Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences
Western Switzerland

MSE | MASTER OF SCIENCE IN ENGINEERING

Machine Learning

T-MachLe

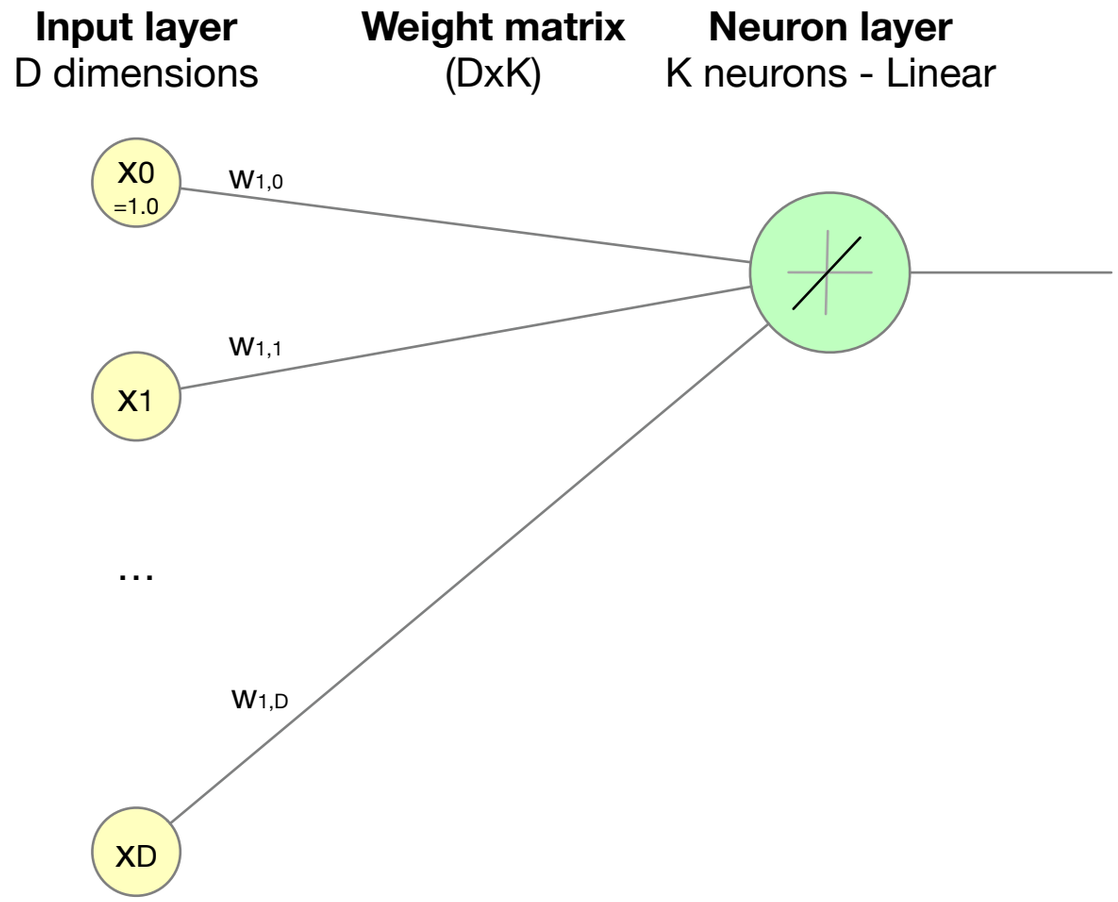
5. System Engineering - Preparing the data - Tuning the system

Jean Hennebert
Andres Perez Uribe

Re-do the exercise of last week PW with Tensorflow

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$h_\theta(\mathbf{x}) = \theta_0 x_0 + \theta_1 x_1$$



- A linear regression is a Dense layer with 1 neuron and a linear activation function

Re-do the exercise of last week PW with Tensorflow

Solution with Tensorflow

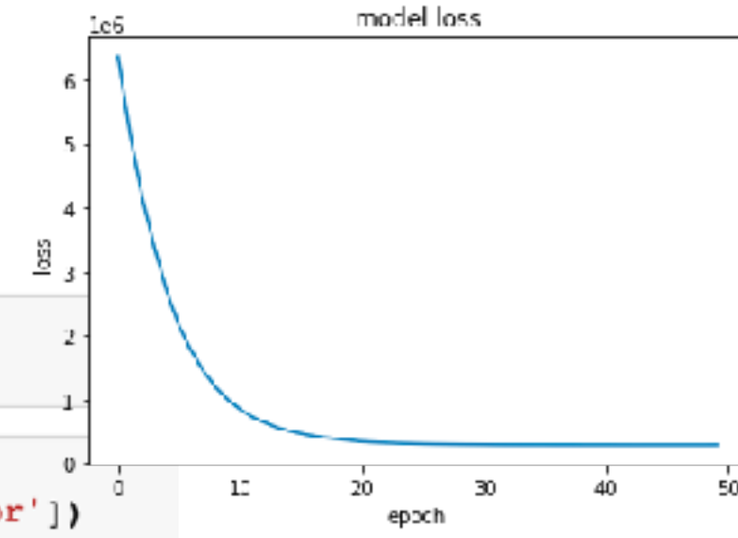
In [12]:
```
1 import tensorflow as tf
```

In [55]:
```
1 D=1
2 model = tf.keras.models.Sequential()
3 model.add(tf.keras.layers.Dense(1, input_shape=(D,), use_bias=True, activatic
4 model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_8 (Dense)              (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
```

In [56]:
```
1 X = living_area
2 y = rent_price
```

In [57]:
```
1 sgd = tf.keras.optimizers.SGD(learning_rate=0.000001)
2 model.compile(optimizer=sgd, loss='mse', metrics=['mean_squared_error'])
3 log = model.fit(X, y, batch_size=32, epochs=50)
4 model.evaluate(X,  y, verbose=2)
5 plt.plot(log.history['loss'])
6 plt.title('model loss')
7 plt.ylabel('loss')
8 plt.xlabel('epoch')
9 plt.show()
```



Debugging / optimising a learning algorithm

Activity

- Suppose you have implemented a linear regression to predict housing prices.
- You get a very good (low) cost function value on the training set.

$$J(\theta) = \frac{1}{2N} \sum_{n=1}^{N} (h_\theta(\mathbf{x}_n) - y_n)^2$$

- However, when you test your hypothesis on a new set of houses, you find that it makes unacceptably large errors in its predictions. What should you try next?

You may find solutions with this…

- Get more training examples
  - Yes but will it really help? When should we stop collecting data?
- Use less features, especially the ones that are bound to 2nd or 3rd orders $x_1^2, x_2^2, x_1^3, \ldots$
  - Yes, it will reduce the “non-linearities” of the system as we are maybe here “overfitting”. But are we overfitting here?
- Verify that some features are not predominant to the others, that you don’t have outliers that are perturbing the modelling
  - Yes, it is generally a good idea to visualise the data, find out about outliers and to normalise the data so that there is no numerical bias before training
- Use new features, ones that bring new, significant information
  - Yes, but it can be costly and imply fancy forms of encoding, e.g. how to encode a text description of the apartment

Remark. All these ideas seem correct a priori. We will provide now some tools to make the right decisions between these options.

Plan - Supervised Learning System Design and Evaluation

5.1 Generalised ML process

5.2 Data preparation - normalisation - encoding

5.3 The problem of overfitting

5.4 Model selection process

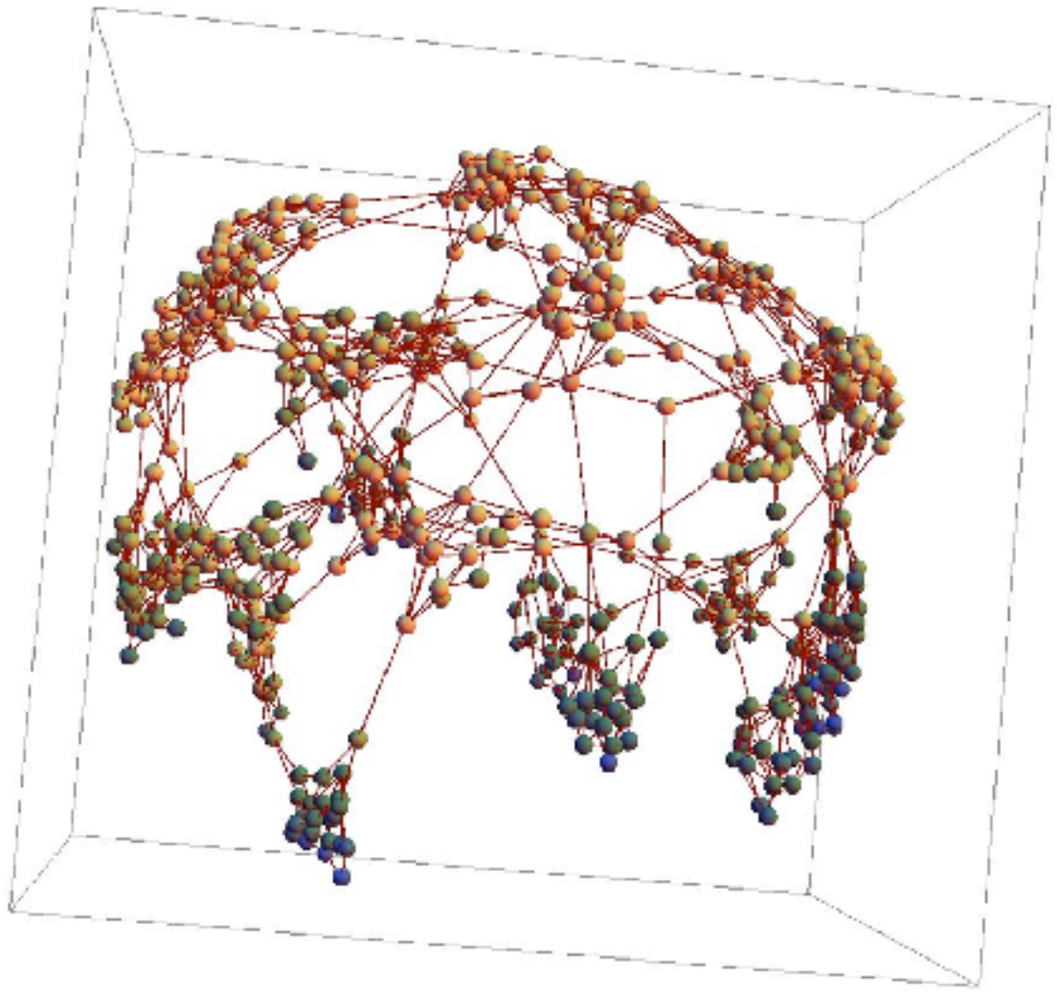
5.4 More data better data?

Practical Work 5

5.1 Generalised ML process

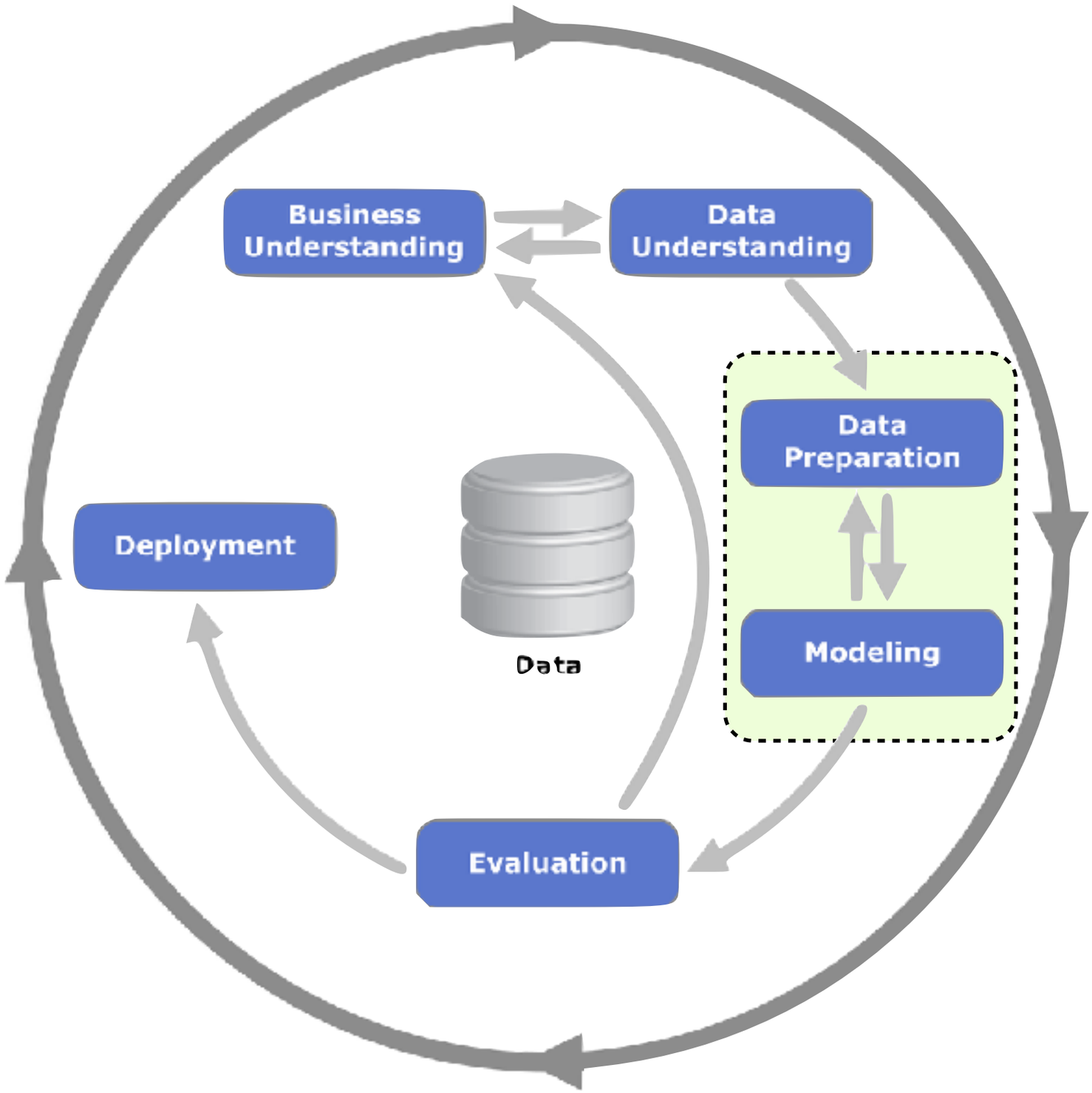
CRISP-DM Process

A Modular View of ML



Standardised Process - CRISP-DM

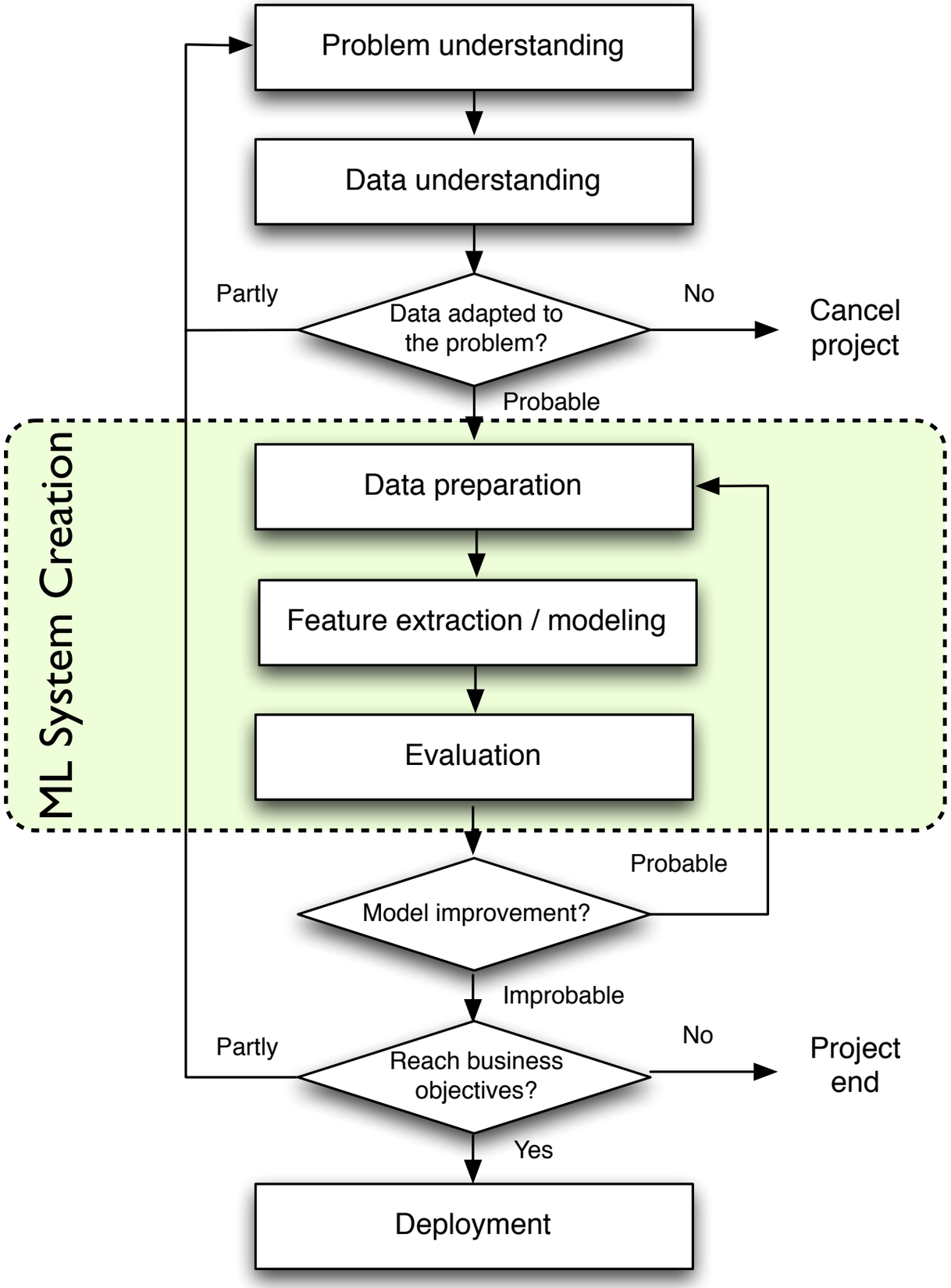
Cross Industry Standard Process for Data Mining



Source https://en.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining

Standardised Process - CRISP-DM

Cross Industry Standard Process for Data Mining



What problem should we solve?
What are the expectations?
What is known about the problem?

What data do we have available?
Is the data relevant to the problem?
Is it valid? Does it reflect our expectations?
Is the data quality, quantity, recency sufficient?

