

Data preprocessing for machine learning

Feature generation from data

Data preprocessing for machine learning

Interesting read: The Bitter Lesson by R. Sutton

<http://www.incompleteideas.net/IncIdeas/BitterLesson.html>

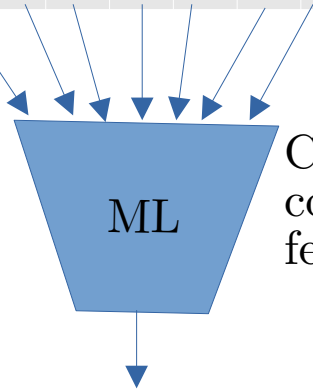
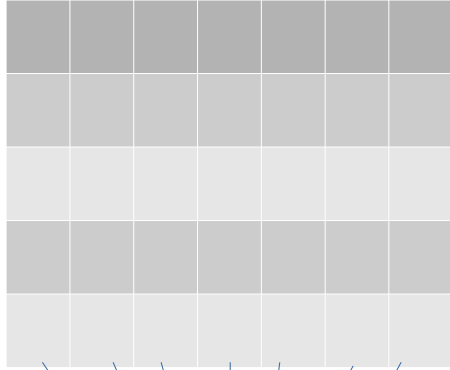
Data & Features

Table of values:

Matrix

Features

Samples



Complex
combination of
features

Output

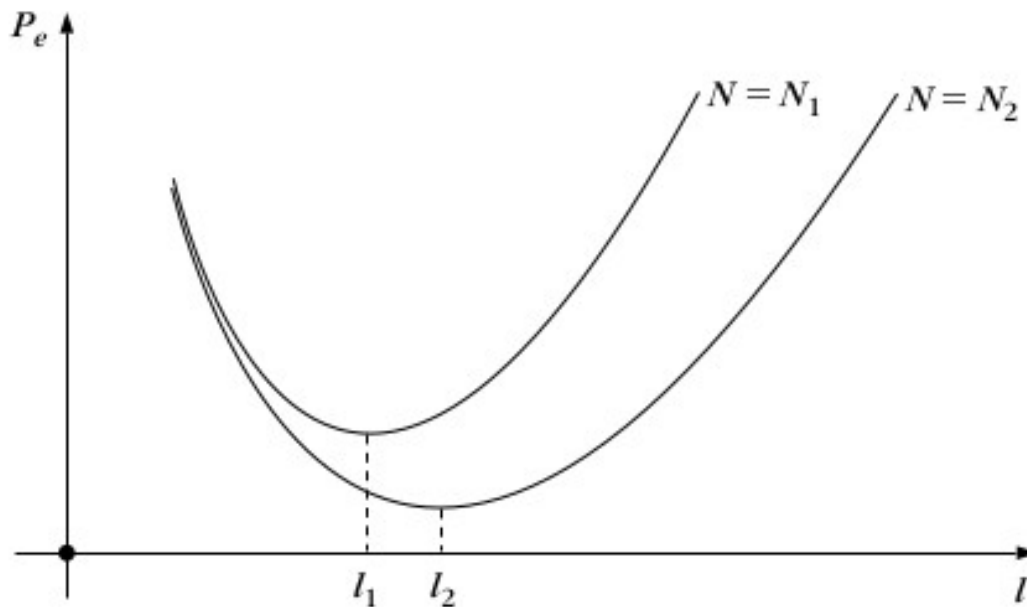
Goal: Make the learning easier, more robust, faster

Select among:

