Machine learning: a hands-on introduction

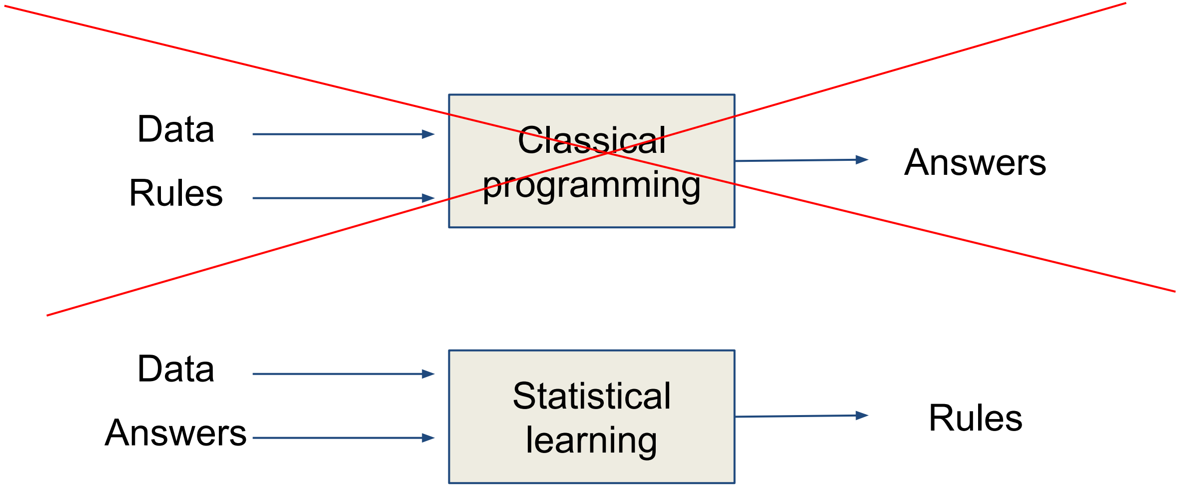
Filippo Biscarini (CNR, Milan, Italy)

filippo.biscarini@cnr.it

Supervised learning

What is learning?

What is learning?



What is learning?

- building a statistical model for **predicting** an **output** based on one or more **inputs**
- statistical learning model is **trained** rather than explicitly programmed

What is learning?

1. <u>Input data</u> (e.g. genome variants, metabolites)
2. <u>Output examples</u> (e.g. disease status, biological characteristics)
3. <u>Performance measure</u>: how well is the algorithm working → adjustment steps

   → **learning**

You can do (statistical) learning in your head!

- The current price of an electric Tesla is $100,000
- The price of an electric Tesla next year will be $90,000
- The price of an electric Tesla in two years will be $81,000
- The price of an electric Tesla in three years will be $72,900
- How much will an electric Tesla cost in five years?

You can do (statistical) learning in your head!

TRAINING DATA

- The current price of an electric Tesla is $100,000
- The price of an electric Tesla next year will be $90,000
- The price of an electric Tesla in two years will be $81,000
- The price of an electric Tesla in three years will be $72900
- How much will an electric Tesla cost in five years?

NEW, UNKNOWN DATA

PRICE = PRICE_0*(1-0.10)^YEARS

PRICE IN FIVE YEARS= $59,049

MATHEMATICAL MODEL

PREDICTION

Why supervised?

<u>Training examples</u>

measured variables / features on *n* examples

$$\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 0.12 & 1.5 & \dots & 0.9 \\ 2.05 & 0.95 & \dots & 1.1 \\ \vdots & \vdots & \ddots & \vdots \\ 3.5 & 0.88 & \dots & 1.75 \end{bmatrix}$$

labels / target variables (e.g. phenotypes) on n examples

Unsupervised learning

measured variables / features
on *n* examples

$$\cancel{\begin{bmatrix}1\\0\\\vdots\\1\end{bmatrix}} = \begin{bmatrix} 0.12 & 1.5 & \dots & 0.9 \\ 2.05 & 0.95 & \dots & 1.1 \\ \vdots & \vdots & \ddots & \vdots \\ 3.5 & 0.88 & \dots & 1.75 \end{bmatrix}$$

The steps of a supervised learning problem

1. Collect the data
2. EDA and data preparation
3. Training a model on the data
4. Evaluate model performance
5. Improve model performance

80% of the time!

rinse and repeat!

model deployment

A little ML jargon

- **example** (record, observation)
- **feature** (independent variable, factor)
- **label** (dependent variable)
- **method**: the statistical method used for a problem
- **model**: the modelling of the problem (e.g. which features to include and how)
- **algorithm**: the technique by which the method is applied to the model and solved
- **training data**: data on which the ML algorithm is trained

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.0	2.0	3.5	1.0	versicolor
6.2	2.2	4.5	1.5	versicolor
6.0	2.2	4.0	1.0	versicolor
6.0	2.2	5.0	1.5	virginica
6.3	2.3	4.4	1.3	versicolor
5.5	2.3	4.0	1.3	versicolor
5.0	2.3	3.3	1.0	versicolor
4.5	2.3	1.3	0.3	setosa
5.5	2.4	3.8	1.1	versicolor
5.5	2.4	3.7	1.0	versicolor
4.9	2.4	3.3	1.0	versicolor
6.7	2.5	5.8	1.8	virginica
6.3	2.5	4.9	1.5	versicolor

Regression and classification

Supervised learning problems

- Regression problems
- Classification problems

Supervised learning problems

- Regression (**predictive**) problems
  - target (continuous) variable, output
- Classification (**predictive**) problems
  - label, class (qualitative variable): binomial, multinomial, ordinal, nominal

"given a set of data, the learning algorithm attempts to optimize a function (the model) to find the combination of feature values that result in the target output"