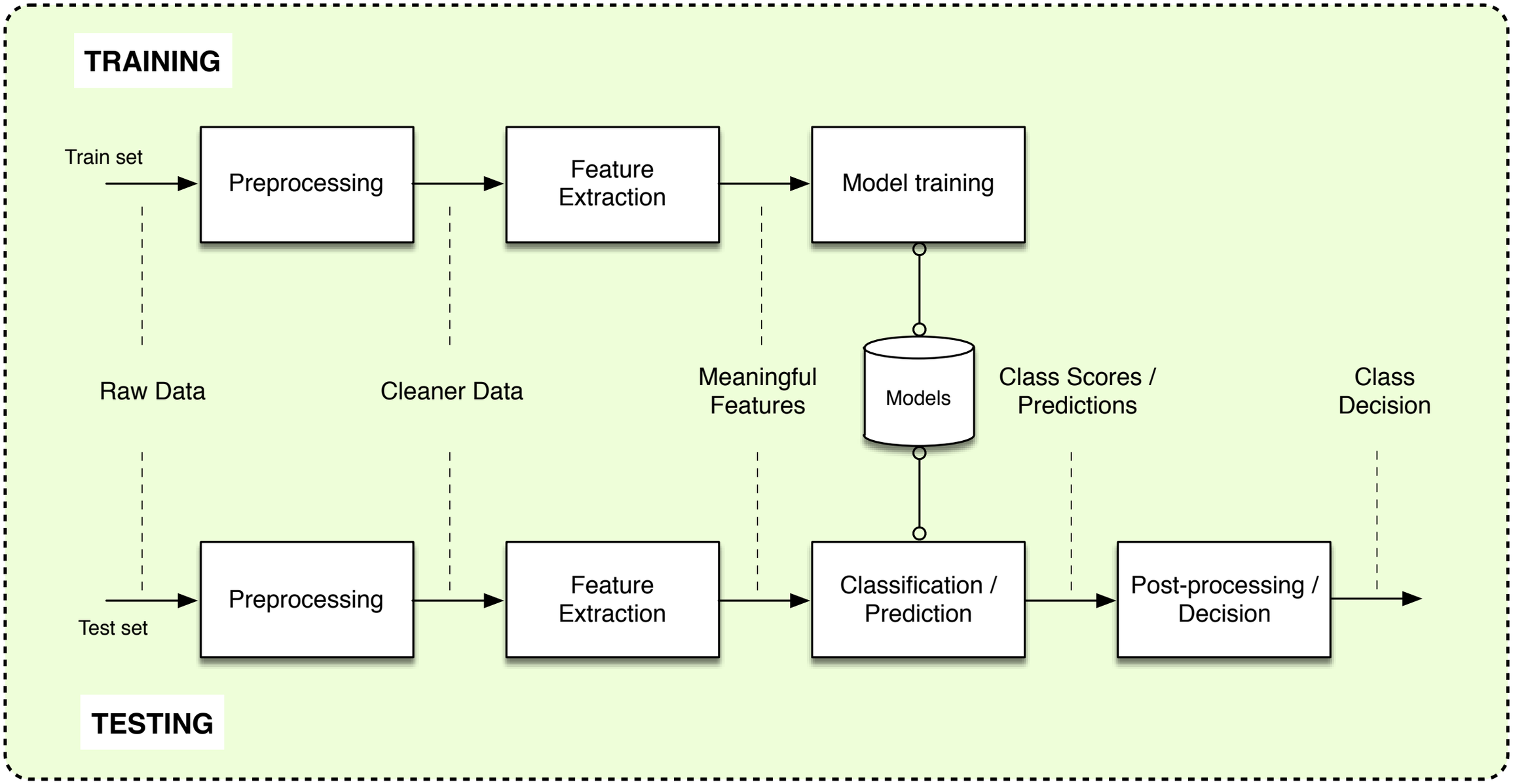
Which data should we concentrate on?
Are we confident in the data?
How may we increase the data quality?

How is the data best transformed for modeling?
What kind of model architecture suits the problem best?
What is the best technique to build the model?

How to evaluate the models?
How good does the model perform technically?
Are we confident that a model A is better than a model B?

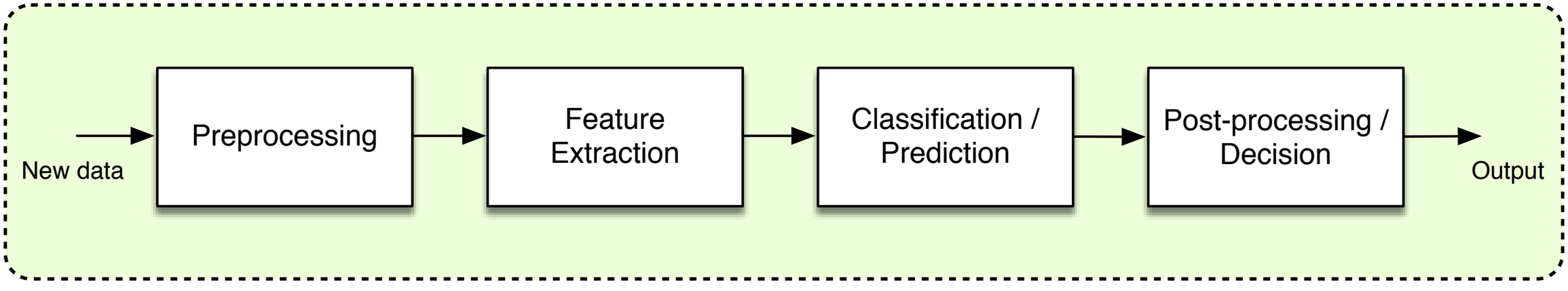
How is the model best deployed?
How do we know that the model is still valid?

ML System Creation - A Modular View



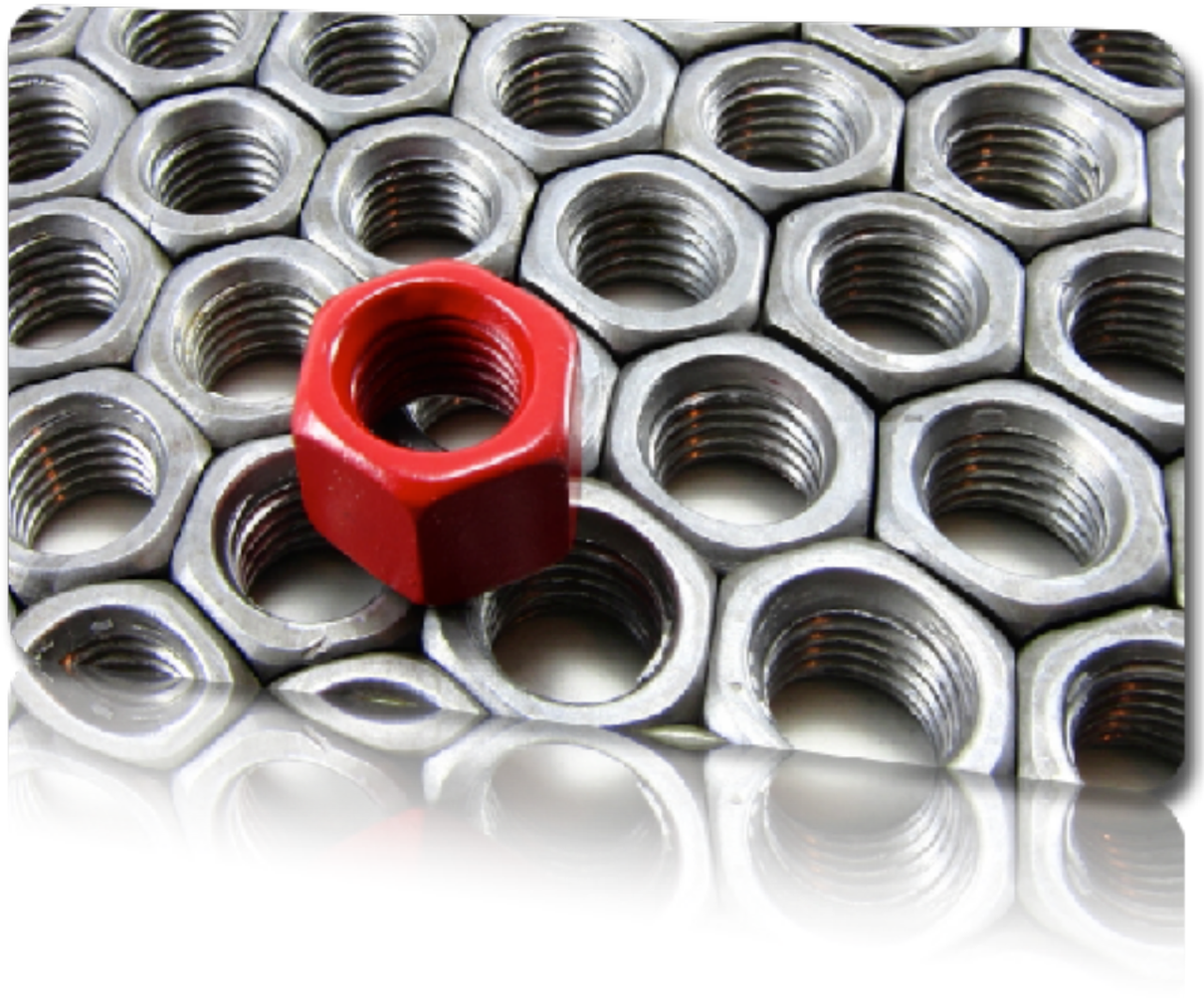
- We first go in an iterative train/test process to find the best configuration of the preprocessing / feature extraction / model choice

In Production System - A Modular View



- Once the system is optimised, we may use it to process new “live” data.
- This is called “inference” time.

5.2 Data Preparation



- Normalisation of numerical values
- Normalisation of output
- Encoding categorical values
- Encoding text values
- Encoding values with discontinuities

Example: some input variables are in very different ranges

Training data set

Surface (m2)	#bedrooms	Monthly
26	1	890
37	1.5	955
57	2	1630
48	2	1390
60	2.5	1700
57	2.5	1630
76	3	2060
80	3.5	2270
103	3.5	2550
142	4	4260
…	…	…

1

⋮

n

⋮

N

x_1 x_2 y

- The number of bedrooms is very small in comparison to the surface
- Some machine learning algorithms are “less sensitive” to features with smaller scales, e.g. KNN
- Other algorithms show problems of convergence with unmatching feature ranges, e.g. gradient descent

Data Normalisation

Normalisation - process aiming at giving similar range and "importance" to the columns of the data set

- Numeric values of features contained in data (*) may be
  - On different scales, ranging over different orders of magnitude:
    Should be brought to similar scales: *Feature scaling*.
  - Centred around different values (on average):
    Should approximately be centred around zero: *Feature centering*.
- This is important for various reasons:
  - *Numerical stability* of the learning algorithm and *improved convergence properties*.
  - Learning algorithms can be biased by bigger or skewed features

(*) **Applies to input data and output data** (output data only in regression problems).

Min-Max rescaling

$$x' = \frac{(x - x_{min})}{(x_{max} - x_{min})}$$

Rescaled into [0,1] range

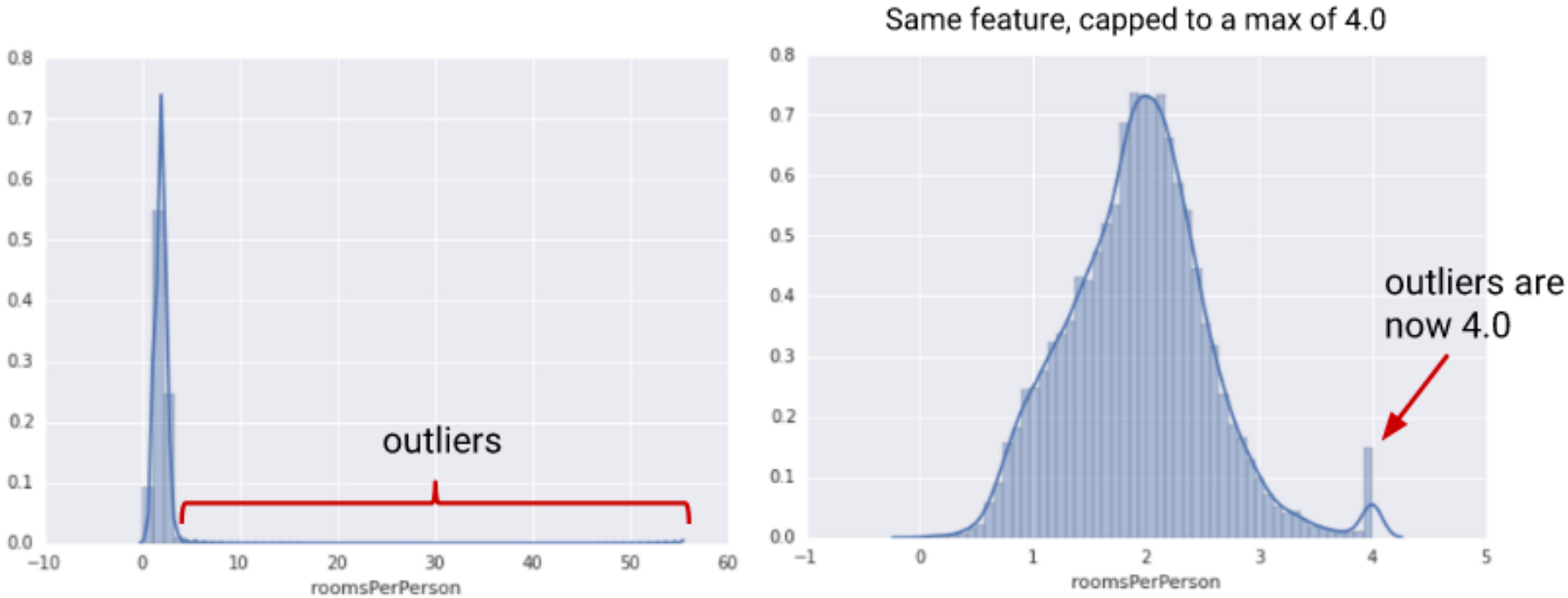
Min-Max normalisation

$$x' = 2\frac{(x - x_{min})}{(x_{max} - x_{min})} - 1$$

Rescaled into [-1,1] range

- The x_{min} and x_{max} are respectively the minimum and maximum values of the considered feature
  - WARNING: you should compute these values on the training set only. Computing them on the whole data set (including the test set) is considered as “cheating”.
- Problem in the case of outliers, see next slide

Feature clipping



From https://developers.google.com/machine-learning/data-prep/transform/normalization

- In case of outliers, min-max rescaling would “squeeze” most samples in a small part of the [0,1] scale
- In this case, apply a feature clipping so that the min and max values are not anymore in the outliers range
- Another solution when there are not so many outliers is to apply z-norm, see next slide

zero-normalisation (or “z-norm”)

$$x' = \frac{(x - \mu)}{\sigma}$$

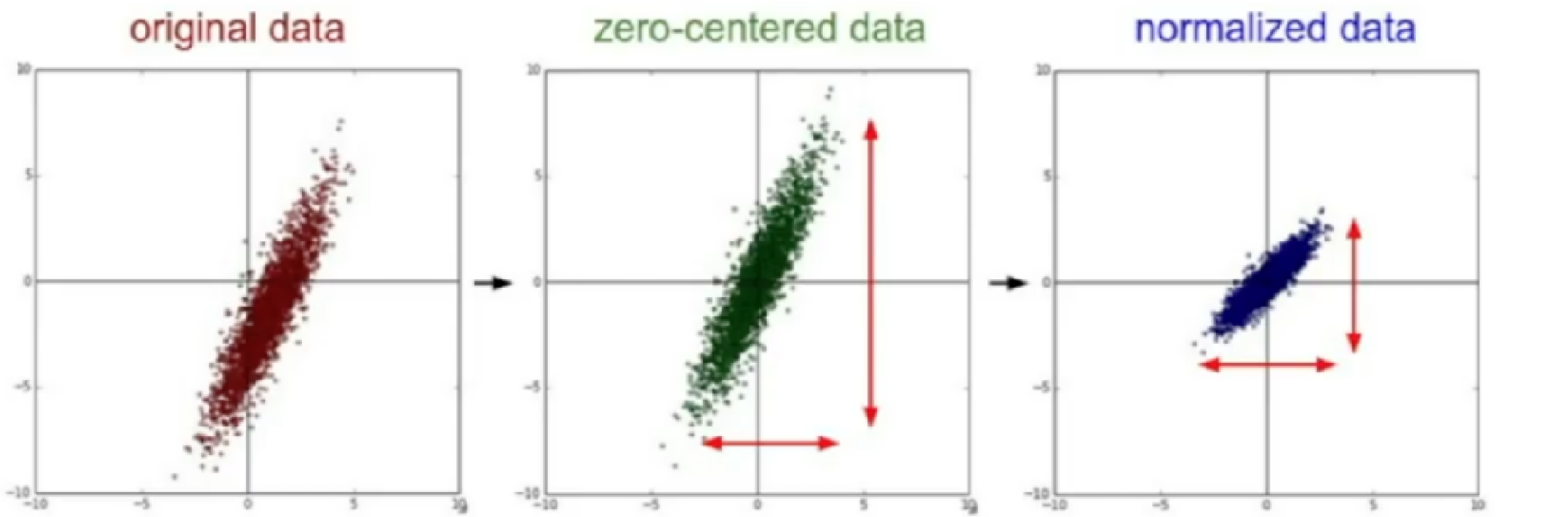
where

$$\mu = \frac{1}{N_{train}} \sum_{n=1}^{N_{train}} x_n$$

$$\sigma^2 = \frac{1}{N_{train}} \sum_{n=1}^{N_{train}} (x_n - \mu)^2$$

- The z-normalisation is rescaling the data with a zero-mean and a unit-variance.
- Less problematic in the case of outliers than min-max rescaling
- The μ and σ are respectively the average and standard deviation for the given feature
  - WARNING (again): you should compute these values on the training set only. Computing them on the whole data set (including the test set) is considered as cheating.

z-norm example




```
X -= np.mean(X, axis = 0)
```

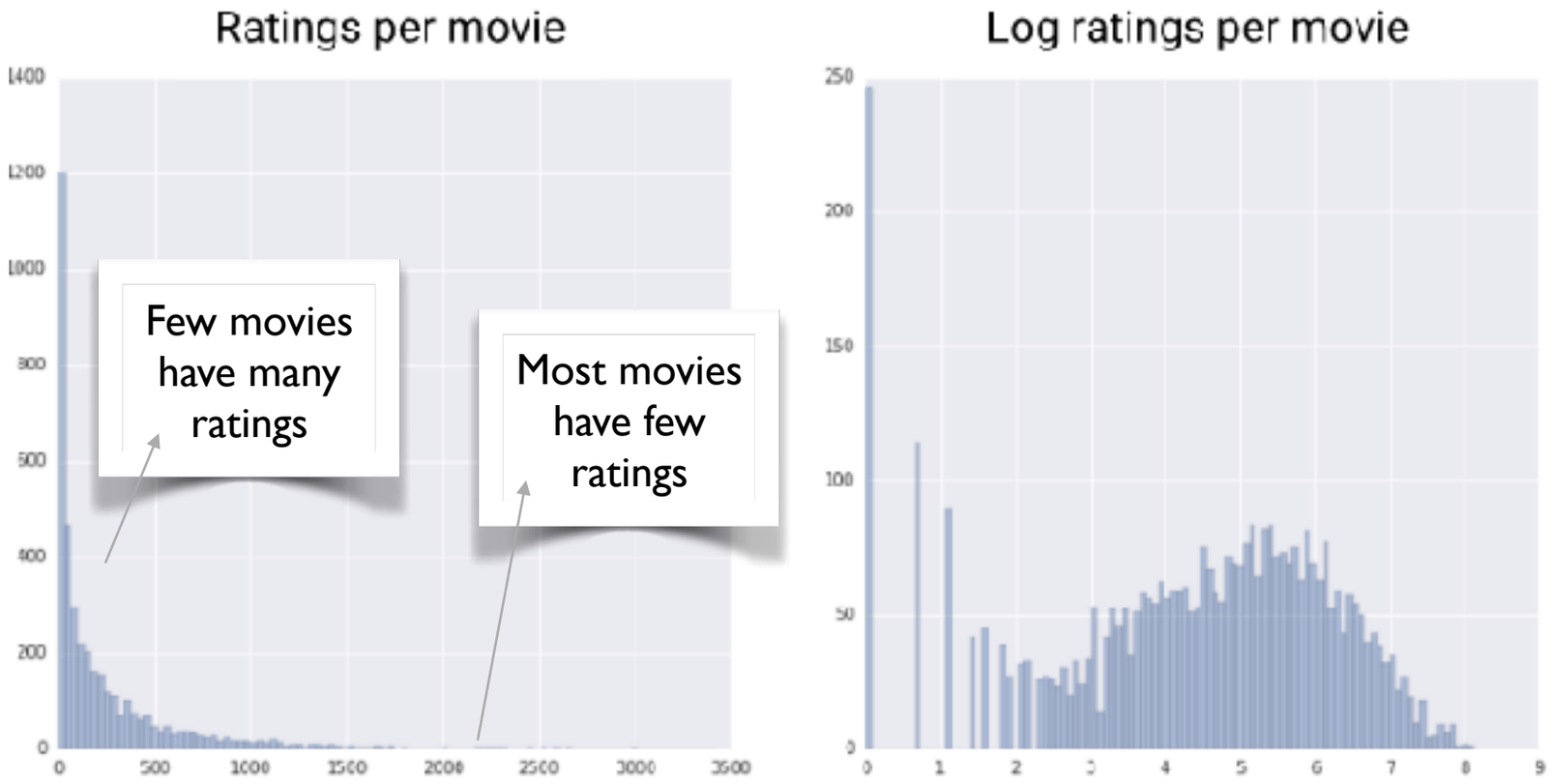
```
X /= np.std(X, axis = 0)
```

From Fei-Fei Li & Justin Johnson & Serena Yeung, April 2018: CS231n Stanford

Log scaling

$$x' = \log(x)$$

- Compression of a wide range to a narrow range
- Helpful when a handful of your values have many points, while most other values have few points
- Ex: x = number of movie ratings
- Warning: your x must be positive!