Good, bad, noisy, unrelated,
correlated... features

Features and information

Probability of error for the ML model



$N_1 < N_2$
Nb of samples

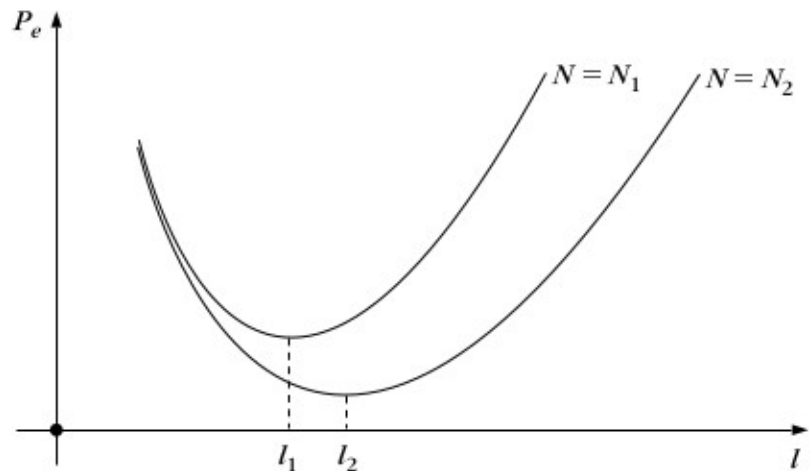
Nb of features

→
Increase of
information

→
Overwhelming
• information masked
• coincidences appear

Features and information

Probability of
error for the
ML model



We want to increase the quality of the data & possibly reduce the number of features.

How?

- Some methods rely only on the data
- Many methods use expert knowledge (not in the dataset):
 - Use apriori knowledge or inductive bias

Data & Features

$$X = \begin{pmatrix} x_1(1) & x_2(1) & \dots & x_N(1) \\ x_1(2) & x_2(2) & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & x_N(K) \end{pmatrix}$$

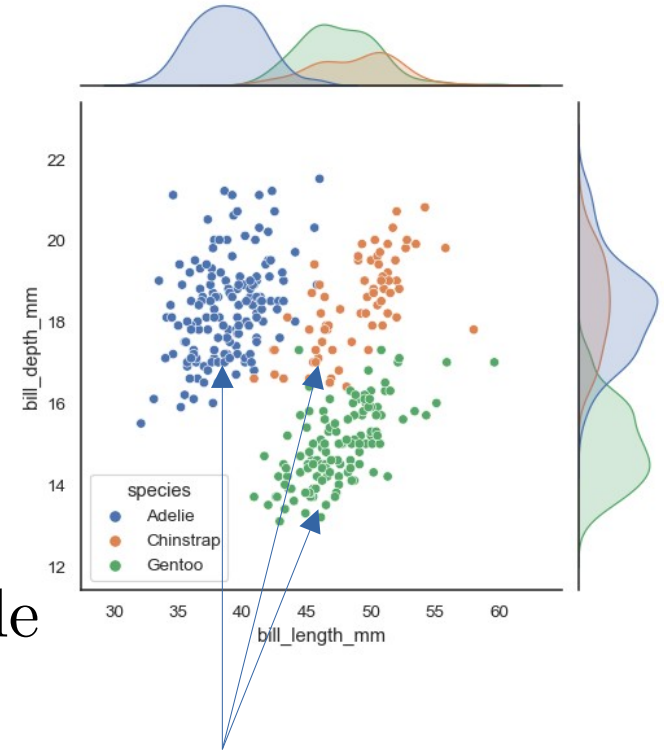
Features

Samples

Mathematical model:

Sample: one realisation of a random variable

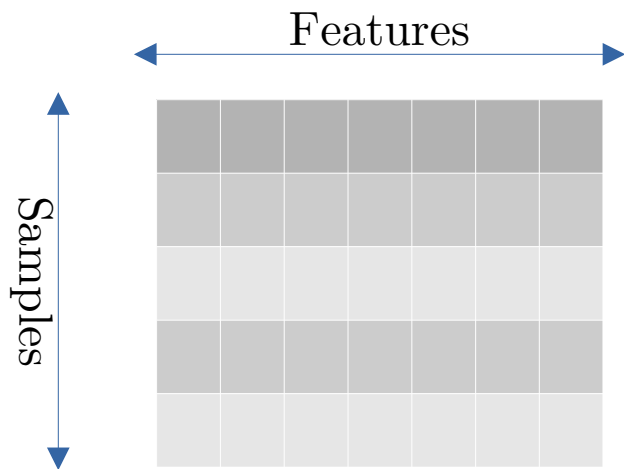
More samples \rightarrow more accurate statistics



Random samples from 3 different distributions?

Recap:

- We need as many samples as possible
- Several features, but not too many
- Features containing information



Pre-processing

Cleaning and filtering

How to get good quality data?