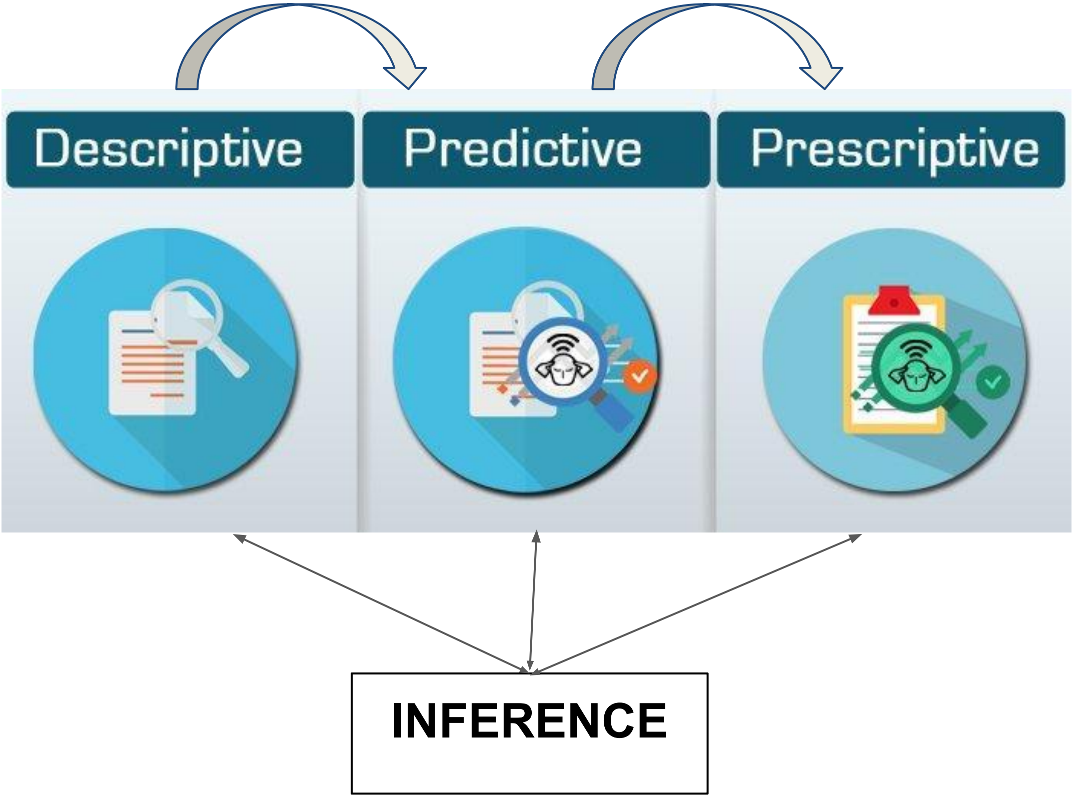
Supervised learning problems

- Regression (**predictive**) problems
    - target (continuous) variable, output
- Classification (**predictive**) problems
    - label, class (qualitative variable): binomial, multinomial, ordinal, nominal

<u>Predict</u>:

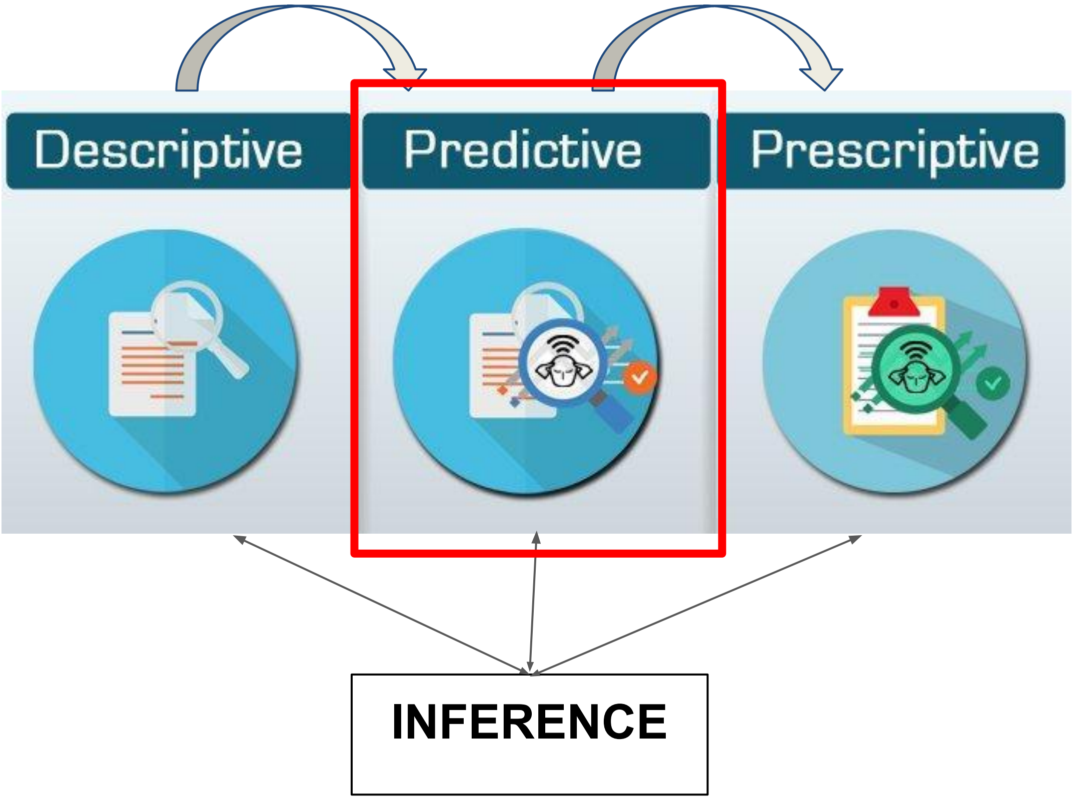
- the future (forecasting, prognosis)
- the unknown/unseen (e.g. sick/healthy, genetic predisposition etc.)
- real time (e.g. control traffic lights at rush hours)
- the past (e.g. when something happened, like conception date based on hormone levels)