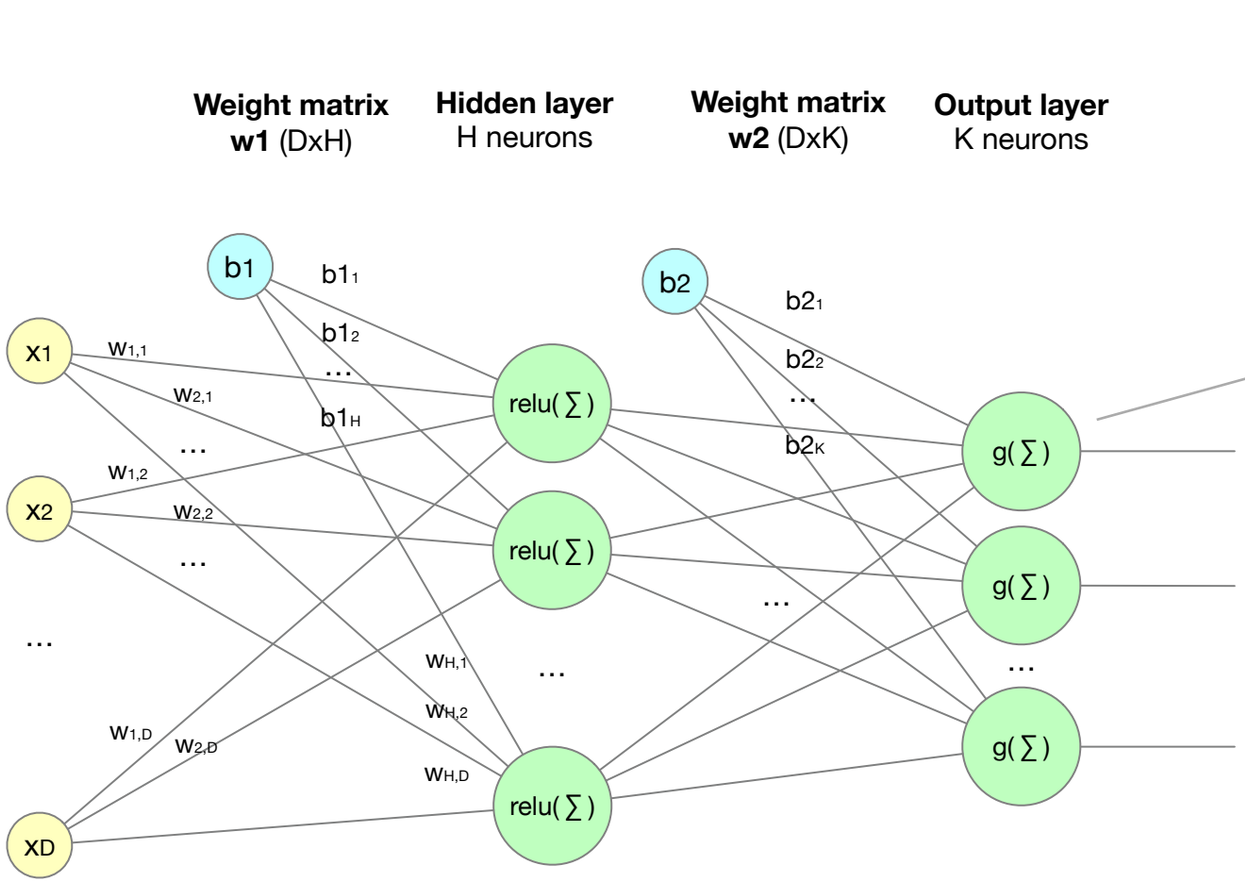


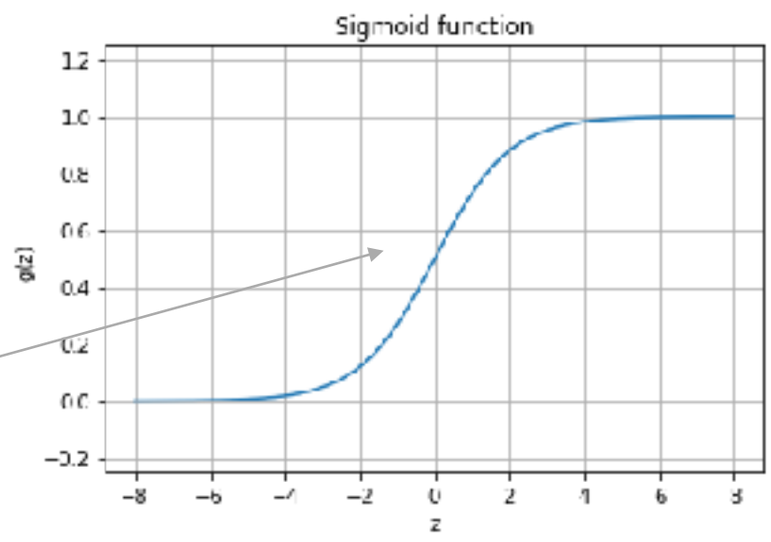
Which one to chose?

- If you have “strong” outliers that are the results of measurement errors or quirks
  - Do feature clipping then min-max rescaling
  - Ex: distribution of salary in a large corporate
- If you have “normal” outliers, i.e. not so extreme and not so frequent
  - Do z-norm that gives less importance to the few outliers
  - Ex: price of a laptop
- If you have no outliers and you know the upper and lower bounds
  - Do min-max rescaling
  - Ex: age of people
- If you have a “tail distribution”
  - Use log scaling
  - Ex: number of ratings for a movie
- ⚠️ The best normalization technique is one that empirically works well, so try new ideas if you think they'll work well on your feature distribution.
- Read https://developers.google.com/machine-learning/data-prep/transform/normalization

Normalisation of the output y

- We need to normalise the target output data when the mapping $\hat{y} = h(x)$ is limited to a given range
- Example: neural network architecture with sigmoid as activation function at the output layer ⇒ min-max rescaling of the targets





The output will be, in this case between 0 and 1. Therefore, the targets {y} should be in the same range.

How would you encode these categorical data?

Activity

1. The grades of a student: A+, A, B+, B, C+, C
2. The degree a person has: High school, Bachelor, Master, PhD.
3. Department where a person works: Finance, Human resources, IT, Production.
4. The city where a person lives: Lausanne, Bern, Zurich, Geneva, etc.

Encoding categorical data - ordinal case

- First question to ask: is the order important or not?
- If yes, use an encoding that preserves the order
  - For example, map the grades of a student A+, A, B+, B, C+, C to decreasing numerical values.

Grade	Encoding
A+	6
A	5
B+	4
B	3
C+	2
C	1

Encoding categorical data - nominal case

1-hot encoding - association of 1 input for 1 category and encoding a given category by setting to 1.0 the associated input and 0.0 elsewhere.

- 1-hot encoding is used in most situations where there is no relationship or order or proximity between categories
  - Example of the department where a person works: Finance, Human resources, IT, Production, Management.

Category	Financ	HR	IT	Prod	Mngmt
Input 1	1	0	0	0	0
Input 2	0	1	0	0	0
Input 3	0	0	1	0	0
Input 4	0	0	0	1	0
Input 5	0	0	0	0	1

For ex:
…
if x is 'Prod':
    x_num = [0,0,0,1,0]
if x is 'Mngmt':
    x_num = [0,0,0,0,1]
…

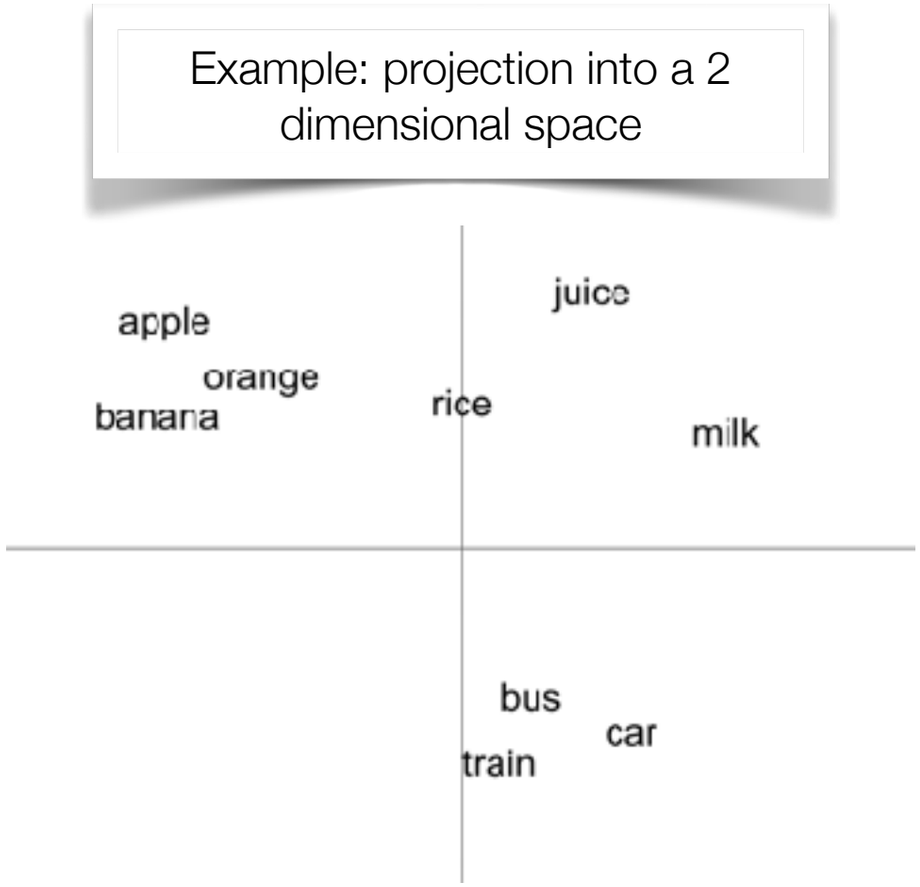
See also https://developers.google.com/machine-learning/data-prep/transform/transform-categorical

Encoding of text data

- Character level:
  - 1-hot encoding can be used when the system takes 1 char at a time (e.g. with recurrent neural networks)
- Word level:
  - 1-hot encoding is applicable when the vocabulary is low, e.g up to a few thousand words
  - Most of the time, a **word embedding** technique is used

Word embedding is a projection of a word into vectors of real numbers, i.e. a mapping of a space where each word has its own dimension to a space that is of lower dimensionality.

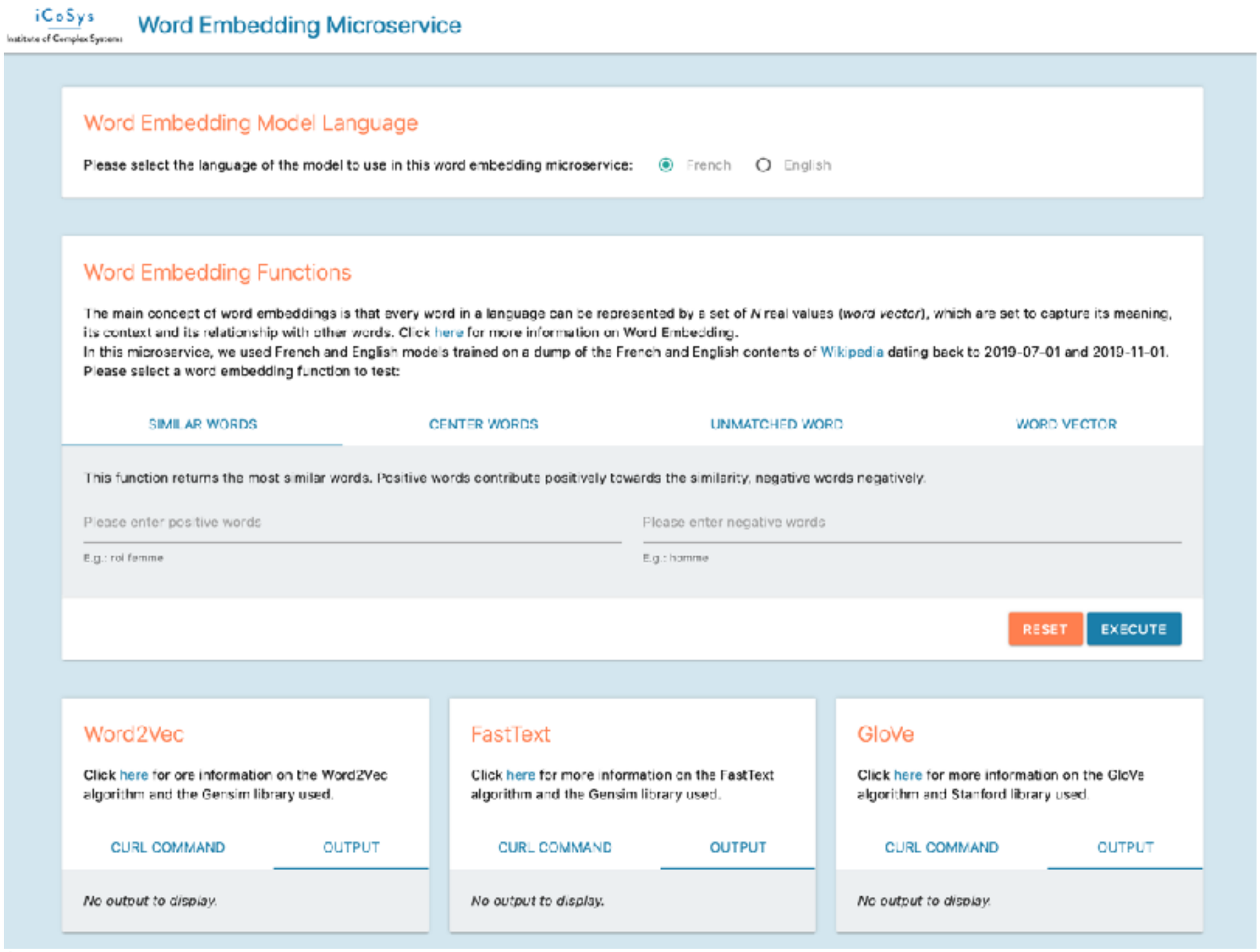
Encoding of text data



Word2vec relates to models producing word embedding into space with **contextual similarity** or words, i.e. words that share common context are located in close proximity.

- Two original papers from Google - Mikolov et al. (2013): https://arxiv.org/pdf/1301.3781.pdf, and https://arxiv.org/abs/1310.4546
- Trained on large corpora such as wikipedia
- Can be seen as a **feature extraction** of words
- How to treat unseen words (proper names, conjugations, etc): subword modelling with FastText

Showcase - https://icoservices.k8s.tic.heia-fr.ch/word-embedding/showcase

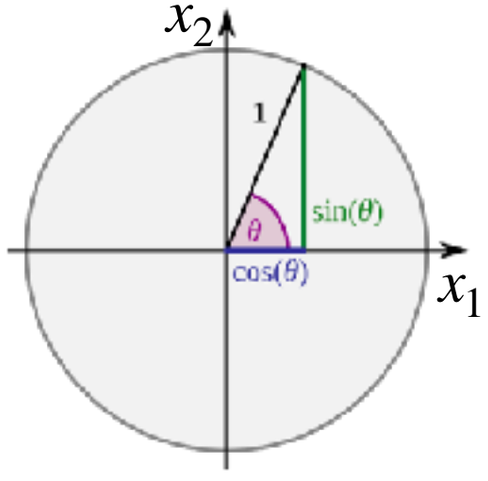


Beware of some discontinuities behaviour

- Encoding latitude and longitudes:
  - Travelling west makes you jump from -180° to +180° a bit after Hawaï
  - Longitude values at the poles are distorted (represent less km than at the equator)
  - If distance between two points is the feature you are looking for, then convert latitude and longitude, e.g. into XYZ coordinates or compute distance in km
  - Bucketing latitude and longitude is also an option to transform into categorical values, see https://developers.google.com/machine-learning/data-prep/transform/bucketing

Beware of some discontinuities behaviour

- Encoding time of the day
  - Numerically: not good to jump from 12h to 1h, from 60s to 1s
  - 2 possibilities:
    - Compute h = time in the day, e.g. 15.95 means 15h57.