- Remove outliers
- Normalize data
- Fill in missing values
- More advanced transformations

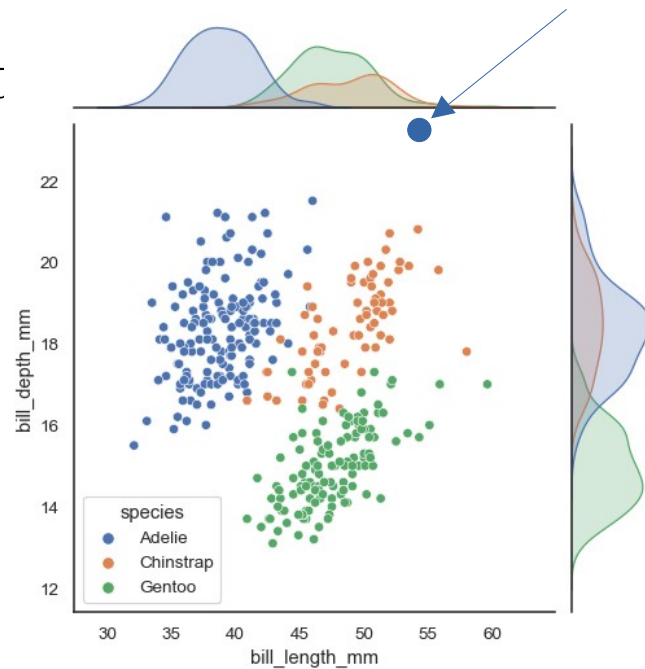
Use your knowledge of the data

- Future directions*: Self-supervised learning

* beyond the scope of this course

Removing outliers

- Outlier or out-of-distribution point
- a data point that differs significantly from other observations.



Impact of outliers

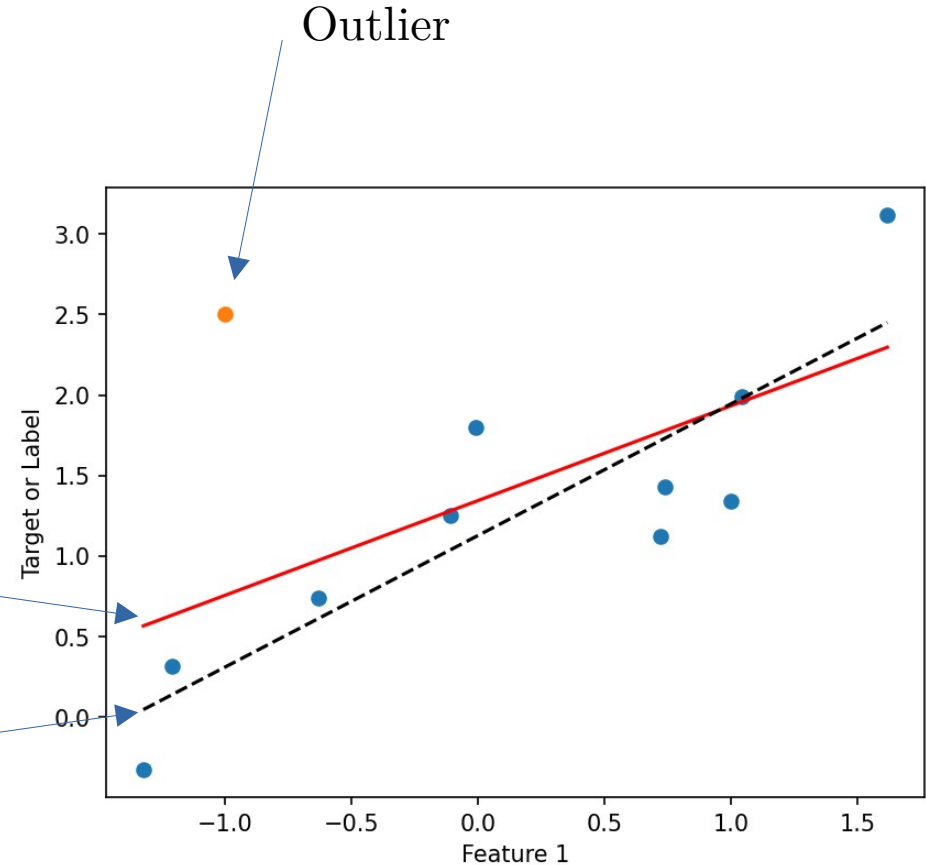
- Example of linear regression
- Loss function

