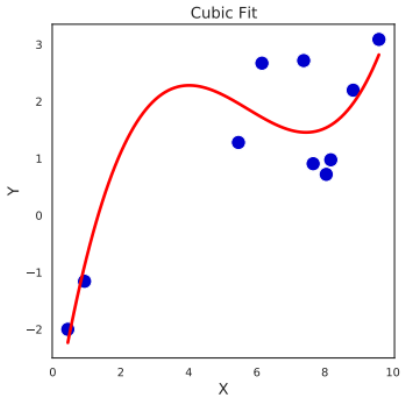
- Given some training data $\{x_i, y_i\}_{i=1..N}$ with limited number of samples ($N = 10$):



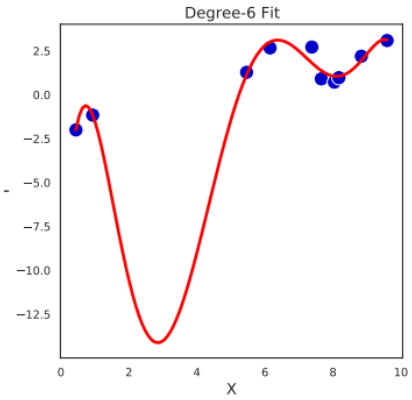
(a) Polynomial degree=1



(b) Polynomial degree=2



(c) Polynomial degree=3



(d) Polynomial degree=6

- So how do we decide which polynomial provides the best fit?

Evaluating Regression Models

- Common metrics for evaluating regression models:
  - Coefficient of determination or $R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$; $\bar{y}$ is the mean of the observed targets
  - Mean absolute error (MAE) $= \frac{1}{N} \sum_i^N \mid y_i - \hat{y}_i \mid$
  - Mean squared error (MSE) $= \frac{1}{N} \sum_i^N (y_i - \hat{y}_i)^2$
  - Root mean squared error (RMSE) $= \sqrt{\frac{1}{N} \sum_i^N (y_i - \hat{y}_i)^2}$
  - $\cdots$ and several others!

OLS: Estimating regression weights

- By defining design matrix $\mathbf{\Phi} = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_M(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_M(x_2) \\ \vdots & \vdots & & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_M(x_N) \end{pmatrix}$
- ω_{OLS} is given by solving: $\mathbf{\Phi}^T\mathbf{y} = \mathbf{\Phi}^T\mathbf{\Phi}\omega$ **the normal equation**

$$\omega_{OLS} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{y} \tag{11}$$

- $(\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T \equiv \mathbf{\Phi}^\dagger$ is also known as the *Moore-Penrose pseudoinverse* of $\mathbf{\Phi}$

The solution can be obtained by this set of linear equations (the normal equation).

However, numerical inversion of matrices can be troublesome, especially if the matrix is large.

OLS: Estimating regression weights

- By defining design matrix $\mathbf{\Phi} = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_M(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_M(x_2) \\ \vdots & \vdots & & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_M(x_N) \end{pmatrix}$
- ω_{OLS} is given by solving: $\mathbf{\Phi}^T\mathbf{y} = \mathbf{\Phi}^T\mathbf{\Phi}\omega$ **the normal equation**

$$\omega_{OLS} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{y} \tag{11}$$

- $(\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T \equiv \mathbf{\Phi}^\dagger$ is also known as the *Moore-Penrose pseudoinverse* of $\mathbf{\Phi}$

The solution can be obtained by this set of linear equations (the normal equation).

However, numerical inversion of matrices can be troublesome, especially if the matrix is large.

OLS may not be working well

Learning from data (many unknowns)

Formalization:

- Input: $\mathbf{x}$ (*customer application*)
- Output: y (*good/bad customer?*)
- Target function: $f : \mathcal{X} \rightarrow \mathcal{Y}$ (*ideal credit approval formula*)
- Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$ (*historical records*)

$\downarrow \quad \downarrow \quad \downarrow$

- Hypothesis: $g : \mathcal{X} \rightarrow \mathcal{Y}$ (*formula to be used*)

Learning from data (many unknowns)

- Focus on supervised learning
  - Unknown target function $y = f(\mathbf{x})$
  - Data set $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$
  - Learning algorithm picks $g \approx f$ from a hypothesis set $\mathcal{H}$

Linear Regression

Classification: Credit approval (yes/no)

Regression: Credit line (dollar amount)

Input: $\mathbf{x} =$

age	23 years
annual salary	\$30,000
years in residence	1 year
years in job	1 year
current debt	\$15,000
...	...