Supervised learning problems



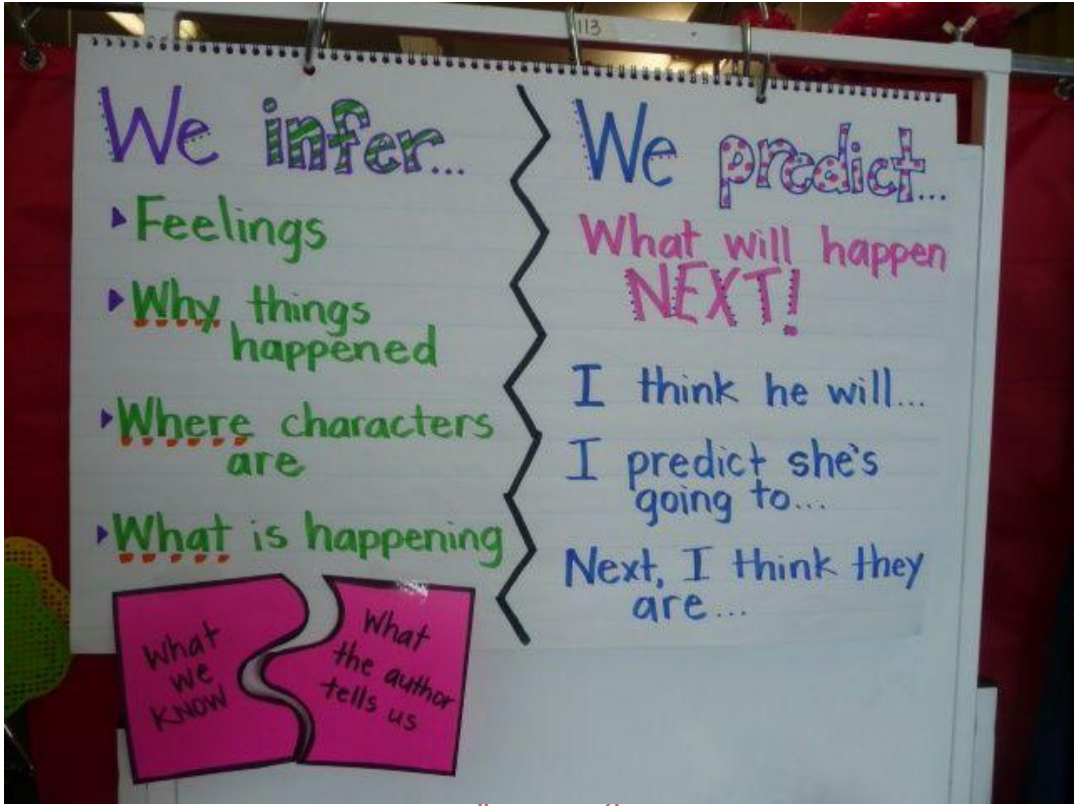
- Know the past
- Predict the future
- Act consequently

Supervised learning problems



- Know the past
- Predict the future
- Act consequently

Inference vs Prediction



- different statistical problems
- different objectives, different rules … different ballparks
- inference is in general more difficult than prediction

Supervised learning problems

- Regression (**predictive**) problems
- Classification (**predictive**) problems

Predictive machines!

- Classifiers
- Predictors/Regressors

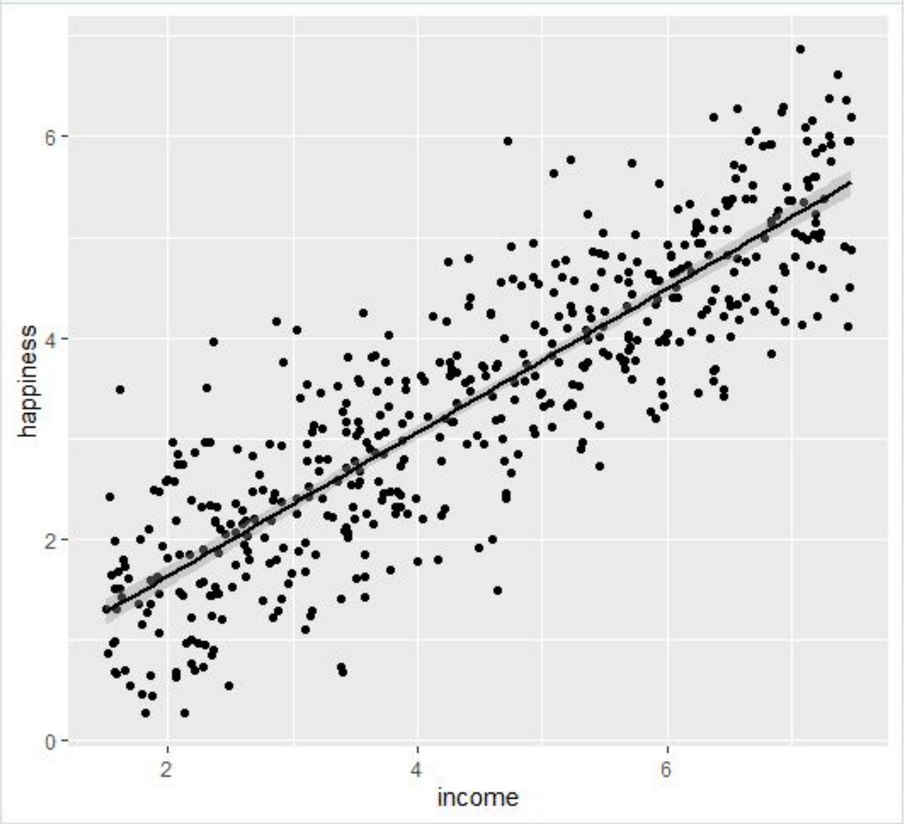
source:
https://blog.bigml.com/2013/03/12/machine-learning-from-streaming-data-two-problems-two-solutions-two-concerns-and-two-lessons/

Regression

Regression problems

- the response variable **y** is **quantitative**
- e.g.: *height*, *weight*, *yield* (milk, crops), *blood sugar concentration*
- **y** = **target** (dependent) variable (a.k.a. response, objective variable)
- **X** = matrix of **features** (continuous, categorical)
- **predictor**: y = f(x) = **P(X)** ← [predictive machine]