Map h into two new features x_1 and x_2 with

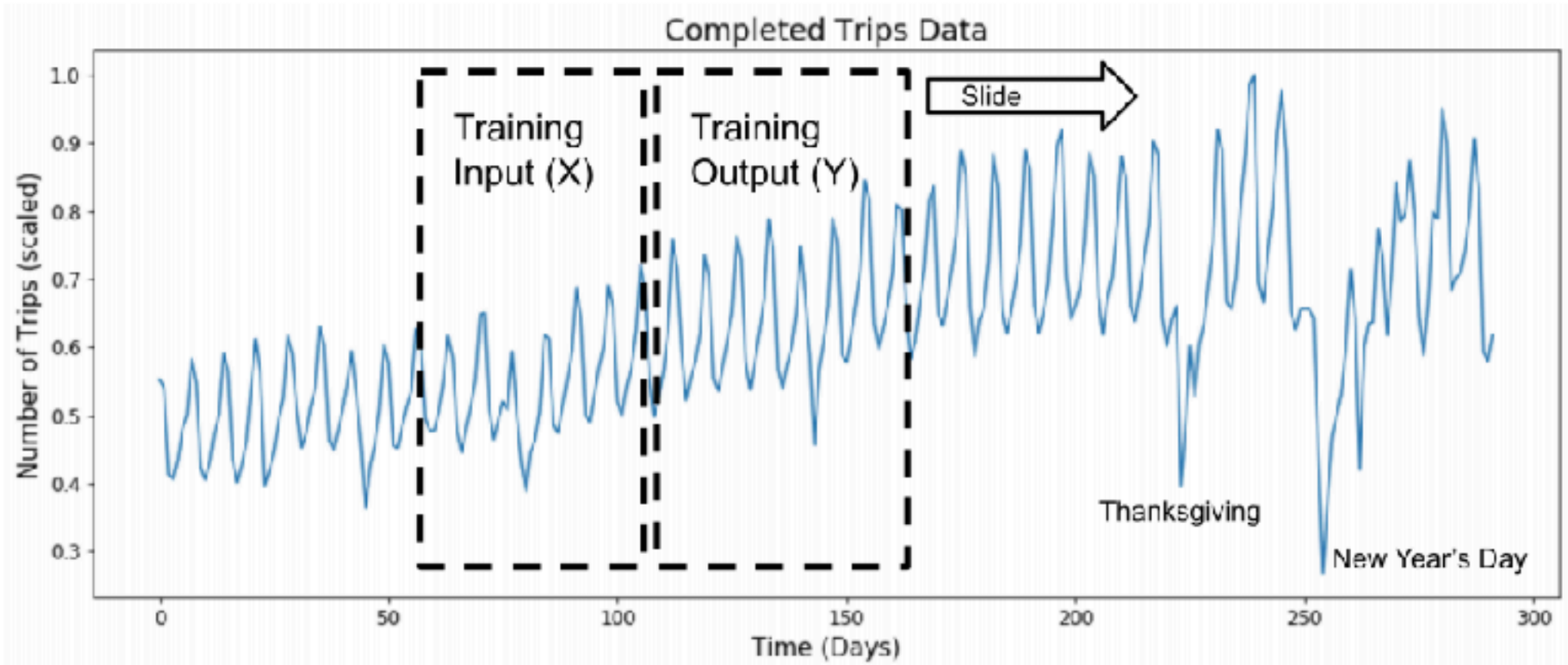
$$x_1 = \cos(\frac{2\pi h}{24}) \text{ and } x_2 = \sin(\frac{2\pi h}{24})$$

  - Use categorical 1-hot encoding dividing the day into buckets, e.g. use 24 buckets if precision at the hour is enough

- Same thing for day number, week number, month number

For time series

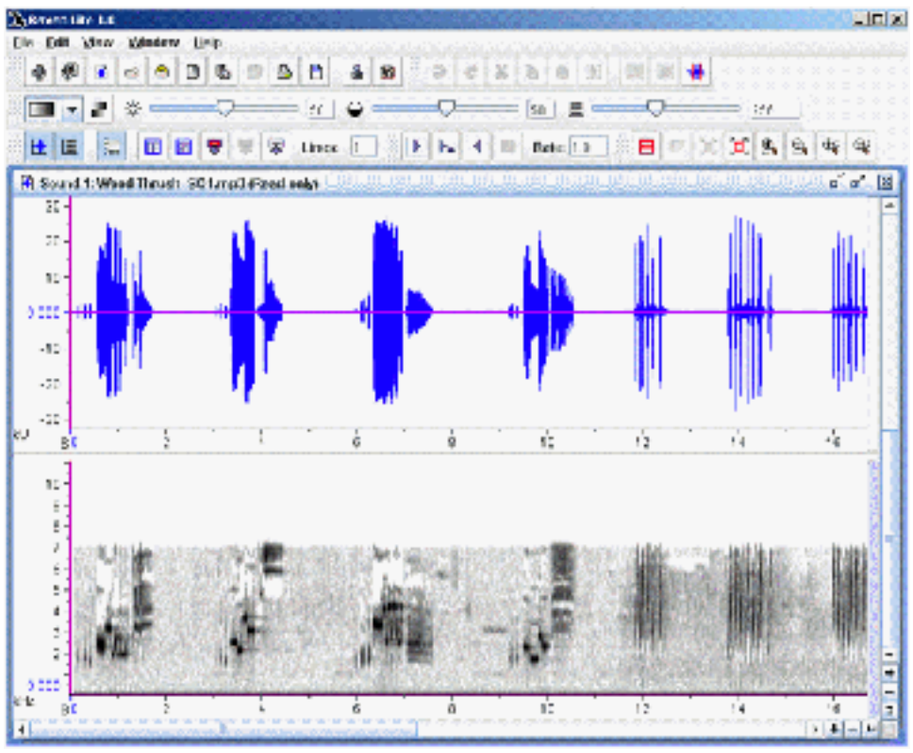
- Often: sliding window principle



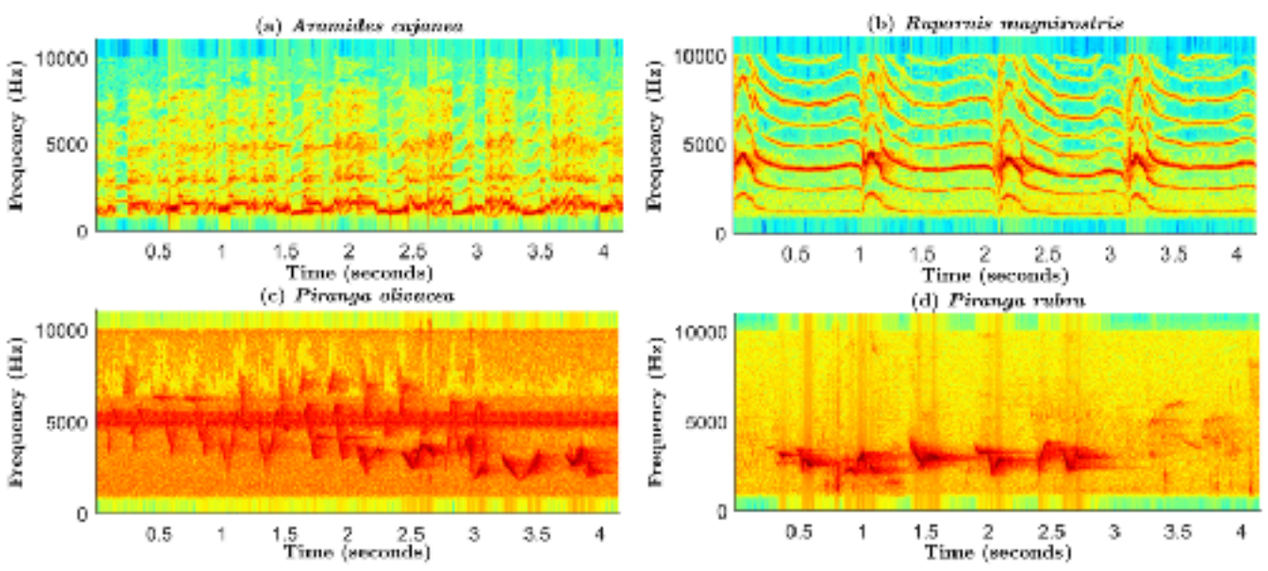
https://machinelearningmastery.com/lstm-model-architecture-for-rare-event-time-series-forecasting/

For time series

- Information can be in the frequency domain. Use Fast Fourier Transform (FFT) and spectrogram to discover in which frequencies the signal is evolving.



https://www.thayerbirding.com/gbna/39/TBS39Help/field_guide/spectrogram_raven_lite.htm



https://journals.plos.org/plosone/article/figure?id=10.1371/journal.pone.0179403.g007

5.3 The problem of overfitting

- Example
- Definition
- Factors triggering overfitting

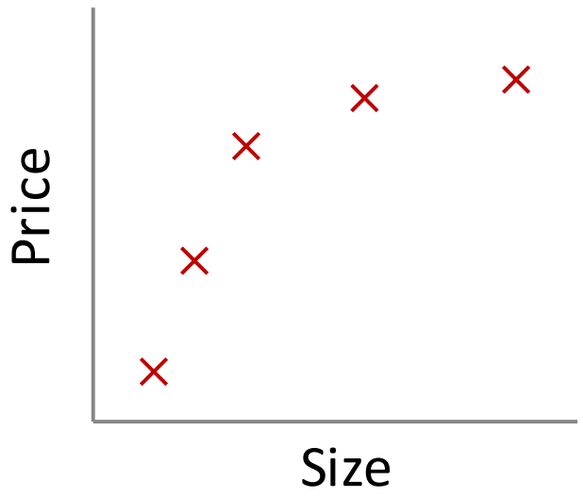
Model parameters vs hyper parameters

Model parameters - set of parameters θ of the mapping function $h_\theta(\mathbf{x})$. These parameters are learnt through model training on the training set.

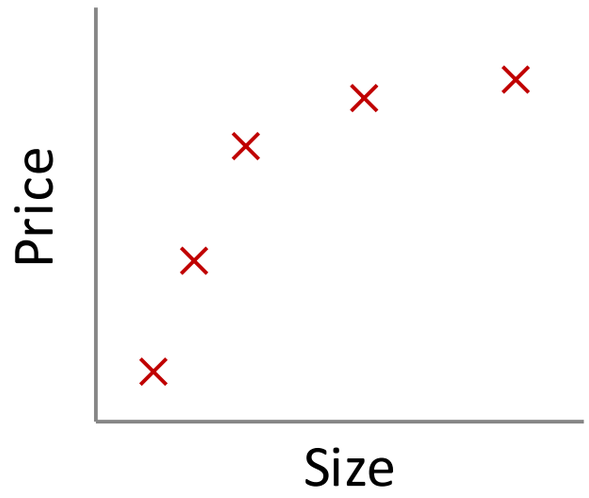
Hyper parameters - set of parameters defining higher level properties of the mapping function $h_\theta(\mathbf{x})$ and learning process. These parameters are also optimised inspecting their performance on data.

- Some examples of hyper parameters:
  - Order of polynomial in linear regression
  - Number of leaves or depth of a decision tree
  - Learning rate (in many models)
  - Number of hidden layers in a deep neural network
  - Number of neurons in a layer of the neural network
  - Number of clusters in a k-means clustering

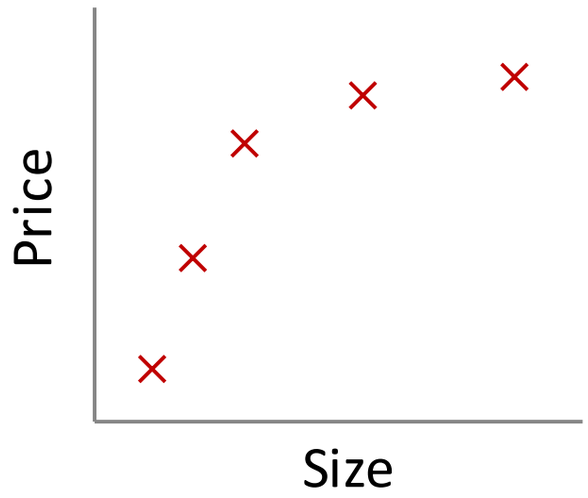
Linear regression example



$\theta_0 + \theta_1 x$

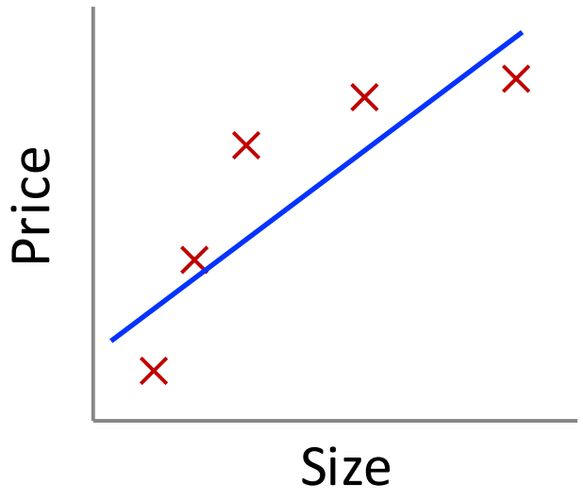


$\theta_0 + \theta_1 x + \theta_2 x^2$

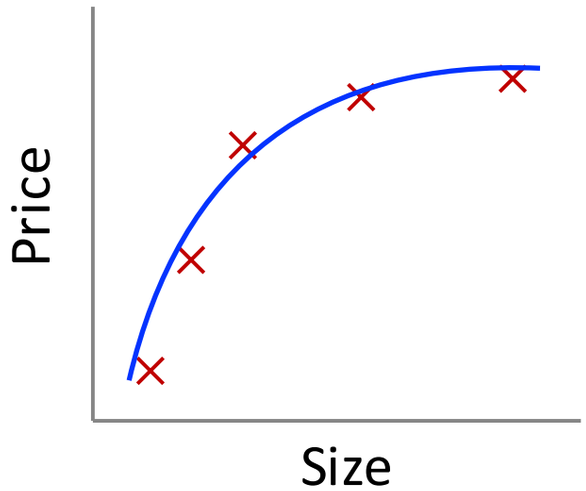


$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

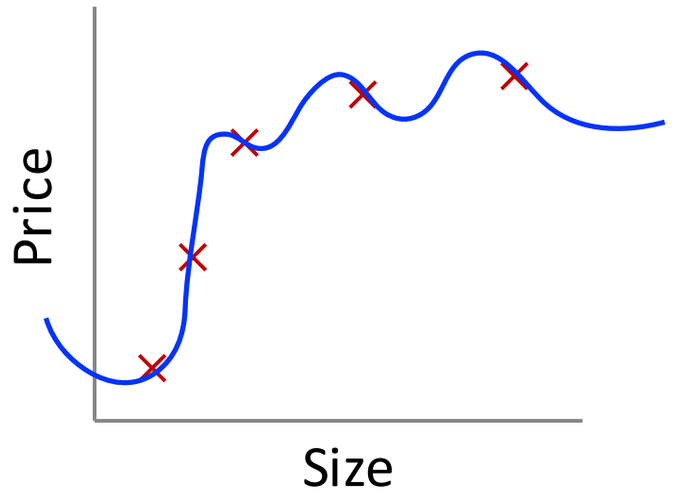
Linear regression example



$\theta_0 + \theta_1 x$



$\theta_0 + \theta_1 x + \theta_2 x^2$



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Underfit "high bias"

Strong "preconception" or "bias" of the hypothesis that the prices are going to vary linearly with the size

Good fit "just right"

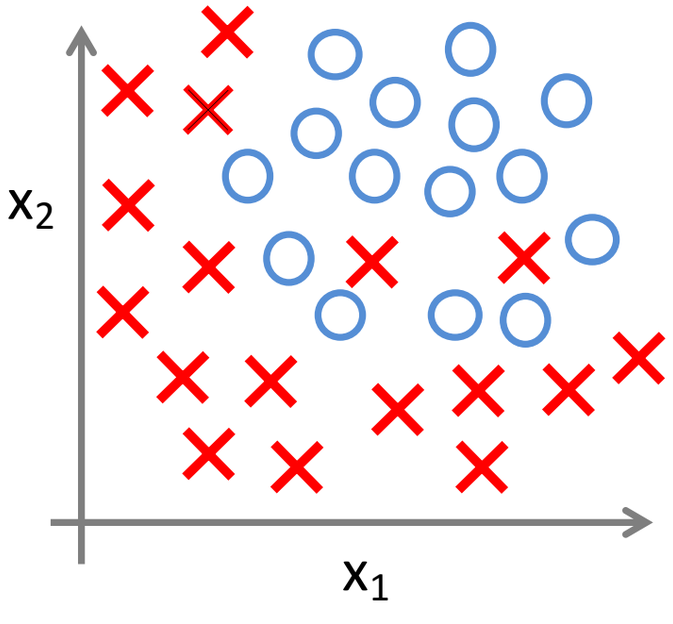
The hypothesis seems to fit just right the underlying data structure.

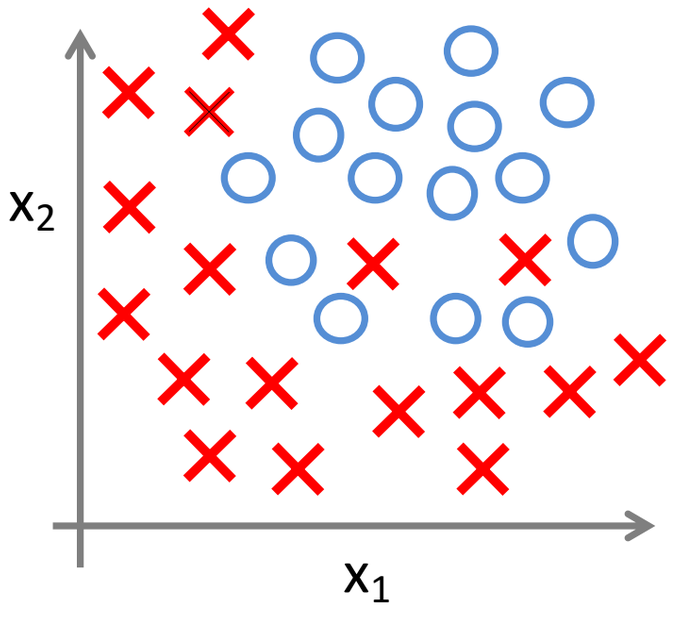
Overfit "high variance"

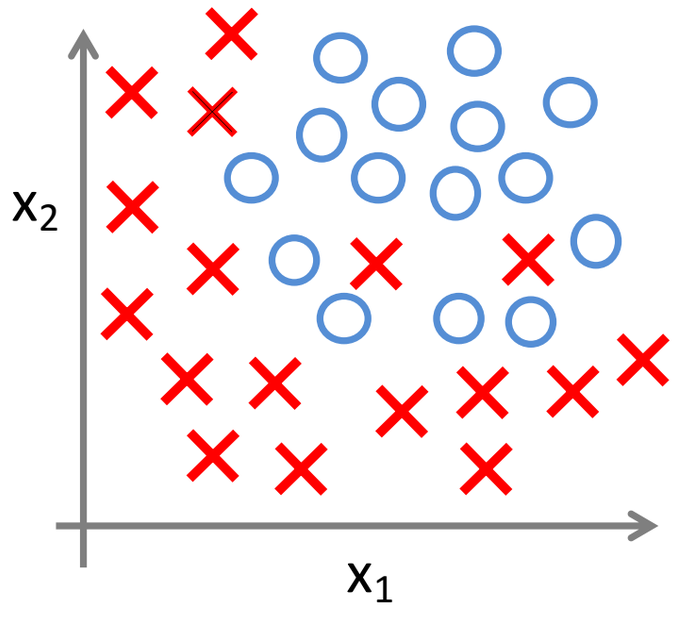
The curve passes through all the training data and then fits perfectly the training data. It is like if the hypothesis is too strong regarding the number of data.