Linear regression output: $h(\mathbf{x}) = \sum_{i=0}^{d} w_i \, x_i = \mathbf{w}^{\mathsf{T}}\mathbf{x}$

Linear Regression

Credit officers decide on credit lines:

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$$

$y_n \in \mathbb{R}$ is the credit line for customer $\mathbf{x}_n$.

Linear regression tries to replicate that.

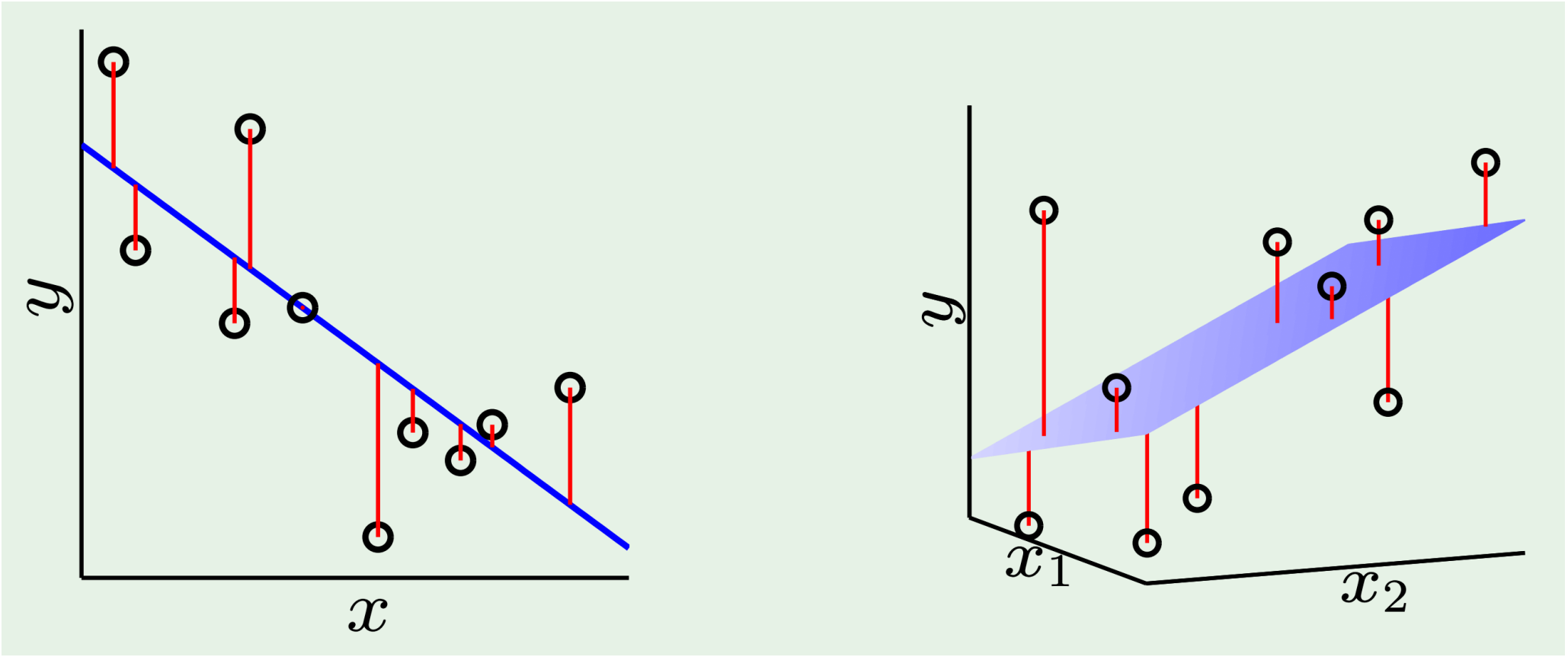
Cost function

How well does $h(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\mathbf{x}$ approximate $f(\mathbf{x})$?