Regression problems - simple regression



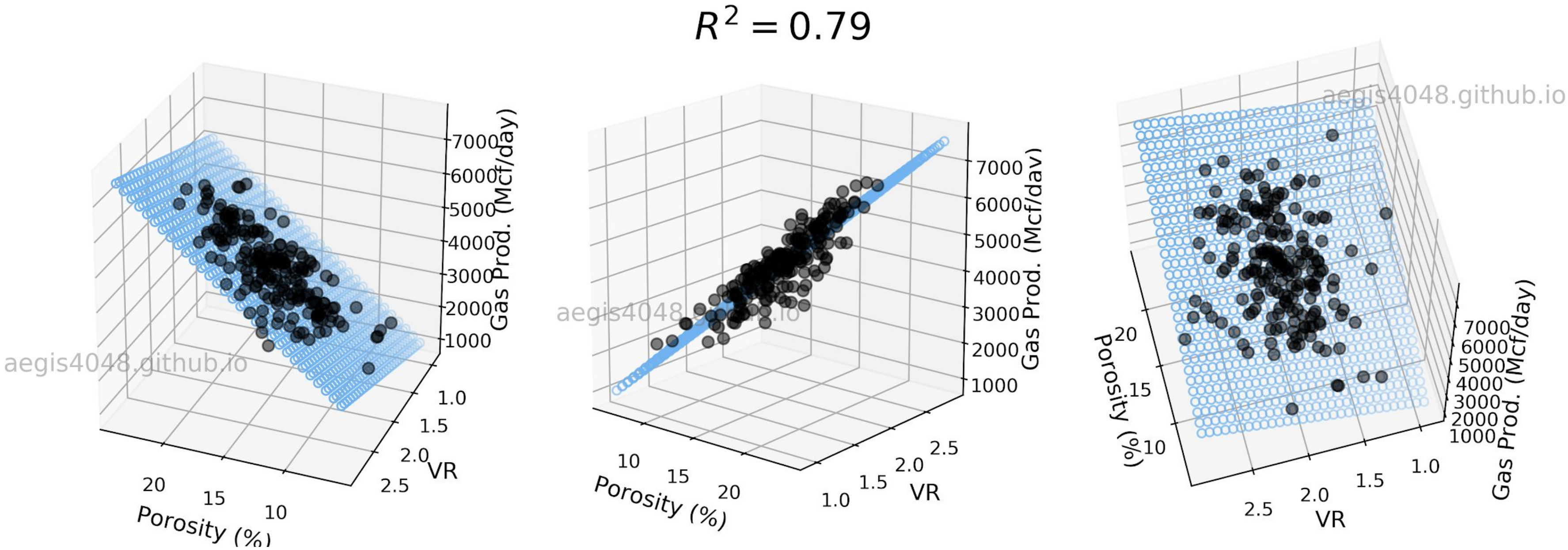
happiness = (intercept) + beta*income

or

income = (intercept) + beta*happiness

Source: https://www.scribbr.com/statistics/linear-regression-in-r/

Regression problems - multiple regression



Source: https://aegis4048.github.io/mutiple_linear_regression_and_visualization_in_python

Multiple linear regression

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \epsilon$$

- y: target variable
- **β**'s: model coefficients
- X's: features (predictors, independent variables, factors)

Multiple linear regression

$$\mathbf{y} = \beta\mathbf{X} + \mathbf{e}$$

- matrix (compact) notation
- vectors of observations (**y**), coefficients (**β**) and residuals (**e**)
- matrix of features (**X**)

Multiple linear regression

$$\mathbf{y} = \beta \mathbf{X} + \mathbf{e}$$

estimation of coefficients

$$\hat{\mathbf{y}} = \hat{\beta} \mathbf{X}$$

→ **predictions!**

- matrix (compact) notation
- vectors of observations (**y**), coefficients (**β**) and residuals (**e**)
- matrix of features (**X**)

Predictions

$$\hat{y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p$$

with the estimated coefficients $\boldsymbol{\beta}$ and the feature values **X** we obtain the predicted values $\hat{y}$

Predictions

$$\hat{y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p$$

with the estimated coefficients $\boldsymbol{\beta}$ and the feature values **X** we obtain the predicted values $\hat{y}$

→ how do we obtain the model coefficients $\boldsymbol{\beta}$?

Estimation of model coefficients

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \epsilon$$

- define a **loss (cost) function**
- **minimise** the loss function

Estimation of model coefficients

- define a **loss (cost) function**

observations	**predictions**
$\mathbf{y}$	$\hat{\mathbf{y}} = \hat{\beta}\mathbf{X}$

difference between observed and predicted values

Estimation of model coefficients

- define a **loss (cost) function**

observations	predictions
$\mathbf{y}$	$\hat{\mathbf{y}} = \hat{\beta}\mathbf{X}$

difference between observed and predicted values

LEAST SQUARES

Estimation of model coefficients

- minimise the **loss (cost) function**

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$RSS = \sum_{i=1}^{n} (y_i - \hat{\beta}_i X_i)^2$$