$$E = \sum_i (y_i - \hat{y}_i)^2$$

Very sensitive to outliers because of the square

Linear regression
including the outlier

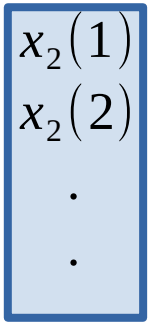
Linear regression
without the outlier



Normalizing data

- Normalize the feature values to mean zero and variance 1
- For feature i , sample j :


$$y_i(j) = \frac{x_i(j) - \mu_j}{\sigma_j}$$

$$x = \begin{pmatrix} x_1(1) & x_2(1) & \dots & x_N(1) \\ x_1(2) & x_2(2) & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & x_N(K) \end{pmatrix}$$


If this feature is $x_{1000} + 1000$

→ The gradient can be much larger or smaller in one direction but the step is the same for all

Gradient step


$$w_{n+1} = w_n - \alpha \nabla E(w_n)$$

Filling missing values

Imputation of missing values

See: <https://scikit-learn.org/stable/modules/impute.html>

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1N} \\ x_{21} & ? & x_{23} & \dots & \cdot \\ x_{31} & x_{32} & x_{33} & \dots & \cdot \\ \cdot & x_{42} & \cdot & \dots & \cdot \\ \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & x_{K2} & \cdot & \dots & x_{KN} \end{pmatrix}$$

Missing value!
What should we do?

Filling missing values

Imputation of missing values

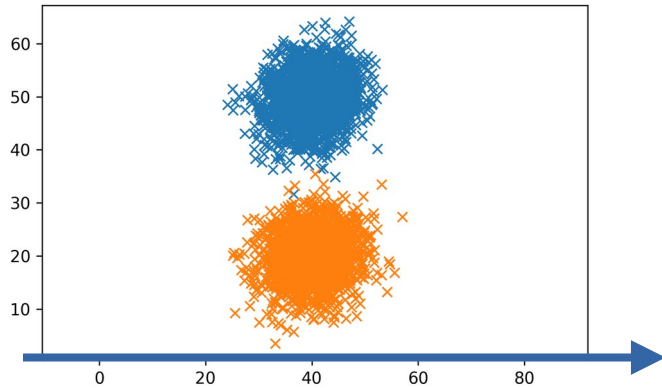
See: <https://scikit-learn.org/stable/modules/impute.html>

- **Simple approach 1:** discard the samples with missing values
- **Simple approach 2:** replace by the mean value of the feature
- **More advanced approach:** find the k most similar samples (using the other features) and replace taking (an average of) their value
- **Even more advanced approach:** train a ML model to predict the missing value from the rest of the data

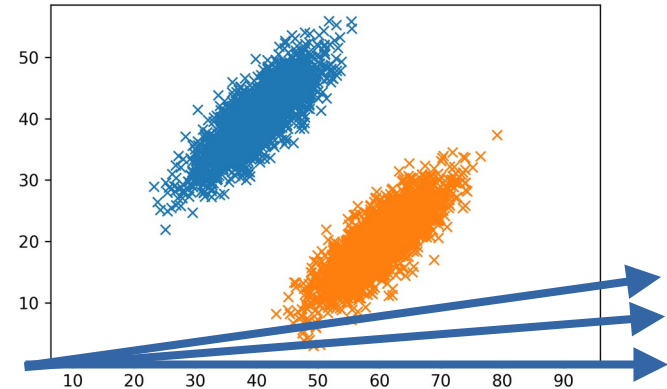
Feature selection

How to reduce the number of features?

- Statistics on the features: looking at the mean/variance of the different classes
- Comparing features (correlations)



Useless feature



Correlated features

When is feature preprocessing is important?

- When the dataset is small
- When the quality of the features is bad (noise, missing values)
- When we do not have a nice table of features, ex: text ...