Classification example - logistic regression

Logistic Regression will be seen soon







$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

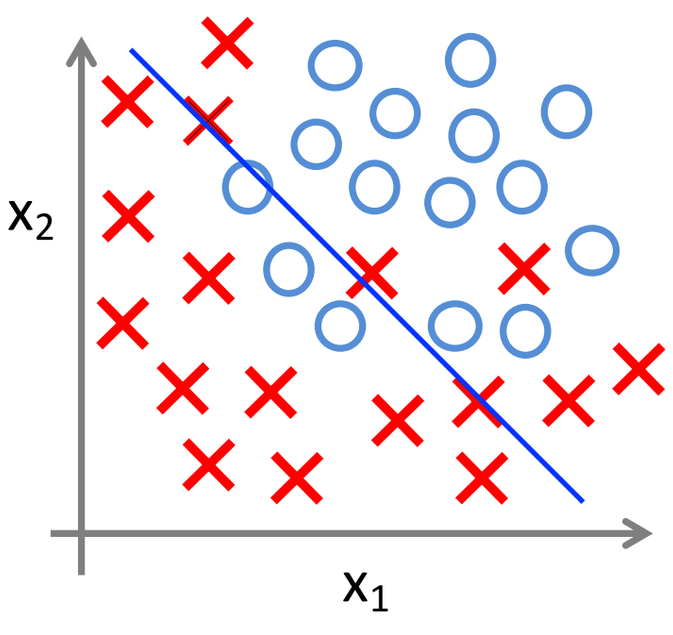
(g = sigmoid function)

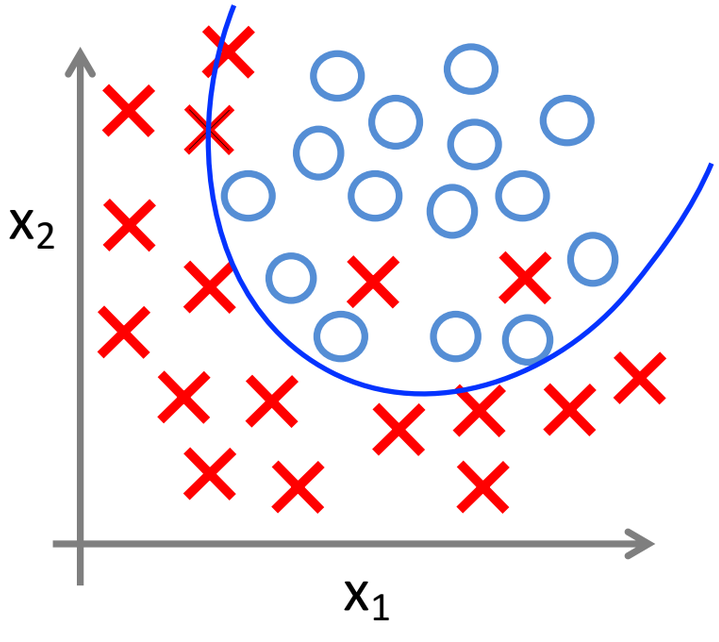
$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$

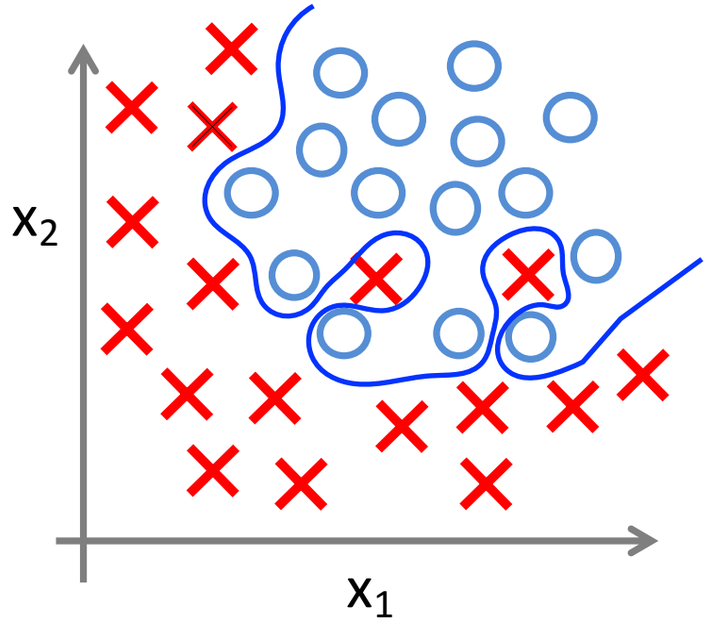
$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$

Classification example - logistic regression

Logistic Regression will be seen soon







$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(g = sigmoid function)

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$

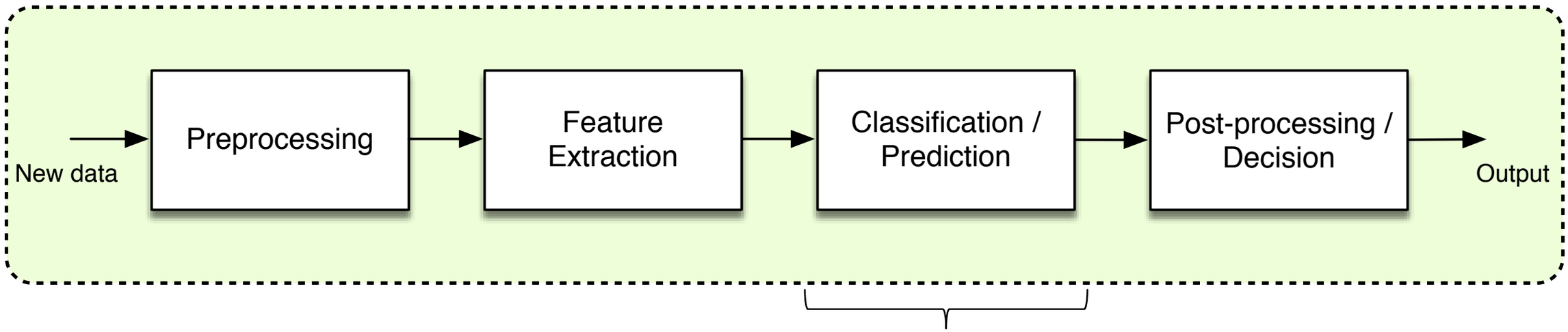
$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$

Underfit
"high bias"

Good fit
"just right"

Overfit
"high variance"

Overfitting definition



- This is mainly concerning the classification / prediction module

Overfitting - happens when the learned hypothesis fits the training set very well, but fails to generalise to new (unseen) examples.

$$J(\theta) = \frac{1}{2N}\sum_{n=1}^{N}(h_\theta(\mathbf{x}_n) - y_n)^2 \approx 0$$

“on the training set”, but …
how is J on the test set?

Overfitting happens when…

- The size of the training set is too small in comparison to the dimension of the input variable
- The number of parameters of the model is too large,
  i.e. the model is too “strong”, too “flexible”
- Example:

$x_1 =$ size of house
$x_2 =$ no. of bedrooms
$x_3 =$ no. of floors
$x_4 =$ age of house
$x_5 =$ average income in neighborhood
$x_6 =$ kitchen size
$\vdots$
x_{100}

With only 50 training samples.

Remark. With an increasing number of dimensions, it is also difficult to visualise the overfitting. On the other hand it was easy to observe it on the example of slide 33, when there was only 1 dimension.

Remark. An option to address overfitting is to select which features to keep (manually) or to let an algorithm determine it for us (see later in the class the chapter about “dimensionality reduction”)

5.4 Model selection process

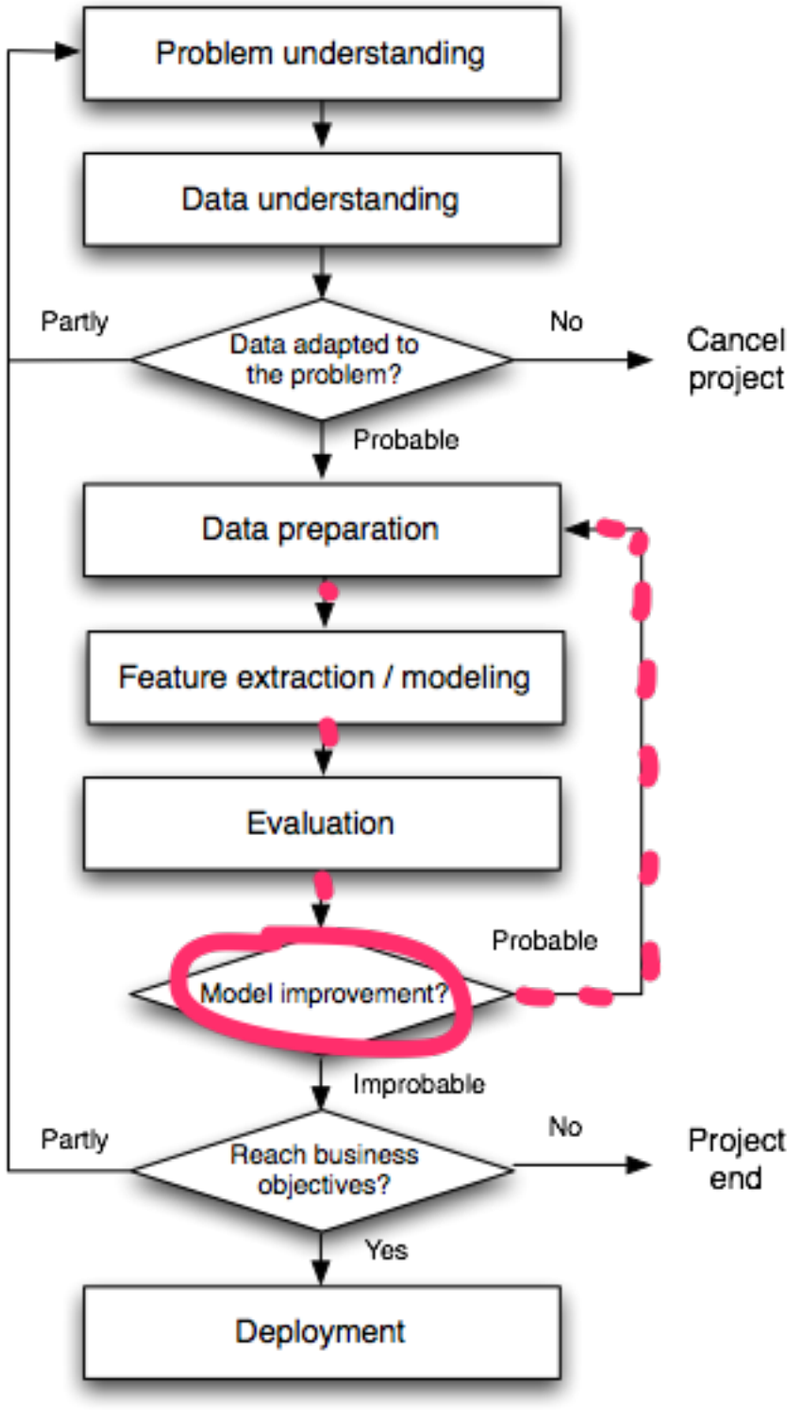
Evaluation on train set

Evaluation on test set

Model selection using cross-validation set

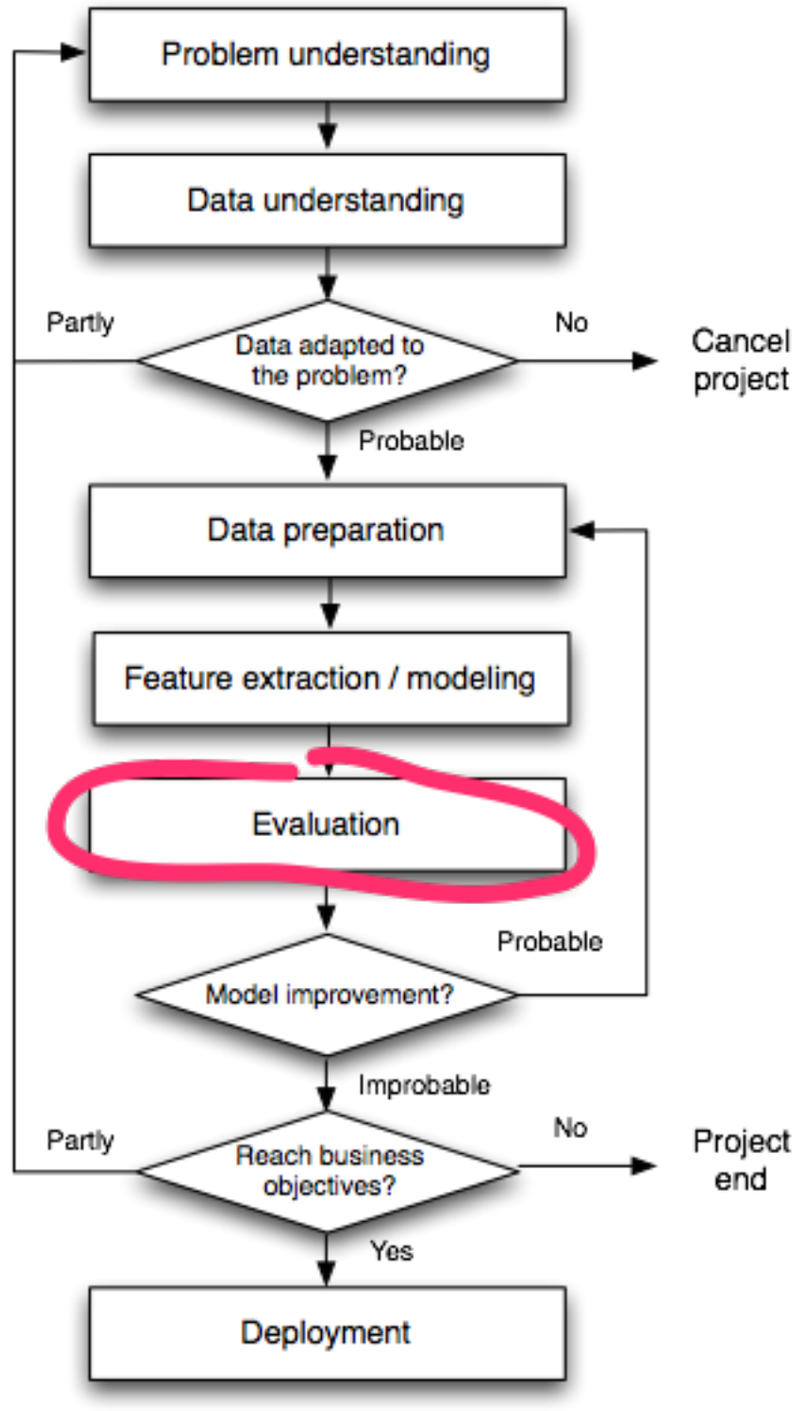
Model selection

- Our goal is to make the system as good as possible.
- We have many options:
  - better features
  - better (stronger) models
    - in a given family of models, we often have **hyper parameters** that we need to tune
      - polynom order
      - number of neurons
      - …
  - better training procedure
    - strategies to set
      - the learning rate
      - the criteria to stop/continue learning
      - the initial parameter values
      - …
- We may run hundreds of training procedures and evaluations to find the best configuration



Model selection

- The idea is to select the model that will give us the best performance.
- The next question is how to evaluate system performance.
  This is not that easy to do if we want to be fair.

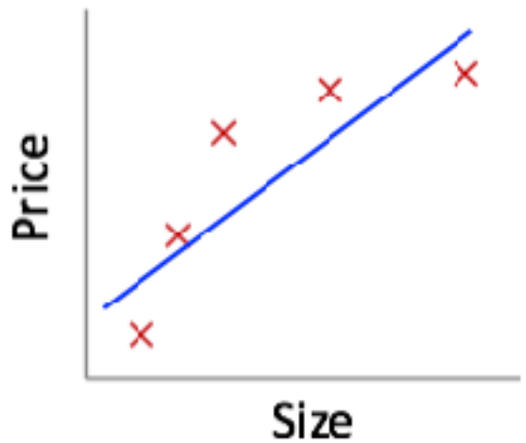


First rule - don’t rely on training set for the evaluation

- Evaluating the learning on the training set is **not** a good idea. As we have seen, overfitting may happen with a cost J going to zero on the training set.

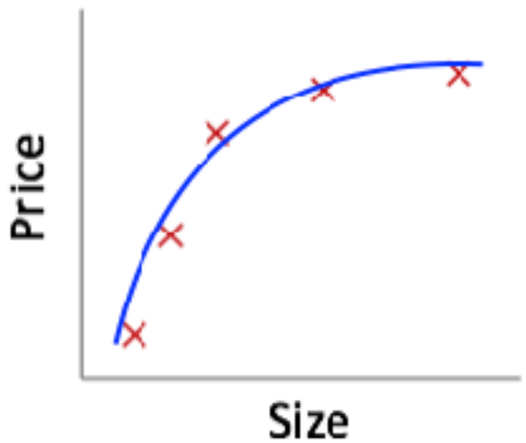
Poor generalisation on new data.

a



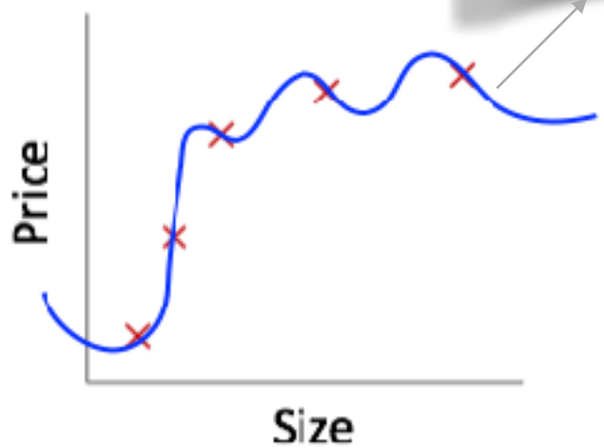
$\theta_0 + \theta_1 x$

b



$\theta_0 + \theta_1 x + \theta_2 x^2$

c



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

$$J(\theta_a) > J(\theta_b) > J(\theta_c)$$

Second rule - get a test set

Test set - set of independent and trustable data used to evaluate the performance of a trained model.

- By independent we mean disjoint to the training set.
- By trustable we mean
  - composed of data acquired in the *same conditions* as for the training set
  - including the same *characteristics* as in the training set (range of values, distribution, etc)
  - large enough to have *confidence* in the evaluation metric

Surface (m2)	Rent (CHF)
20	700
26	890
37	955
40	1100
45	1250
50	1350
57	1630
60	1650
60	1700
65	1800
57	1630
76	2060
80	2270
103	2550
120	2700
130	2800
142	3000
150	3250
…	…

Surface (m2)	Rent (CHF)
20	700
26	890
37	955
40	1100
45	1250
50	1350
57	1630
60	1650
60	1700
65	1800
57	1630
76	2060
80	2270
103	2550
120	2700
130	2800
142	3000
150	3250
…	…

Training set

For ex 70%

$$\{(\mathbf{x}_n^{train}, y_n^{train}); n = 1, \ldots, N_{train}\}$$

Test set

For ex 30%

$$\{(\mathbf{x}_n^{test}, y_n^{test}); n = 1, \ldots, N_{test}\}$$

The chosen test set is not **trustable**. Not in the same range of surface as in the training set.

Surface (m2)	Rent (CHF)
40	1100
142	3000
45	1250
130	2800
80	2270
76	2060
150	3250
60	1650
26	890
76	2060
57	1630
20	700
60	1700
120	2700
103	2550
37	955
65	1800
50	1350
…	…

We need first to **randomise** the full set of data and then split training/test set.