$$\underset{(\beta)}{\text{minimize}} \; (RSS)$$

LEAST SQUARES

Estimation of model coefficients

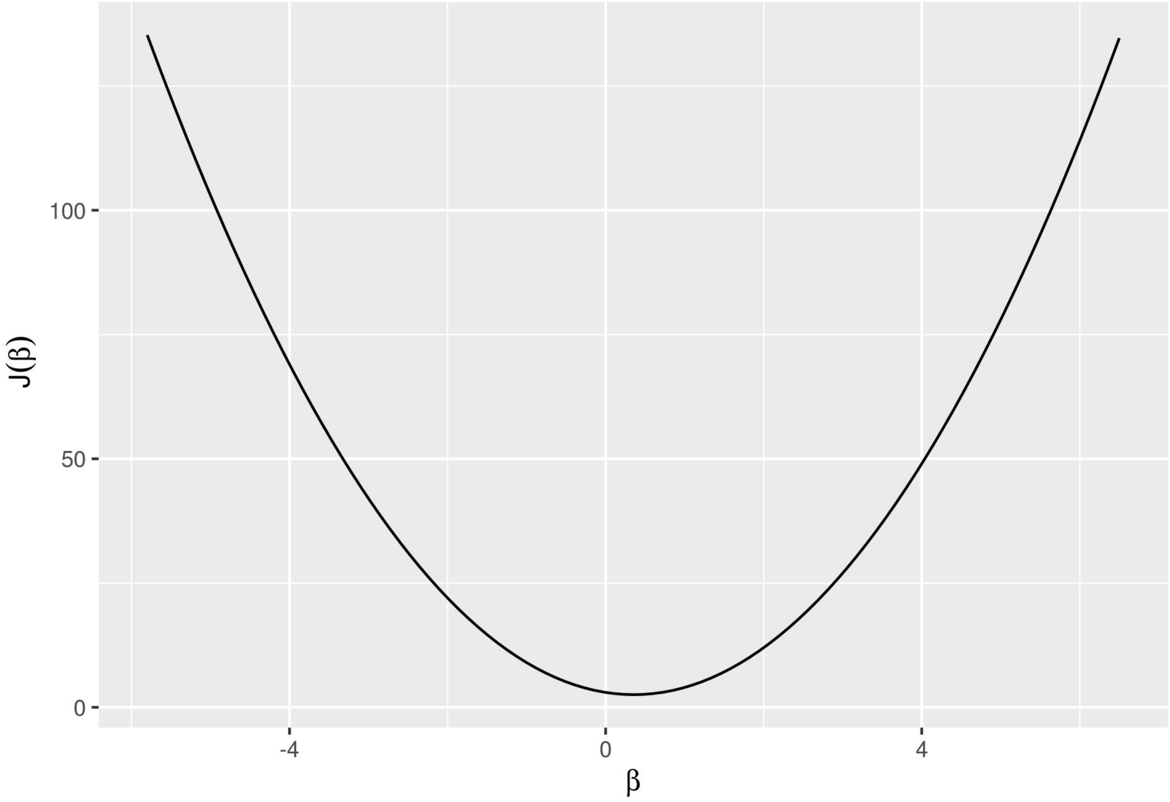
- minimise the **loss (cost) function**

$$J(\beta) = \frac{1}{2n}\sum_{i=1}^{n}\left(\beta_i X_i - y_i\right)^2$$

$$\underset{\beta}{\text{minimize}}\, J(\beta)$$

modified (normalized) RSS function

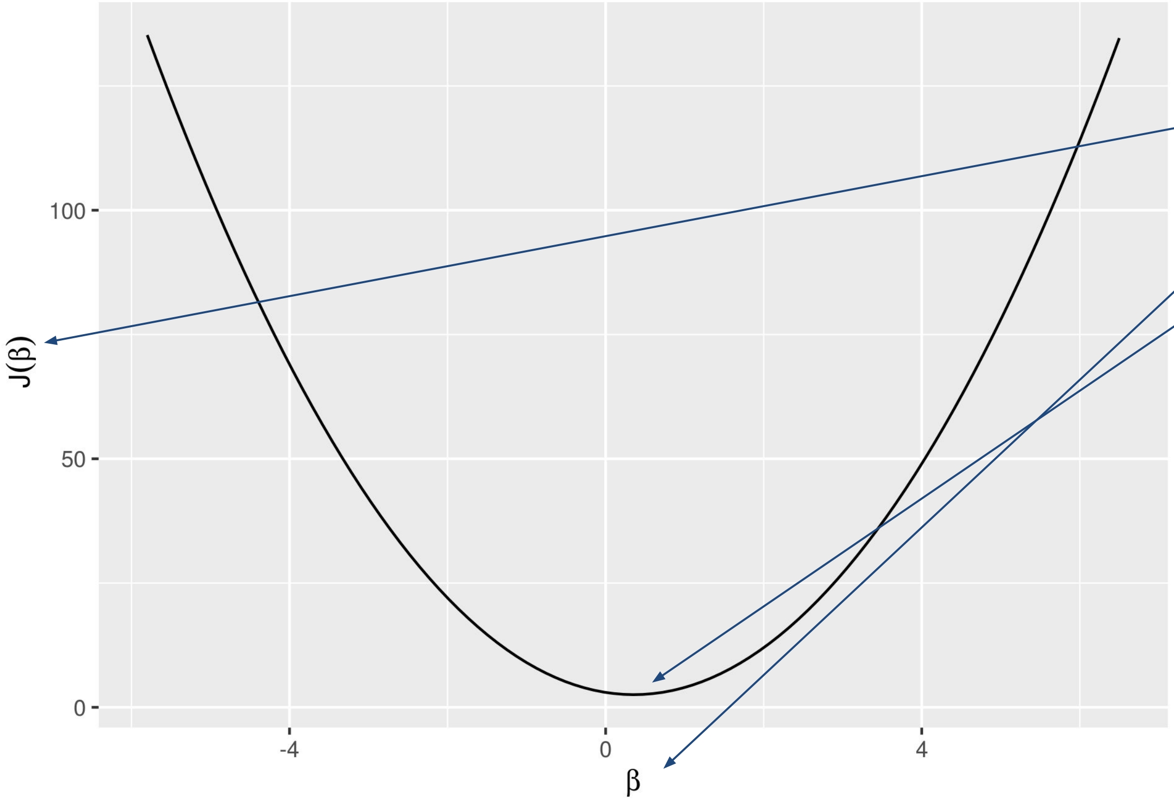
Minimise the loss function



Simple linear regression (1 parameter):

$$y = \beta \cdot x$$

Minimise the loss function

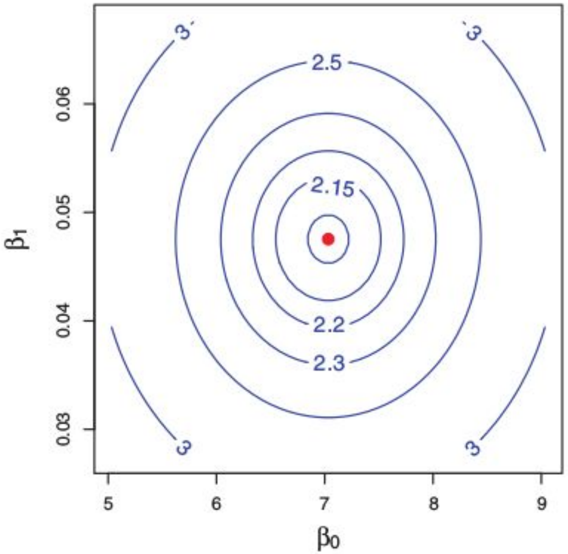


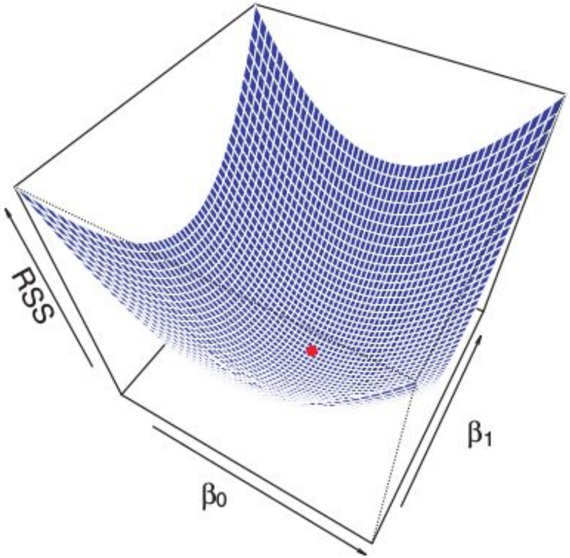
1. loss function
2. model parameters
3. minimum