In linear regression, we use squared error $(h(\mathbf{x}) - f(\mathbf{x}))^2$

$$\textbf{in-sample error: } E_{\text{in}}(h) = \frac{1}{N}\sum_{n=1}^{N}(h(\mathbf{x}_n) - y_n)^2$$

Cost function



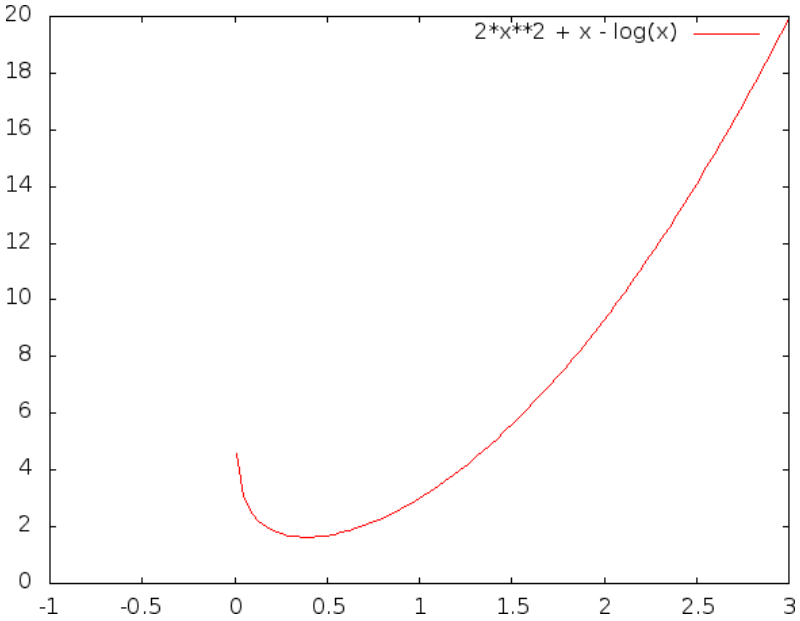
Cost function

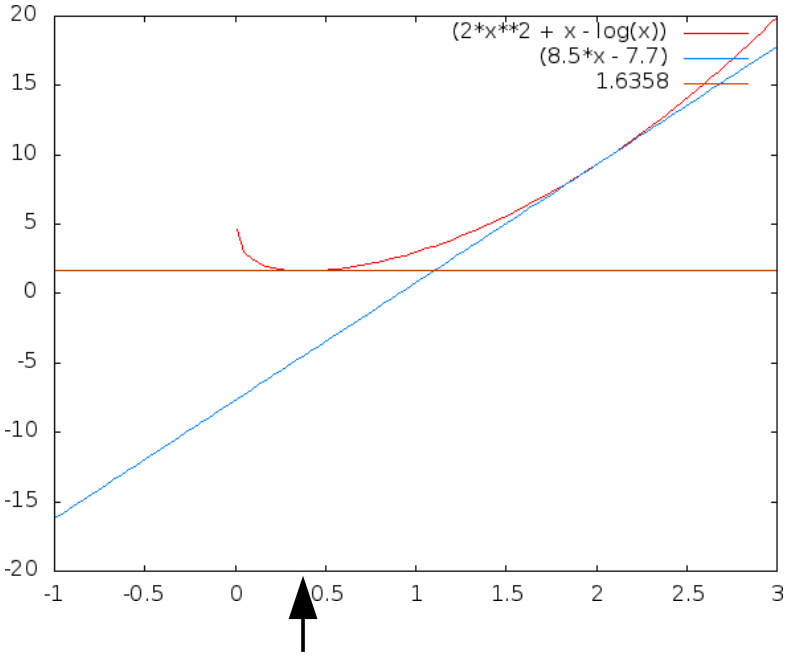
$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n - y_n\right)^2$$

$$= \frac{1}{N}\|\mathrm{X}\mathbf{w} - \mathbf{y}\|^2$$

where $\mathrm{X} = \begin{bmatrix} —\mathbf{x}_1{}^{\mathsf{T}}— \\ —\mathbf{x}_2{}^{\mathsf{T}}— \\ \vdots \\ —\mathbf{x}_N{}^{\mathsf{T}}— \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$