Training set

For ex 70%

$$\{(\mathbf{x}_n^{train}, y_n^{train}); n = 1, \ldots, N_{train}\}$$

Test set

For ex 30%

$$\{(\mathbf{x}_n^{test}, y_n^{test}); n = 1, \ldots, N_{test}\}$$

Procedure (not so good)

- Learn the parameters θ from the training set by minimising the training error $J_{train}(\theta)$

$$J_{train}(\theta) = \frac{1}{2N_{train}} \sum_{n=1}^{N_{train}} (h_\theta(\mathbf{x}_n^{train}) - y_n^{train})^2$$

- Compute the test set error

$$J_{test}(\theta) = \frac{1}{2N_{test}} \sum_{n=1}^{N_{test}} (h_\theta(\mathbf{x}_n^{test}) - y_n^{test})^2$$

- Amongst all the models you trained, select the one that minimises the test set error.

This procedure is not good if you do **intensive** model selection.

Example - linear regression

- Let's assume we want to select the best model amongst these, i.e. the best polynomial order

1. $h_\theta(x) = \theta_0 + \theta_1 x$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3$

$\vdots$

10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10}$

Train set	Test set
$J_{train}(\theta^{(1)})$	$J_{test}(\theta^{(1)})$
$J_{train}(\theta^{(2)})$	$J_{test}(\theta^{(2)})$
$J_{train}(\theta^{(3)})$	$J_{test}(\theta^{(3)})$
$\vdots$	$\vdots$
$J_{train}(\theta^{(10)})$	$J_{test}(\theta^{(10)})$

Example - linear regression

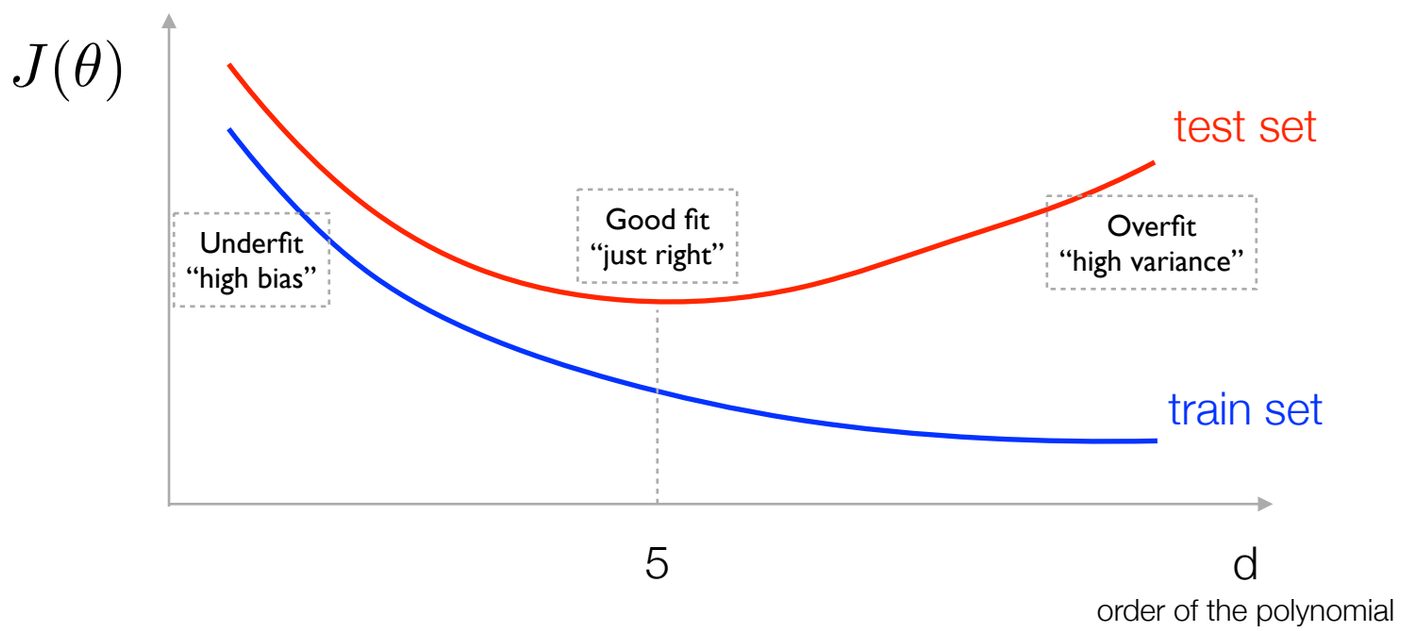
- Let's assume we want to select the best model amongst these, i.e. the best polynomial order

1. $h_\theta(x) = \theta_0 + \theta_1 x$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3$

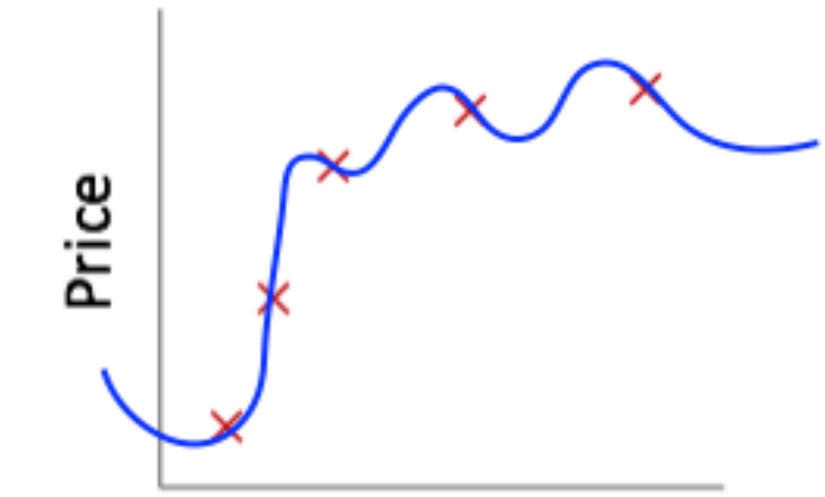
$\vdots$

10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10}$

Train set	Test set
$J_{train}(\theta^{(1)})$	$J_{test}(\theta^{(1)})$
$J_{train}(\theta^{(2)})$	$J_{test}(\theta^{(2)})$
$J_{train}(\theta^{(3)})$	$J_{test}(\theta^{(3)})$
$\vdots$	$\vdots$
$J_{train}(\theta^{(10)})$	$J_{test}(\theta^{(10)})$



Why did we say this procedure can be criticised? 2 slides ago



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

- As seen before, we know we can have overfitting on the training set
- Once parameters $\theta_0, \theta_1, \ldots, \theta_4$ are fit to the training set, the error measure on that data (the training error $J_{train}(\theta)$) is likely to be lower than the actual generalisation error.
- *Well, we can also have a kind of overfitting on the test set if we do intensive model selection*.

We can also have a “kind of” overfitting on the test set

- Let’s go back to this example:

1. $h_\theta(x) = \theta_0 + \theta_1 x$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3$

$\vdots$

10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10}$

- If we chose model number 5, we have actually “tuned” the order of the polynomial on the test set.
- The “hyperparameter” *polynomial order* has been fit on the test set
- How well does the model generalise? If we report the test set error $J_{test}(\theta^{(5)})$, it is likely to be an optimistic estimate

Third rule - get a cross validation set

Validation set - set of independent and trustable data used to **select models**, for example by selecting the best hyper-parameters of the models.

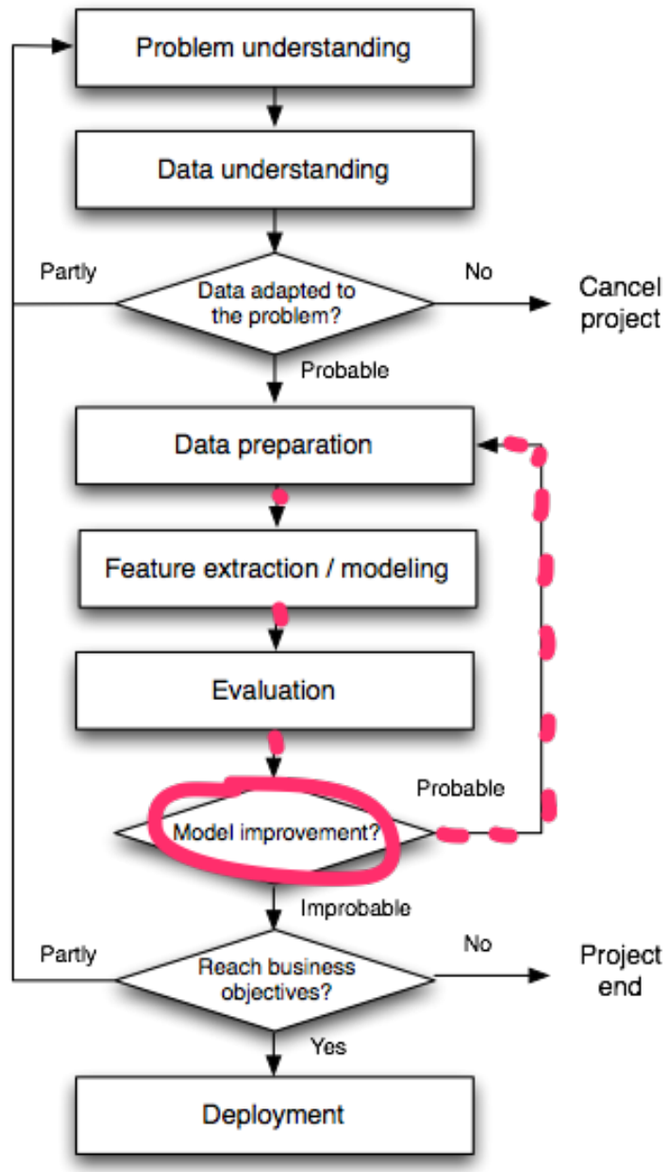
Same as before:

- By independent we mean disjoint to the training set.
- By trustable we mean
  - composed of data acquired in the *same conditions* as for the training set
  - including the same *characteristics* as in the training set (range of values, distribution, etc)
  - large enough to have *confidence* in the evaluation metric
- In the end we now have three sets:
  1. Train set: to train the model
  2. Validation set: to select the best model among the different training we did
  3. Test set: to evaluate the performance
  - 1 and 2 are used many times, 3 is used as few as possible

Summary: model improvement and evaluation procedure

Divide your data set into three independent and trustable data sets: the **training** set, the **validation** set and the **test** set.

- The train set is used to learn the parameters of the model.
- The validation set is used for model improvement, for example tuning the hyperparameters.
- The test set is used to report model performance.



Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

Really bad

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

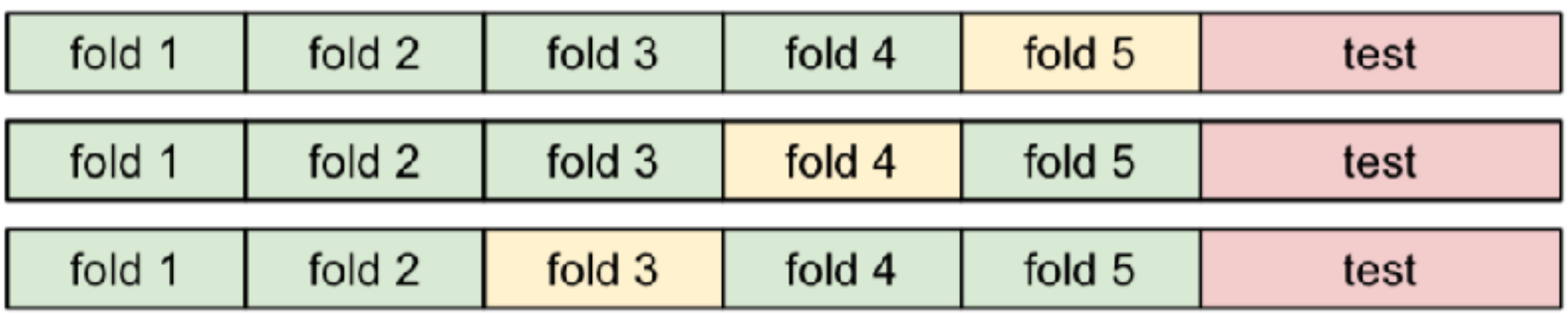
Ok-ish

but not so recommended

Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better

Idea #4: **Cross-Validation**: Split data into **folds**, try each fold as validation and average the results



Surface (m2)	Rent (CHF)
40	1100
142	3000
45	1250
130	2800
80	2270
76	2060
150	3250
60	1650
26	890
76	2060
57	1630
20	700
60	1700
120	2700
103	2550
37	955
65	1800
50	1350
…	…

Don't forget to **randomise** the full set of data and then split training/cross-val/ test set.

Training set

For ex 60%

$$\{(\mathbf{x}_n^{train}, y_n^{train}); n = 1, \ldots, N_{train}\}$$

Cross-validation set

For ex 20%

Test set

For ex 20%

$$\{(\mathbf{x}_n^{test}, y_n^{test}); n = 1, \ldots, N_{test}\}$$

Example

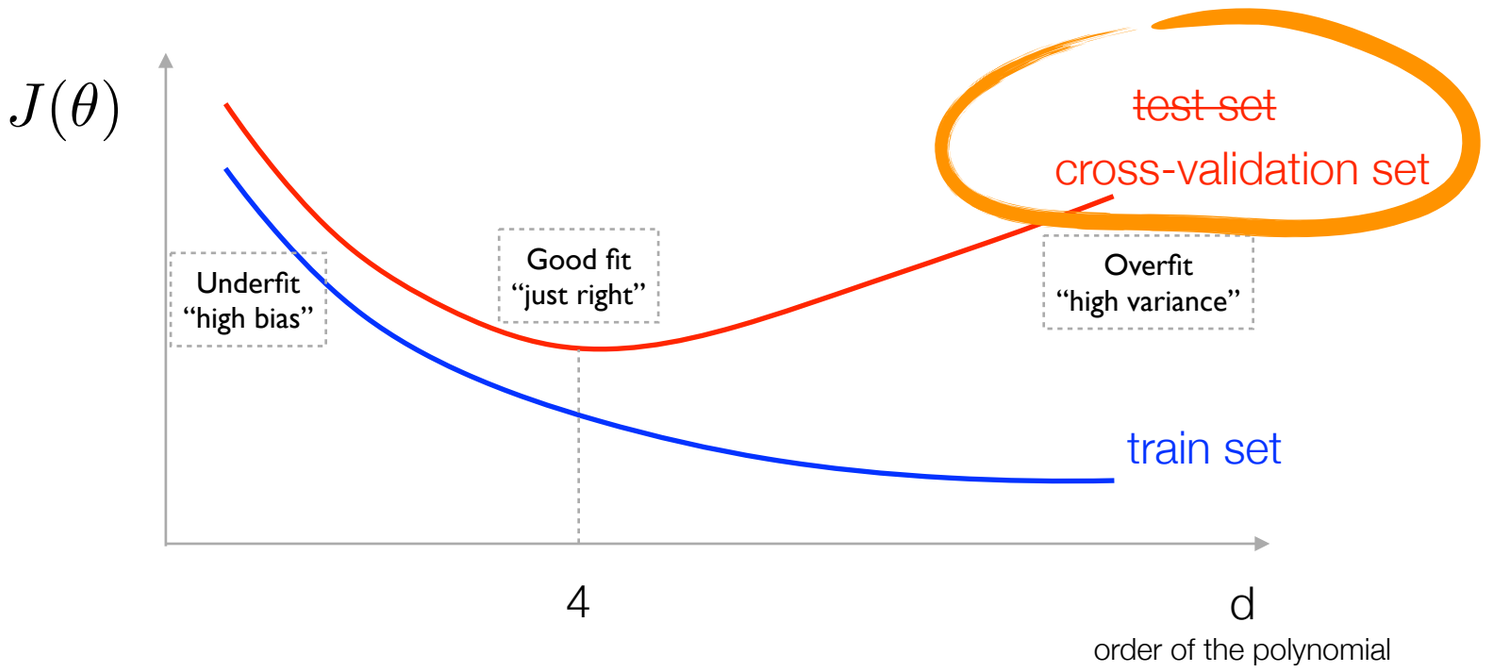
- Let's assume we want to select the best model amongst these:

1. $h_\theta(x) = \theta_0 + \theta_1 x$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3$

$\vdots$

10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10}$

Train set	cv set	Test set
$J_{train}(\theta^{(1)})$	$J_{cv}(\theta^{(1)})$	
$J_{train}(\theta^{(2)})$	$J_{cv}(\theta^{(2)})$	
$J_{train}(\theta^{(3)})$	$J_{cv}(\theta^{(3)})$	
$\vdots$	$\vdots$	$J_{test}(\theta^{(4)})$
$J_{train}(\theta^{(10)})$	$J_{cv}(\theta^{(10)})$	



- Select the model with degree 4 as it minimises the error $J_{cv}(\theta^{(4)})$ on the cross-validation set
- Report $J_{test}(\theta^{(4)})$, the generalisation error using the test set.

5.5 More data better model?

Learning curves

High bias

High variance

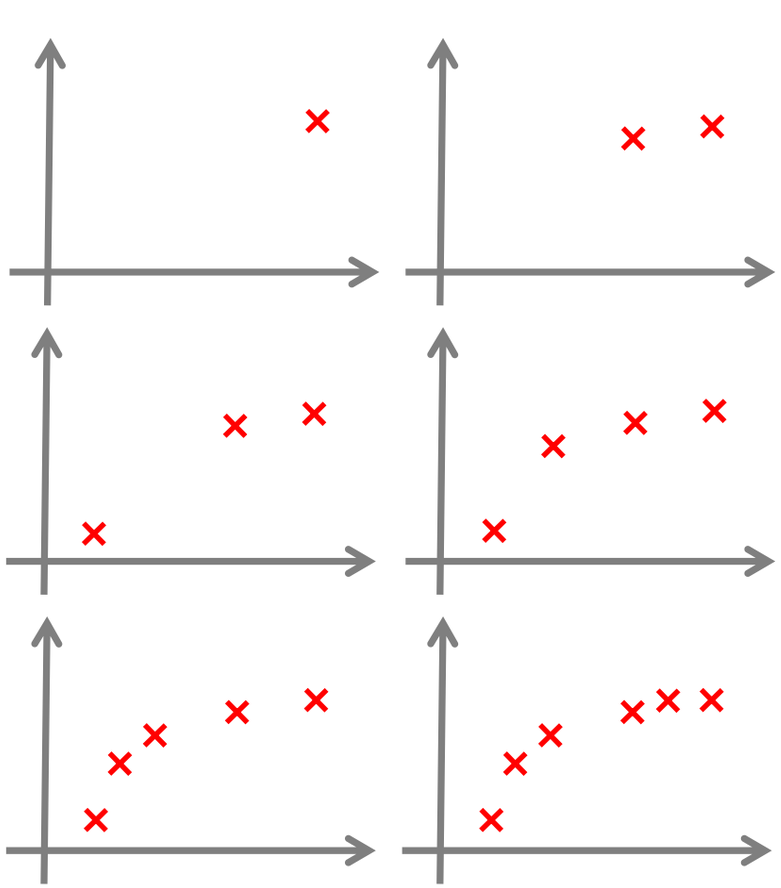
Should we always get more data?