Simple linear regression (1 parameter):

$$y = \beta \cdot x$$

Minimise the loss function

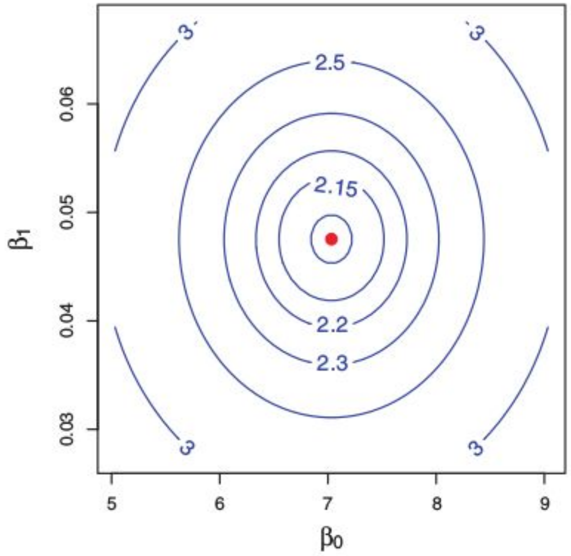


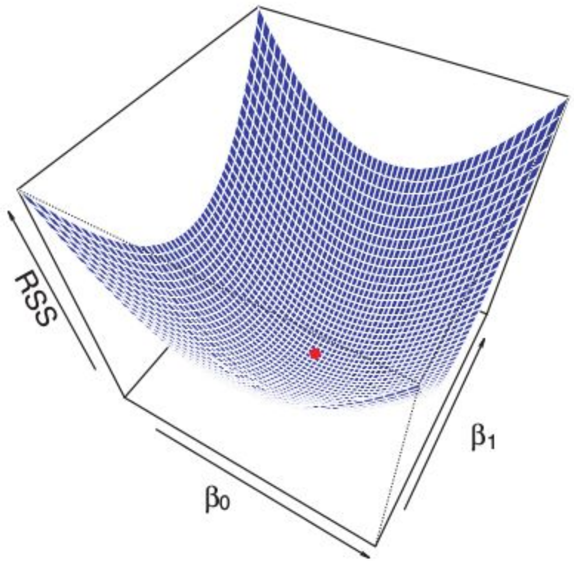


Multiple linear regression (e.g. 2 parameters):

$$y = \beta_0 + \beta_1 \cdot x$$

Minimise the loss function





Multiple linear regression (e.g. 2 parameters):

$$y = \beta_0 + \beta_1 \cdot x$$

Multiple linear regression (> 2 parameters):

→ m-dimensional hyperspace

Minimise the loss function

- Demonstration 1.1
- Exercise 1.1

→ 1.introduction_to_ml.Rmd

Minimising the cost function

- the defined cost function is convex (it has a minimum!)
- can be minimised by **gradient descent**
- machine learning perspective: gradient descent is a general algorithm to solve models
- alternatively:
    - maximum likelihood
    - (non-)linear least squares

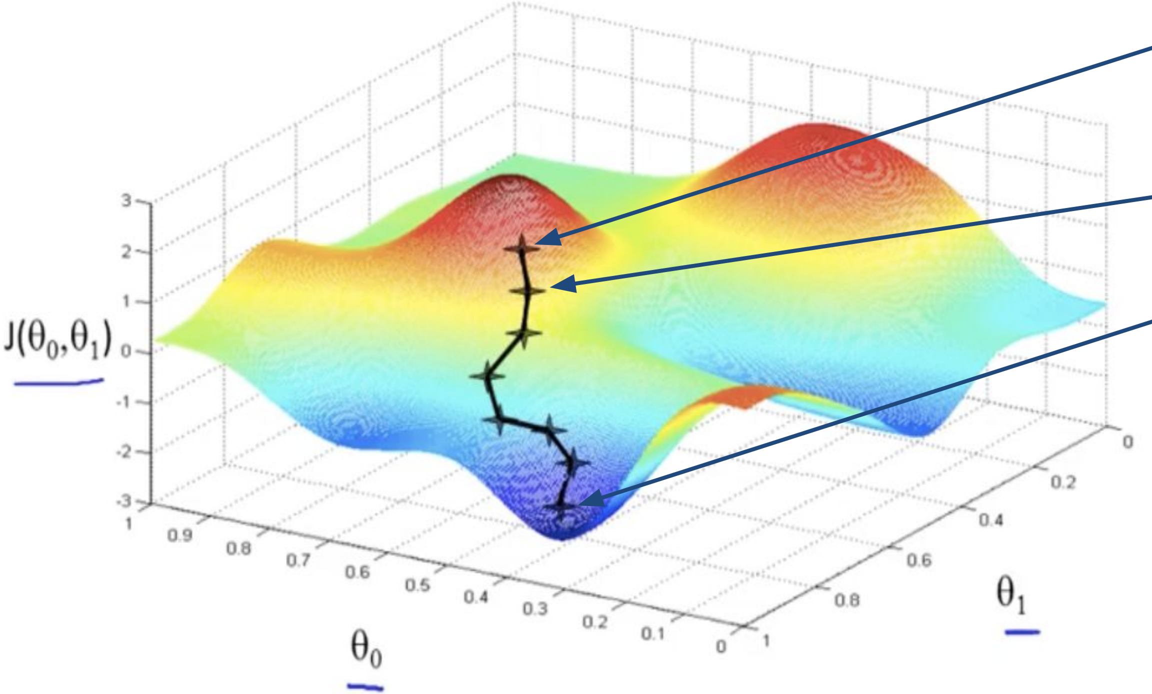
Loss function: finding the minimum?

Gradient Descent:

$$\underset{\beta}{\text{minimize}}\, J(\beta)$$

1. Start with initial values for β : (initialisation)
2. Change β in the direction of reducing J(β) : (descent)
3. Stop when the minimum is reached : (minimisation)