Minimizing the Cost function
Goal: find the minimum point

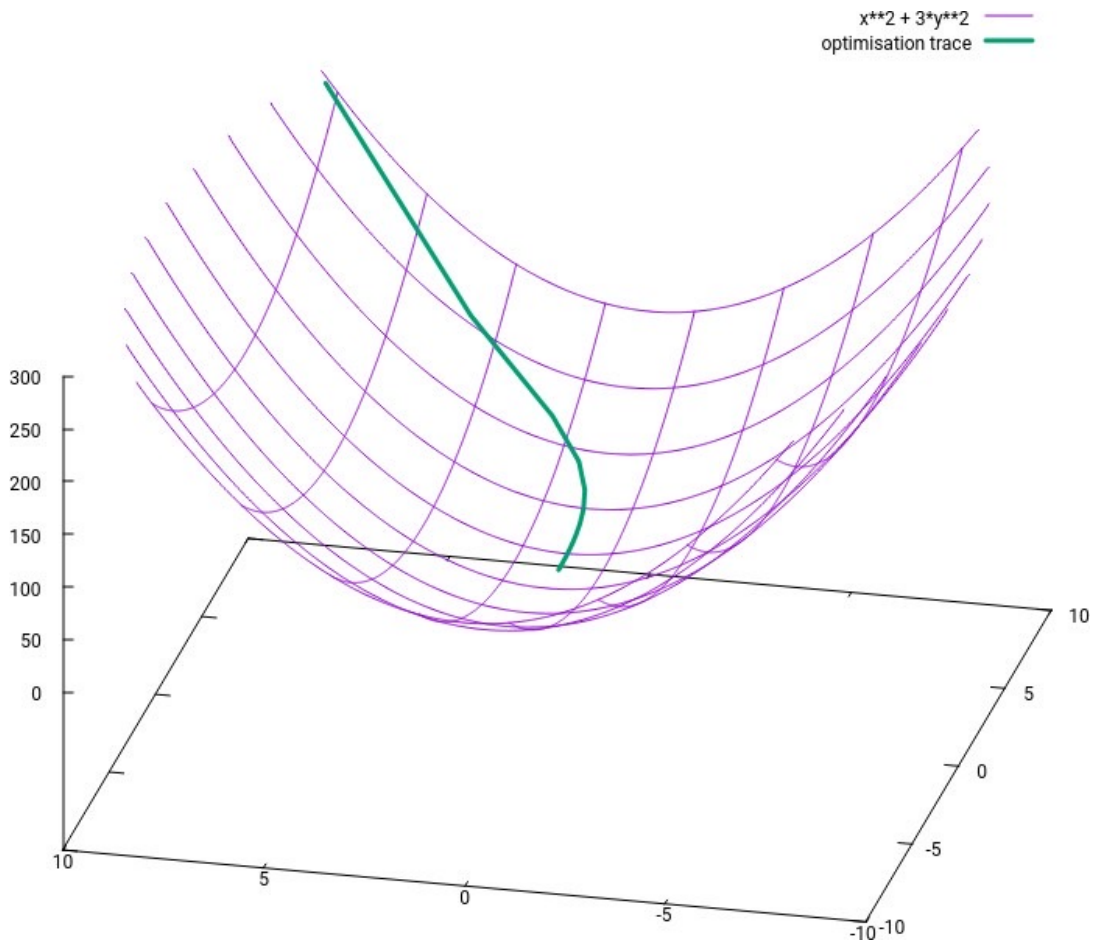




The minimum is at 0.39