- In many case yes, getting more data will help learning better models
  - more data allows to increase the number of features
  - more data allows to use more complex models
  - more data allows to reduce the risk of overfitting
- However, getting more data is sometimes costly…
- … and in some cases it does not help
- How can we detect if we have enough data? *(see next slides)*

Learning curves

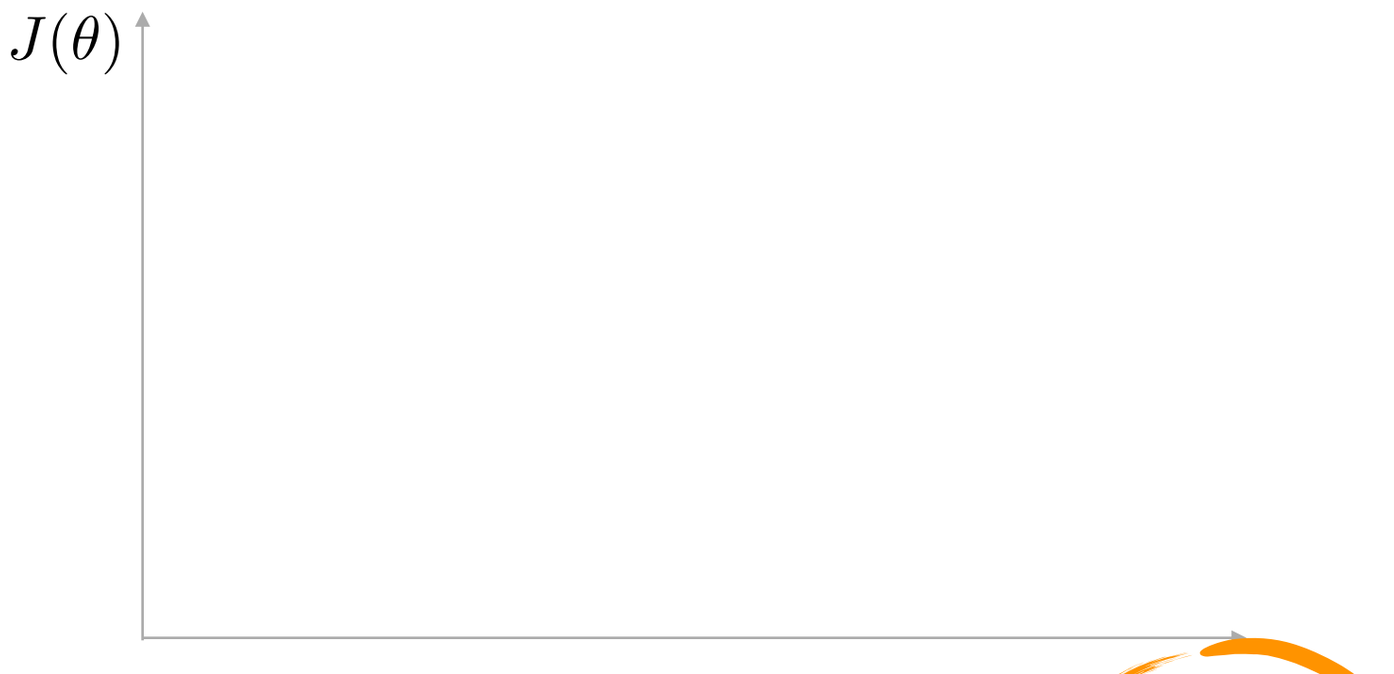
- The idea is to plot the evolution of the cost functions using increasing training set size

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$



$$J_{train}(\theta) = \frac{1}{2N_{train}} \sum_{n=1}^{N_{train}} (h_\theta(\mathbf{x}_n^{train}) - y_n^{train})^2$$

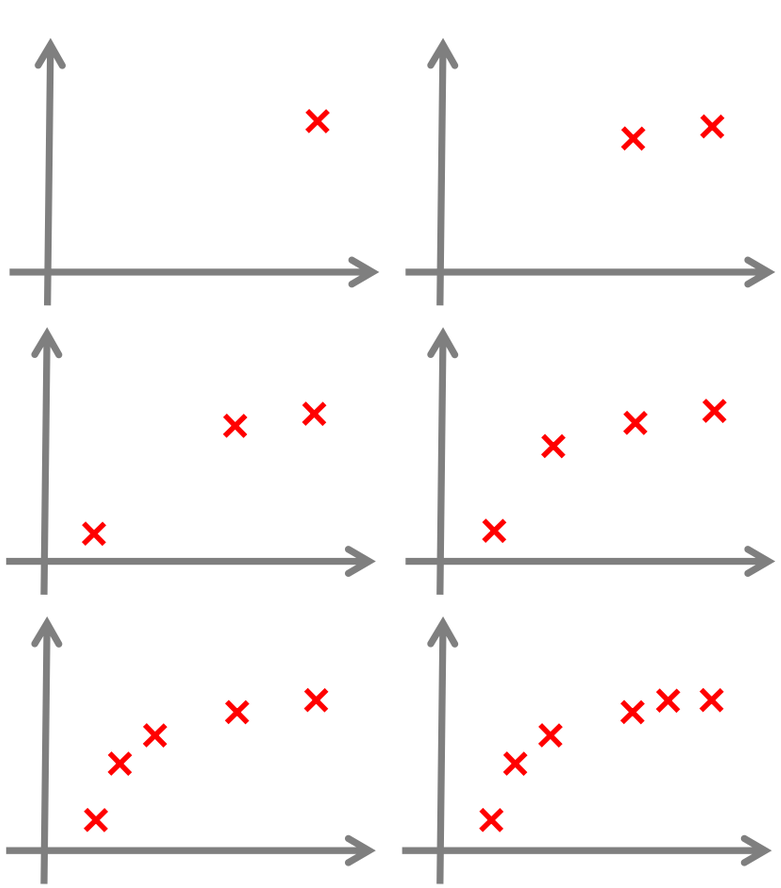
$$J_{cv}(\theta) = \frac{1}{2N_{cv}} \sum_{n=1}^{N_{cv}} (h_\theta(\mathbf{x}_n^{cv}) - y_n^{cv})^2$$



Learning curves

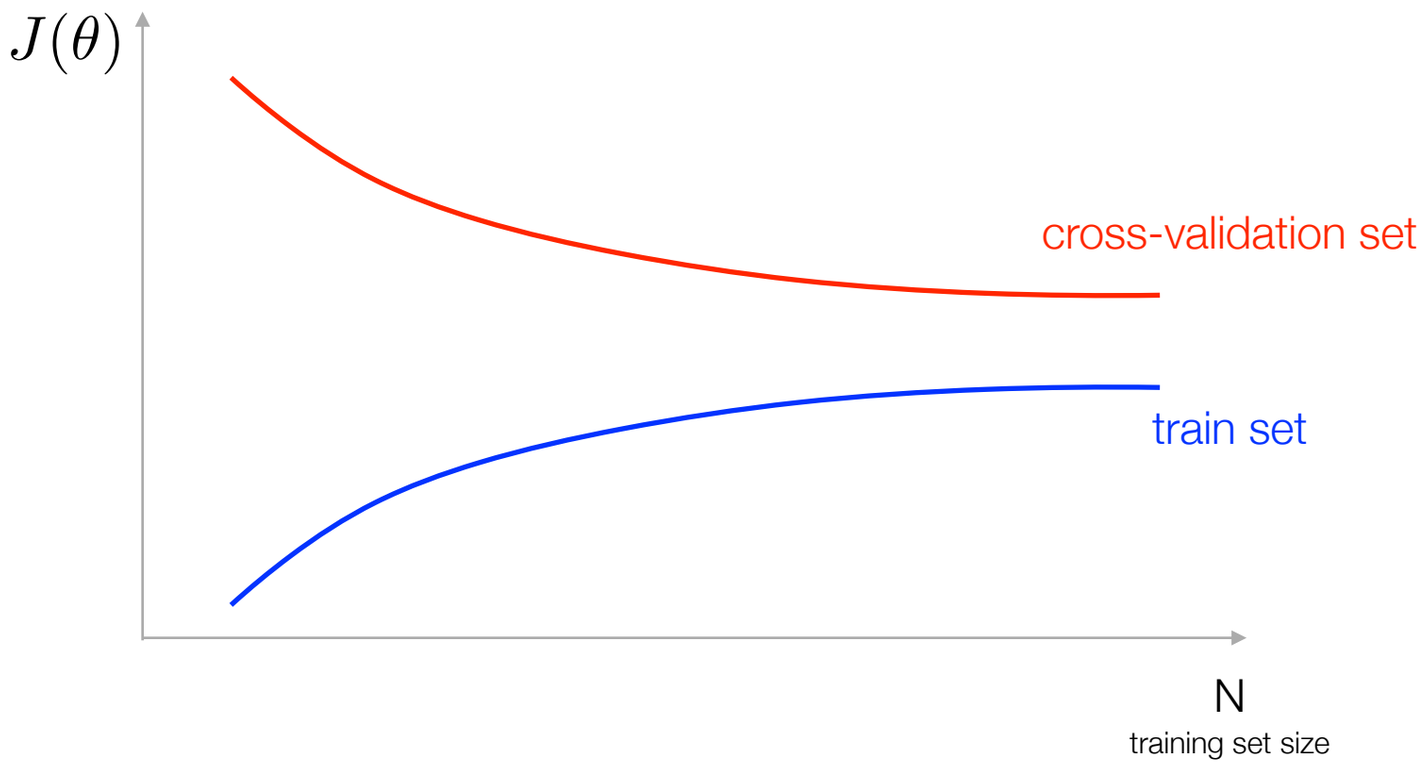
- The idea is to plot the evolution of the cost functions using increasing training set size

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

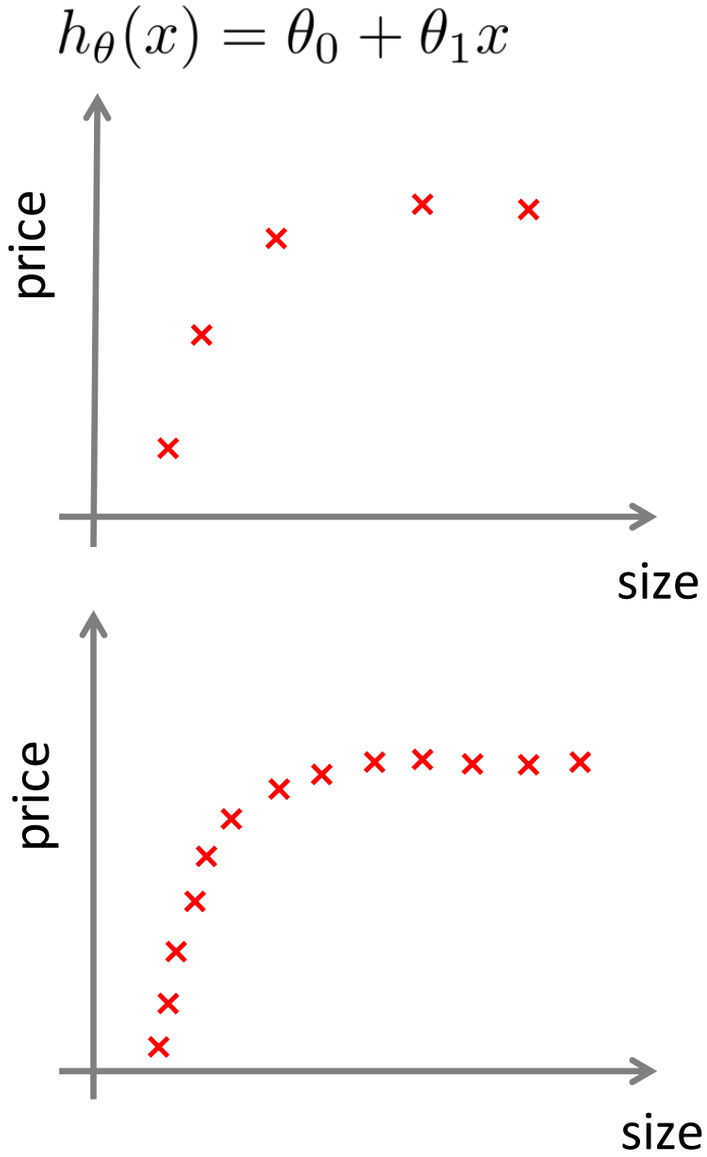


$$J_{train}(\theta) = \frac{1}{2N_{train}} \sum_{n=1}^{N_{train}} (h_\theta(\mathbf{x}_n^{train}) - y_n^{train})^2$$

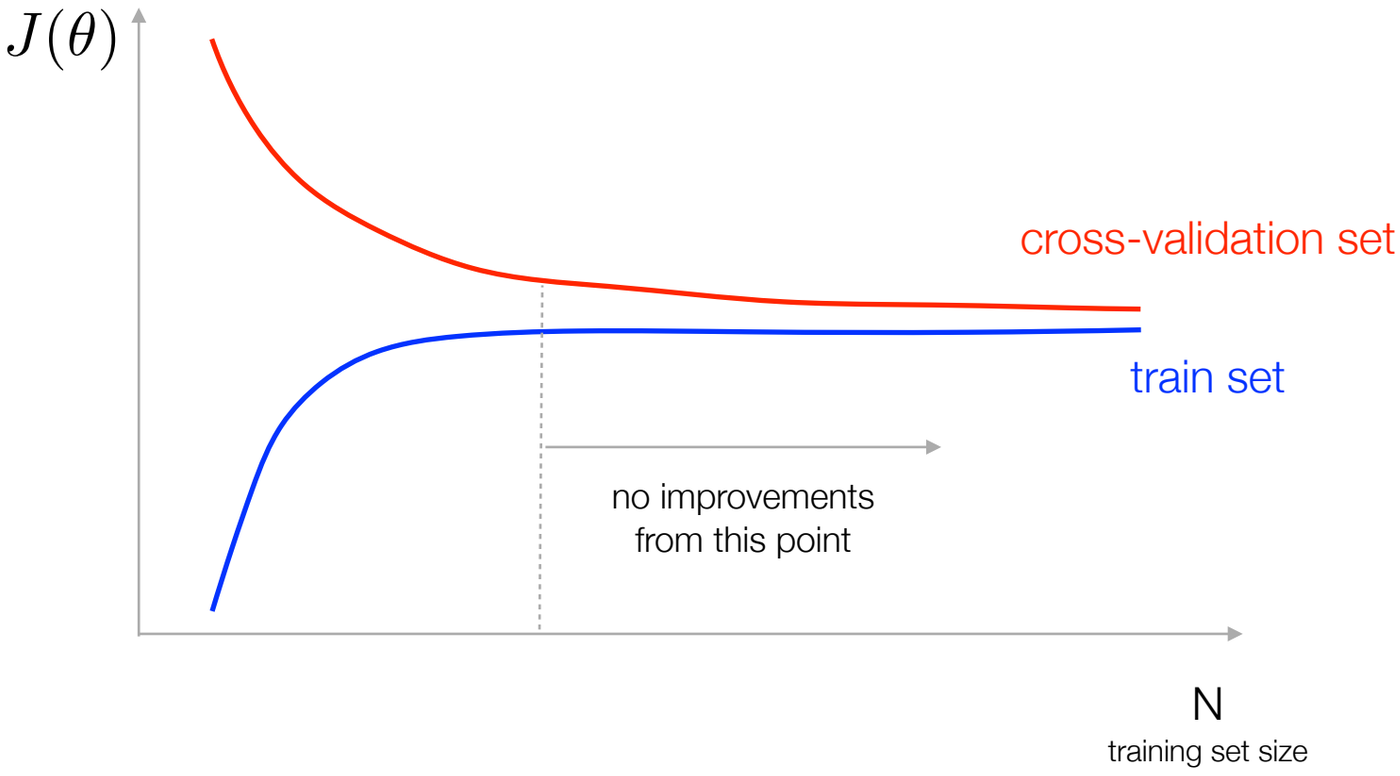
$$J_{cv}(\theta) = \frac{1}{2N_{cv}} \sum_{n=1}^{N_{cv}} (h_\theta(\mathbf{x}_n^{cv}) - y_n^{cv})^2$$

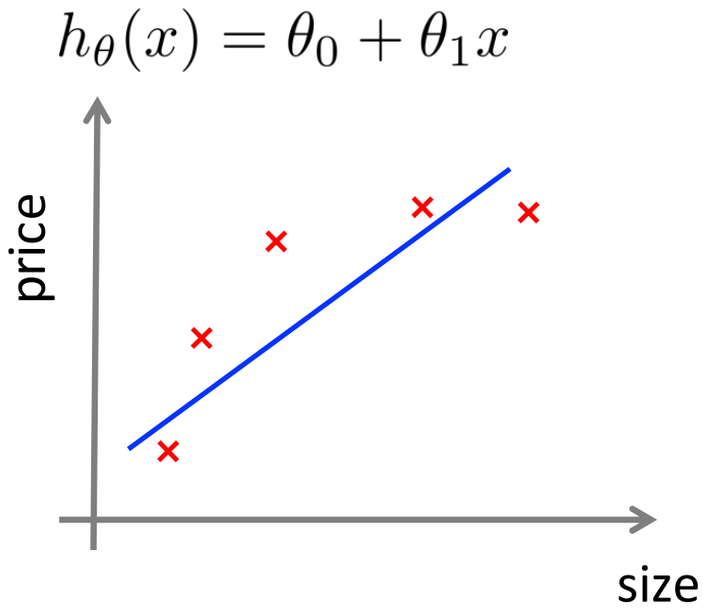


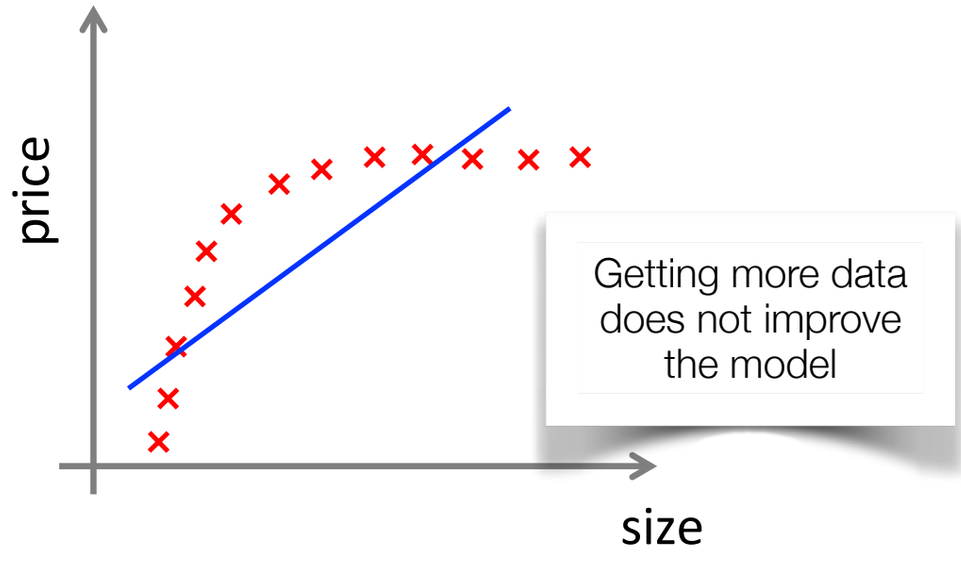
Underfitting situation - “high bias”