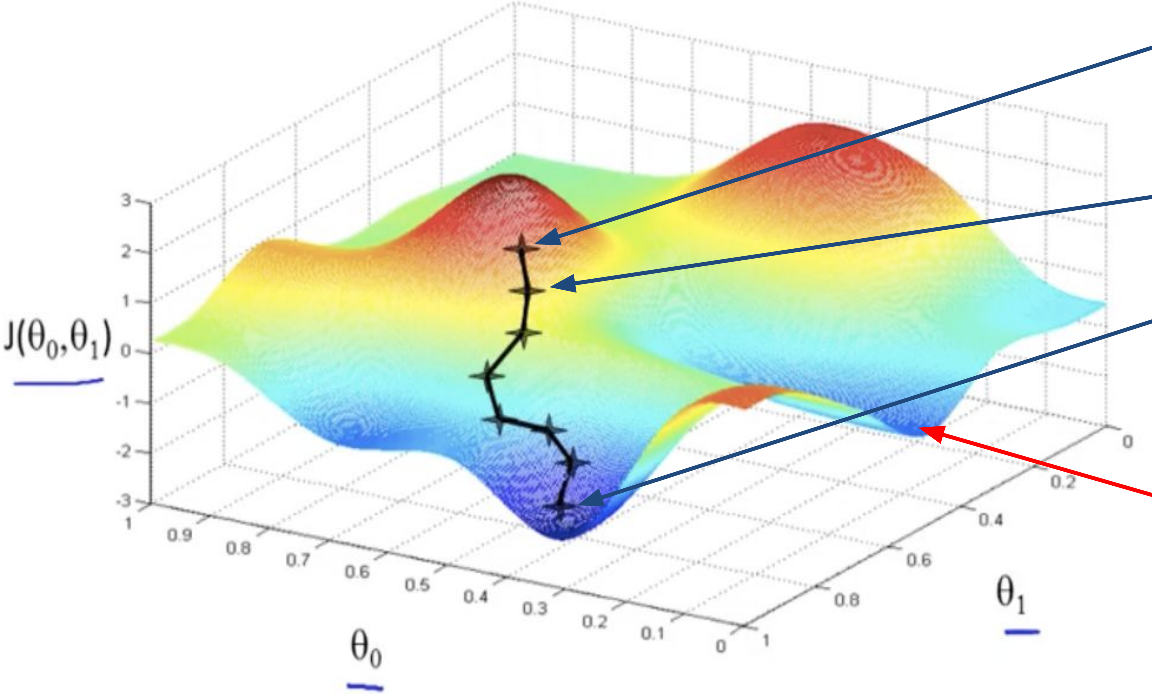
Gradient descent



1. starting point (initialisation)
2. find the steepest direction around the starting point
3. take one step in this direction
4. repeat until convergence (local minimum)

Source: Andrew Ng
https://medium.com/@DBCerigo/on-why-gradient-descent-is-even-needed-25160197a635

Gradient descent



1. starting point (initialisation)
2. find the steepest direction around the starting point
3. take one step in this direction
4. repeat until convergence (local minimum)

changing initialisation may lead to a different local minimum!

Source: Andrew Ng
https://medium.com/@DBCerigo/on-why-gradient-descent-is-even-needed-25160197a635

Gradient descent

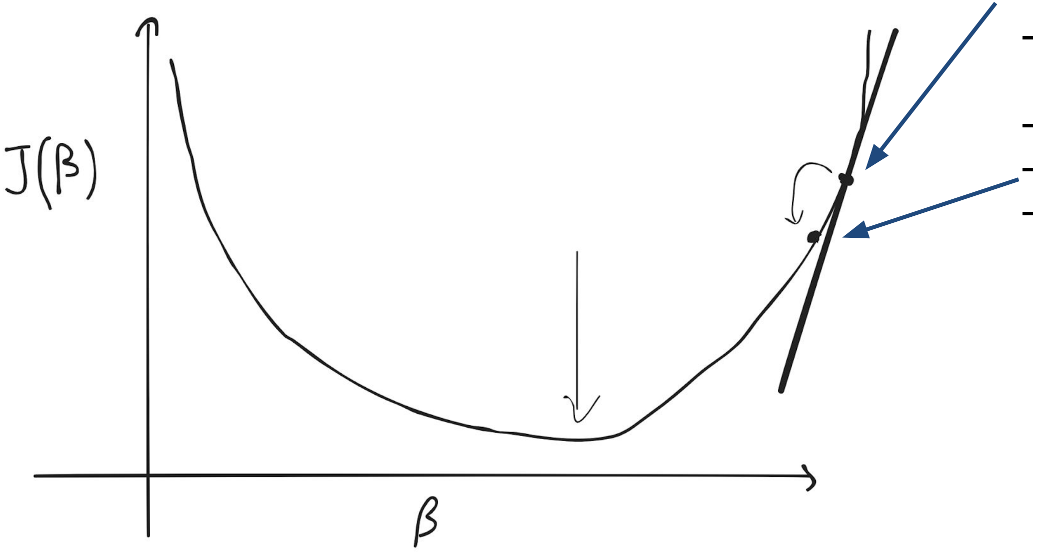
$$\beta_j := \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\beta)$$

assignment operator

learning rate (size of the steps)

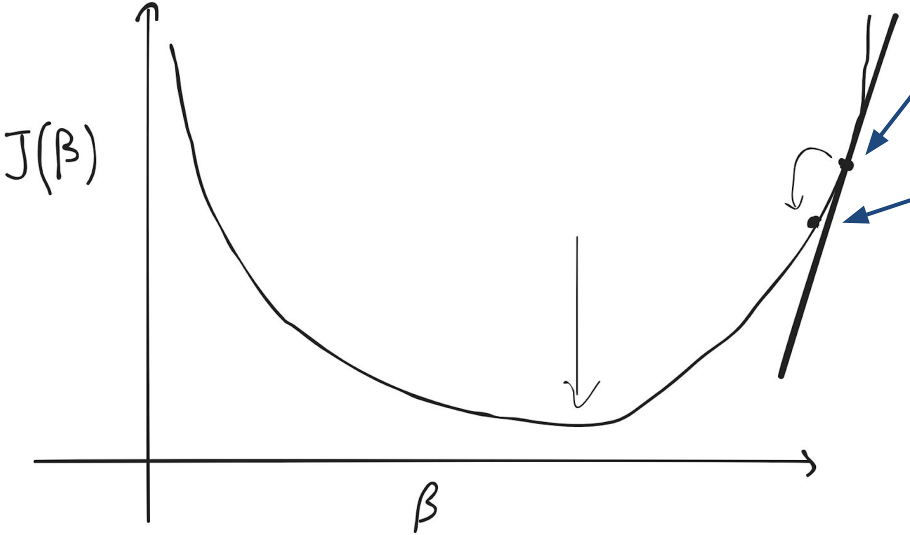
Partial derivative of J($\boldsymbol{\beta}$) with respect to $\boldsymbol{\beta}_j$

Gradient descent



- starting point (initial value for β)
- calculate the (partial) derivative in that point
- → positive slope
- update the value for β
- repeat