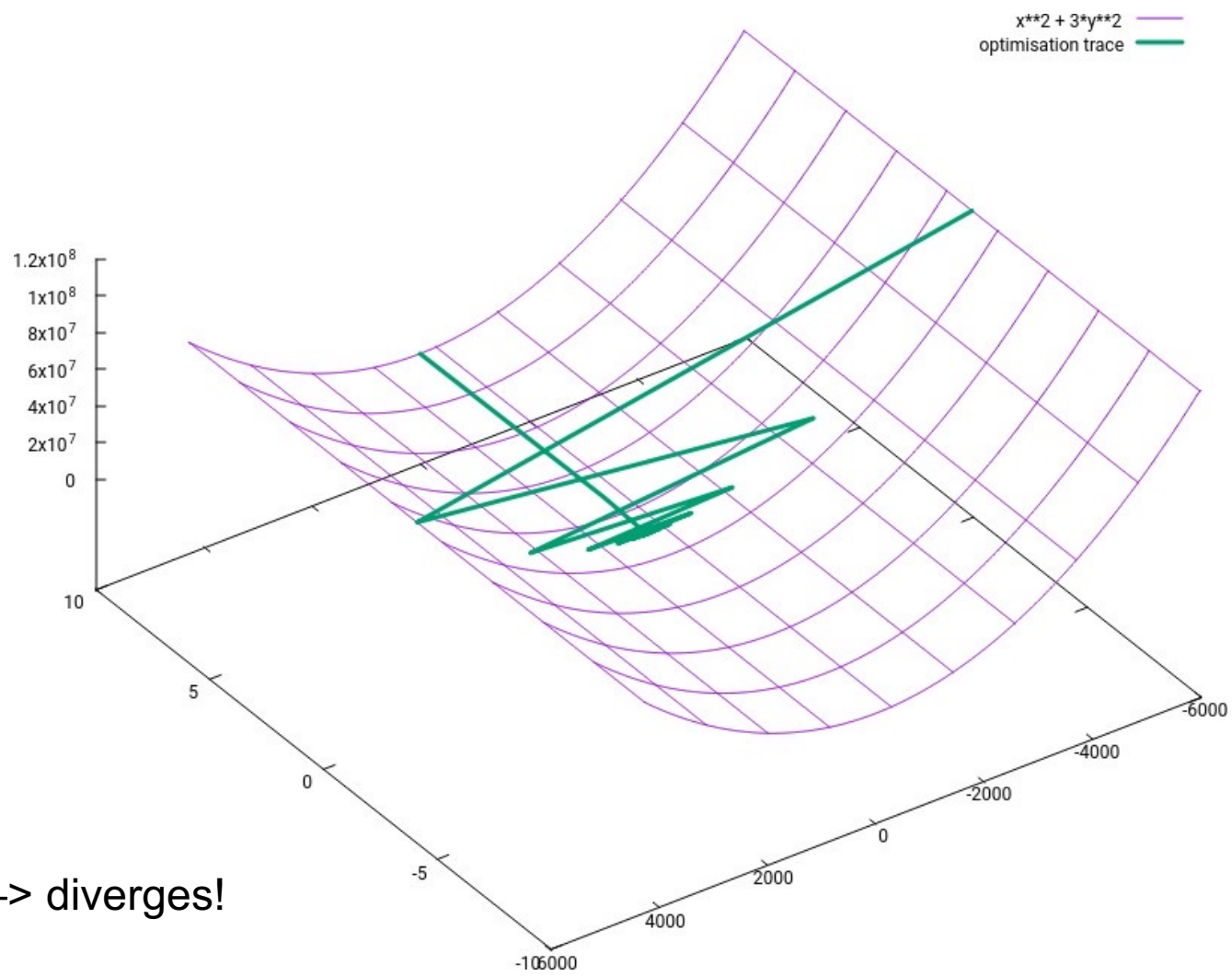
Gradient descent

Gradient descent in 3D



Step size: 0.1

Gradient descent in 3D



Step size: 0.14 –> diverges!