Underfitting situation - “high bias”



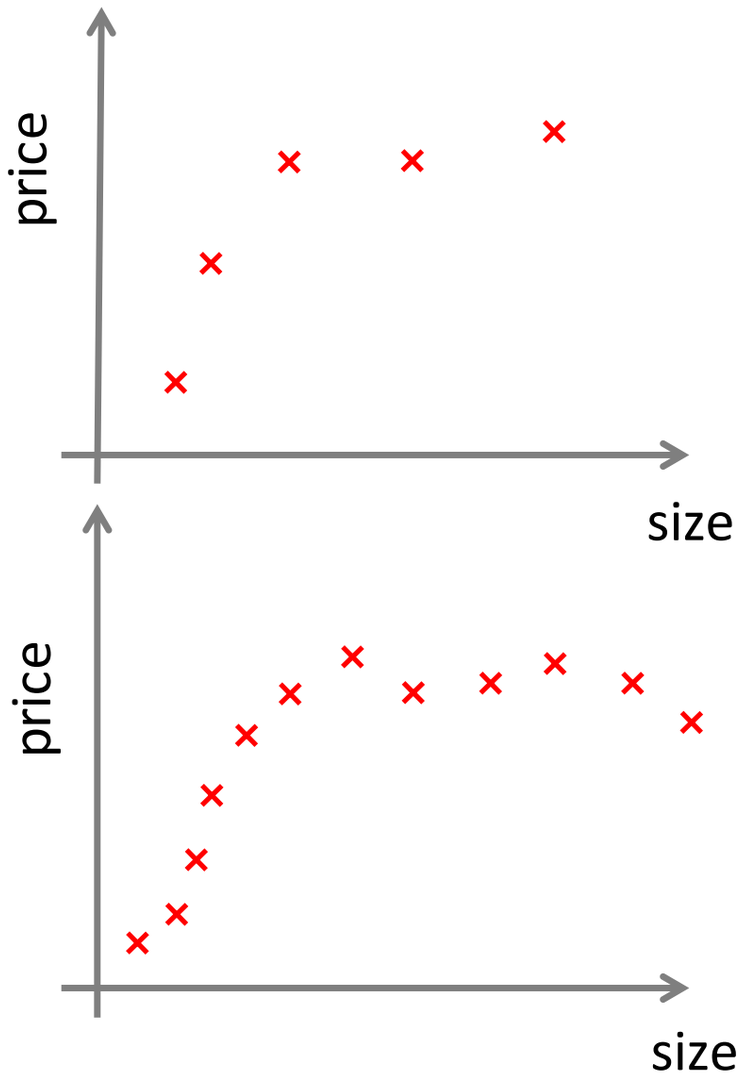




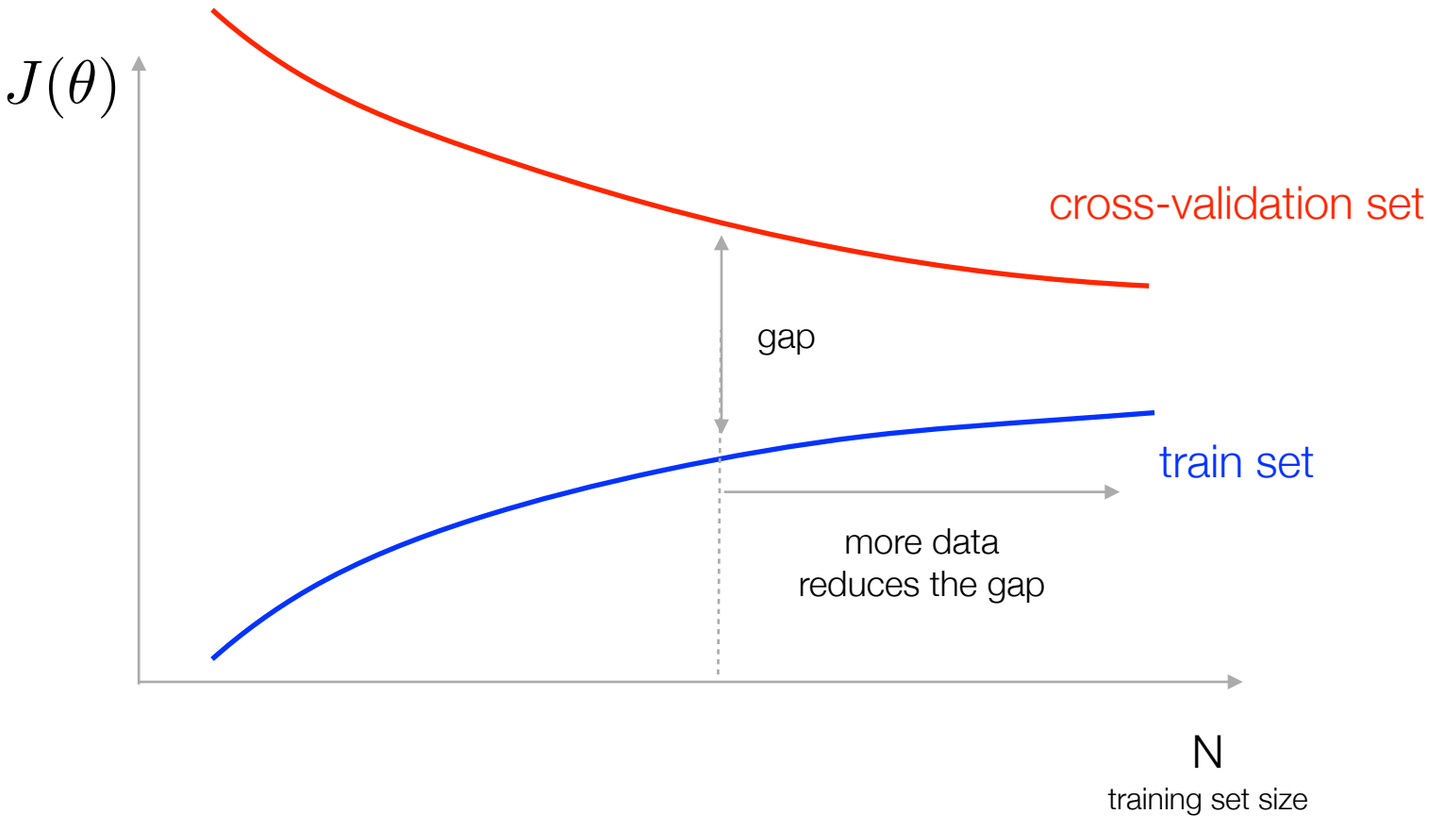
- If the model has high bias, getting more data will not help.
- “when the difference between the blue and red curve is small, then having more data for that model will not help”

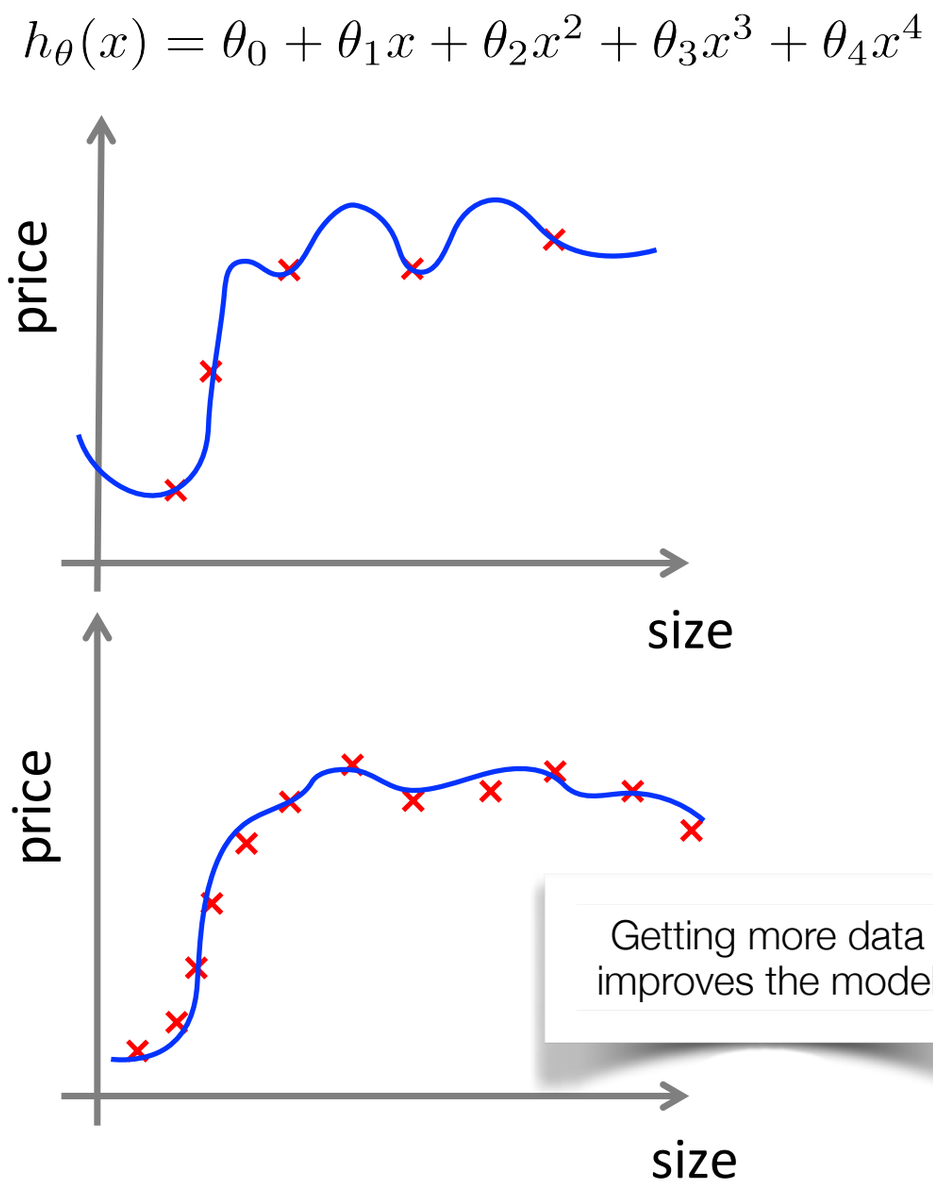
Overfitting situation - “high variance”

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

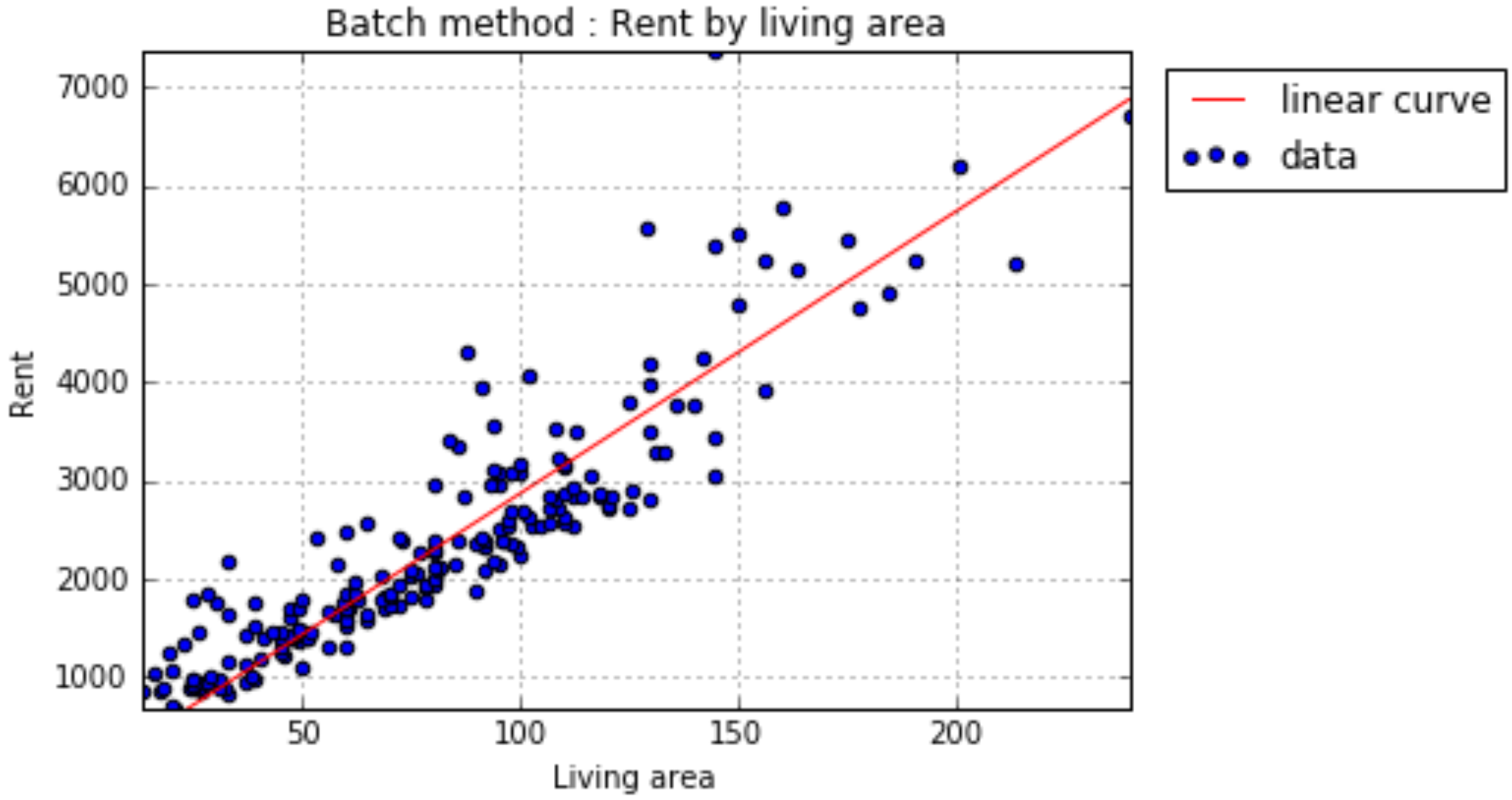


Overfitting situation - “high variance”



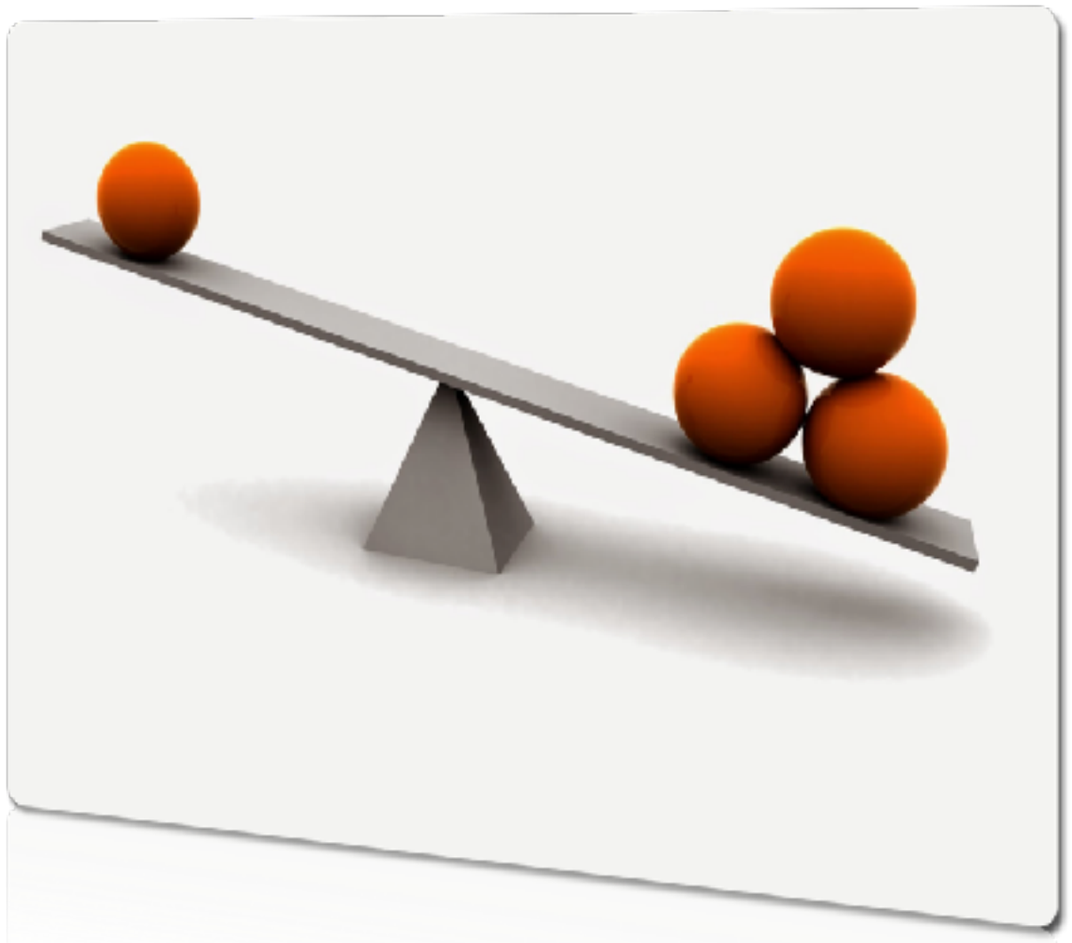


- If the model is overfitting, getting more data will probably help
- “when the difference between the blue and red curve is large, then having more data for that model will help avoiding overfitting”



- Looking at the problem of last week, we probably are facing a high bias problem with a linear regression model. Getting more data will probably not help a lot.

Other problems



Unbalanced training set

Inappropriate data

Unbalanced training sets

A few samples of sharks

Many examples of tuna

- Many algorithms are sensitive to unbalanced training sets, especially the *discriminative* ones. They tend to give (naturally) more weights to the most represented class.
  - SVM, neural networks, …
  - Such models actually model intrinsically the *priors* according to Bayes formula (see previous chapters)
- Some viable solutions:
  - resample the classes to make the training set balanced
  - give more importance to the cost on the less represented class

Inappropriate data

- In some problems, the data is just inappropriate to build a decent system
  - The data do not explain *enough* the predicted value
    - Ex: prediction of house price using the age of the landlord
  - The data is very *noisy*
    - Ex: speech recognition for beer ordering in a crowded bar
  - The ground truth is *not reliable*
    - Ex: a sickness prediction system where the diagnostics given by the doctors are unsure
  - The quantity of data is not large enough

Conclusions

- Data preparation
  - We need numerical data
  - Beware of outliers, beware of discontinuities
  - Data normalisation techniques: z-norm, min-max rescaling, clipping, log scale
  - Encoding categorical values: ordinal encoding, 1-hot encoding
  - Encoding text: word embedding, word2vec
- Procedure for machine learning:
  - Split your data into 3 sets
    - **training set**: to train the model parameters
    - **cross-validation set**: to optimise the hyper-parameters
    - **test set**: to report the generalisation error
- Badly performing models
  - Data in different scales: try to normalise your data
  - Underfitting models - High bias situation
    - Try getting more features
    - Try more complex models
  - Overfitting models - High variance situation
    - Try using less features
    - Try getting more data
  - Think again about your data set: unbalanced situation? appropriate data?

References

- Coursera, Machine learning, Andrew Ng, Stanford University, https://www.coursera.org/course/ml