Gradient descent

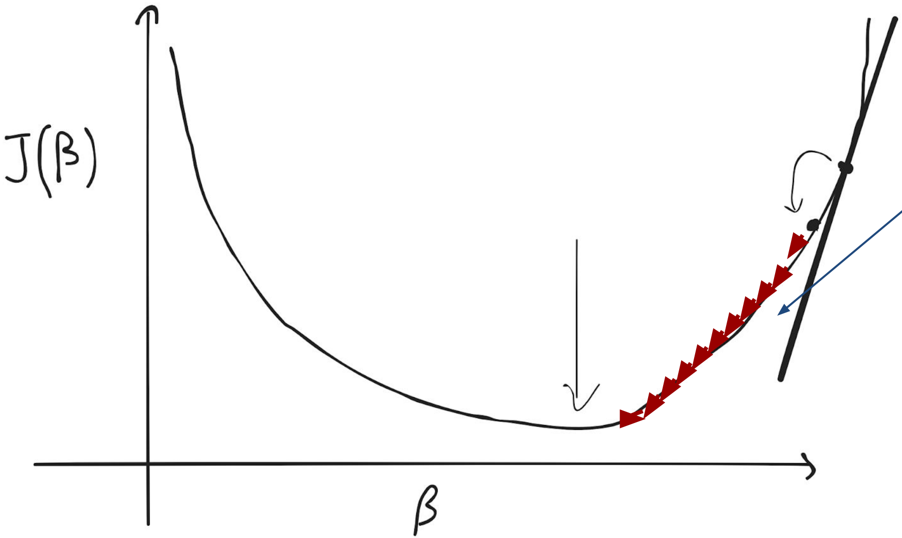


- starting point (initial value for β)
- calculate the (partial) derivative in that point
- → positive slope
- update the value for β
- repeat

$$\beta_j := \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\beta)$$

- positive slope → reducing the value of β (and the other way around)

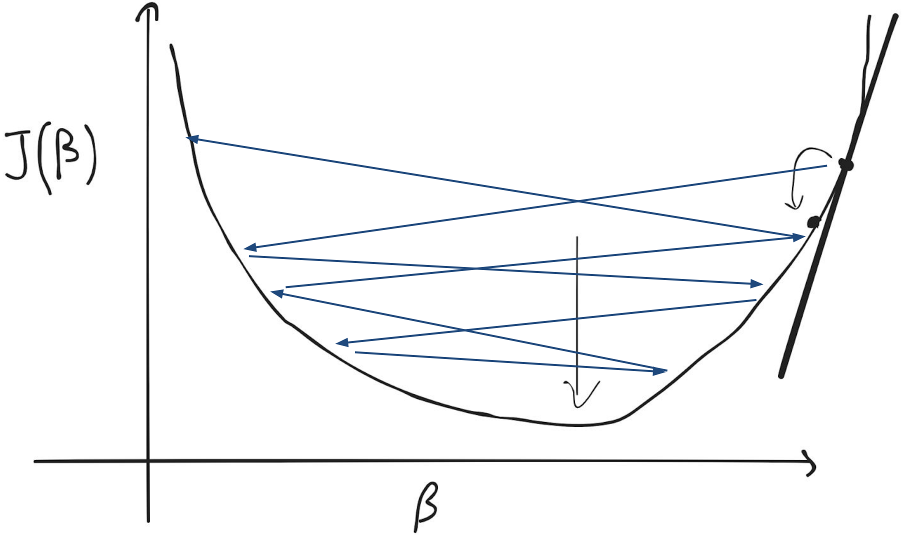
Gradient descent



- **α** controls the size of the updating step
- small **α** → slow descent

$$\beta_j := \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\beta)$$

Gradient descent

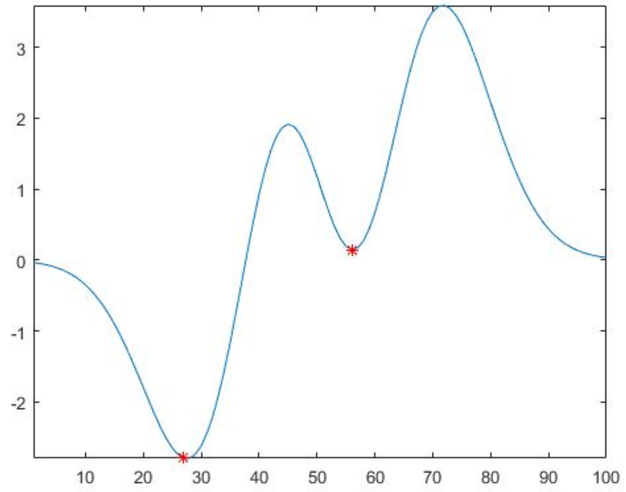


- $\boldsymbol{\alpha}$ controls the size of the updating step
- large $\boldsymbol{\alpha}$ → overshooting: failure to converge

$$\beta_j := \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\beta)$$

Gradient descent - recap

- general method to **solve machine learning models** (e.g. multiple linear regression)
- optimise (minimise) the loss function → **optimiser**
- importance of the **learning rate**
- local minimum → **momentum**

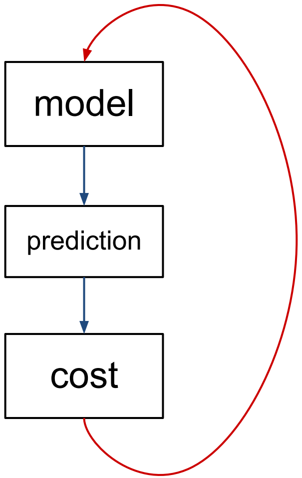


Linear regression - recap

1) $$y = X \cdot \beta + e$$

2) $$\hat{y} = X\hat{\beta}$$

3) $$J(\beta) = \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \hat{\beta}_i X_i \right)^2$$



[minimise J(B) - take derivatives - and update parameters]

Take away message

- **linear regression** from the machine learning perspective →a **predictive machine**
- to be able to make predictions, we first need to **estimate parameter coefficients**
- define a **loss function** and then use **gradient descent** to **minimize it**
- partial derivatives are used to **update** the values of the **parameters**
- **gradient descent** is a general method to minimise the loss (cost) function for a variety of machine learning models

Measuring performance

- we have our model
- we have estimated the parameters (coefficients) of the model
- we can now get predictions from our predictive machine

Measuring performance

- we have our model
- we have estimated the parameters (coefficients) of the model
- we can now get predictions from our predictive machine

→ how well are we doing?

(root) Mean squared error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f(x_i)\right)^2$$

- average squared difference between predictions and observations

$$RMSE = \sqrt{MSE}$$

- on the same scale as the target variable

Mean absolute error (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - f(x_i)|$$

- less sensitive to outliers

and the normalized version:

$$NMAE = \frac{MAE}{\bar{y}}$$

Correlations

- **Pearson's** linear correlation coefficient: $\rho_{y,\hat{y}}$

- **Spearman's** rank correlation coefficient: $\rho_{ry,r\hat{y}}$

rank variables!

Measuring performance

- Demonstration 1.2
- Exercise 1.2

→ introduction_to_ml.Rmd