Linear Regression

- Using the OLS solution
- Minimising the cost function by gradient descent

Linear Regression
with linear and non-linear basis functions

Linear models with multiple M features:

The set of basis functions defines a set of components of a vector

$$\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_M(\mathbf{x}))^\top$$

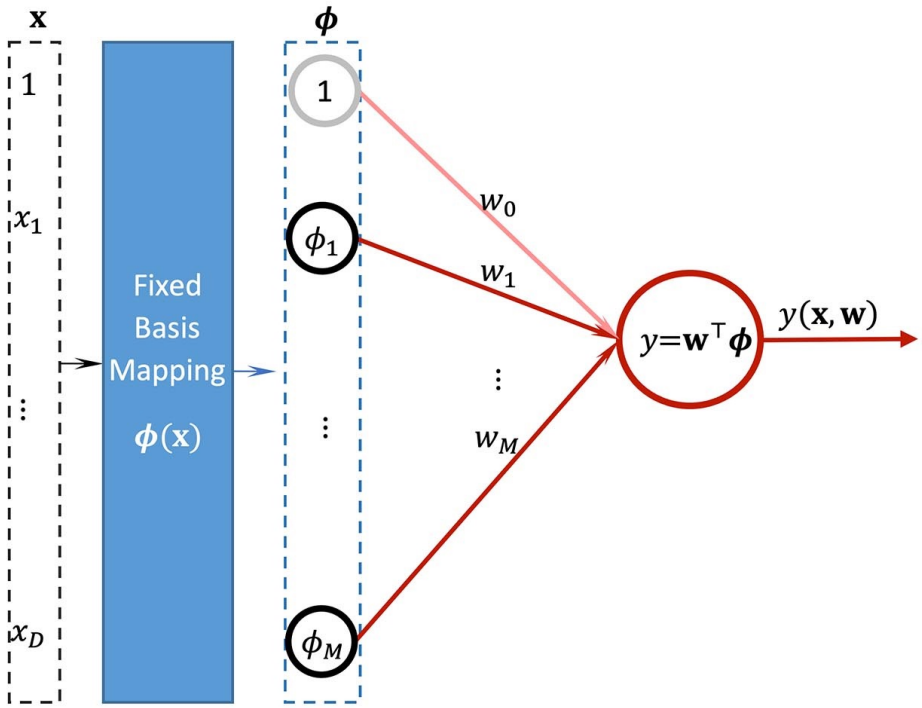
$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) + \cdots + w_M\phi_M(\mathbf{x})$$

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \boldsymbol{\phi}$$

Where $\mathbf{w} = (w_0, w_1, w_2, \ldots, w_M)$ and $\boldsymbol{\phi} = (\phi_0, \phi_1, \phi_2, \ldots, \phi_M)^{\mathrm{T}}$ and $\phi_0 = 1$.

EXAM1	EXAM2	EXAM3	FINAL
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142
53	46	55	101
69	74	77	149
47	56	60	115
87	79	90	175
79	70	88	164
69	70	73	141
70	65	74	141
93	95	91	184
79	80	73	152
70	73	78	148
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

Linear Regression
with linear and non-linear basis functions



we can use any basis function in our linear model

as long as the weights are all linear

Linear Regression
with linear and non-linear basis functions

Polynomial basis functions

$$w_0 + w_1 x_1^2 + w_2 x_2^2 + \cdots + w_D x_D^2$$

Gaussian basis functions/radial basis functions

$$\phi_j(x) = e^{-\frac{1}{2\sigma^2}(x-\mu_j)^2}$$

Sigmoidal basis functions

$$g(\alpha) = \frac{1}{1+e^{-\alpha}}$$

tanh basis functions

$$h(\alpha) = \frac{e^{2\alpha}-1}{e^{2\alpha}+1}$$

Summary

- Linear and non-linear basis functions may be used to formulate a linear regression function
- OLS used to estimate linear regression weights by minimising sum of squared residuals
- OLS solution boils down to computing pseudoinverse of the Design Matrix
- Linear regression models can be fit to data using gradient descent