Feature generation

<https://www.kaggle.com/code/ryanolbrook/creating-features>

<https://www.kaggle.com/learn/data-cleaning>

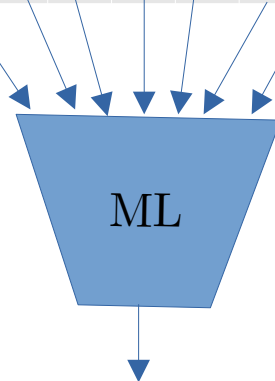
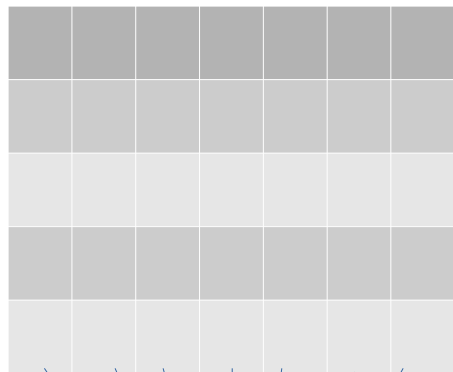
Not all raw data are matrices of real numbers

Table of values:

Matrix

Features

Samples



Complex
combination of
features

Output

Methods we have seen:

- 1) expect vectors of real numbers per sample,
- 2) the position of a feature in the vector is not important (as long as it keeps the same position during training and testing)

Not all raw data are matrices of real numbers



Table of values:
Matrix

1) Vectors of real numbers per sample

What about categorical data?
Text?

2) the position of a feature in the vector is not important

What about time-series? Images? Point clouds? Networks?
How to account for the spatial structure?

From categories to numbers

1) Assign a number to each category

Marks A B C D E F \rightarrow 1 2 3 4 5 6

fruits: Apples Oranges Bananas \rightarrow ?

Close category should have close numbers

No similarity between categories? \rightarrow **one-hot-encoding**

Apples	Oranges	Bananas
1	0	0
0	1	0
0	0	1

Apples + Bananas
1
0
1

- Categories all at the same distance
- Adding categories makes sense

But:

- Size increase 1 value \rightarrow n values
- Binary, not real

From categories to numbers

Instead of one-hot-encoding, we could have hash values
 n binary numbers $\rightarrow d$ real numbers with $d < n$

- Very good if we have many categories
- But: close hash values do not mean similar samples. Distance between hash values does not make sense

Ideally: we would like an encoding where similar categories are close, when we have this information about the categories. (we see that later on)

Text data

How to turn a text into a feature vector?

- Cut the text in elementary pieces or “tokens”: words, n-grams or letters
- Most common: tokens = words

The bag of words

- Select a list of important keywords
- Each entry in the vector correspond to a keyword
- 1 if the keyword is present (or count), 0 otherwise

Can use a list of “stop-words” which define unimportant words

Note: the positions of the words in the text are lost

TF-IDF

Most common words are not very informative : “the”, “and”, ...
How to select relevant ones **automatically**?

TF-IDF: Term Frequency - Inverse Document Frequency

Data: a set of n text documents

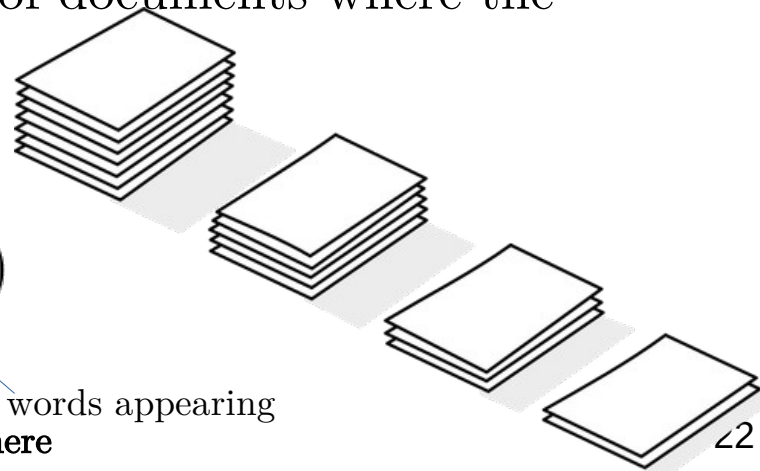
- $k(t,d)$ number of term t in document d
- Term Frequency in document d : $tf(t,d) = k(t,d) / \sum_d k(t,d)$
- Inverse document frequency: $idf(t)$ the number of documents where the term t appear

$$idf(t) = \log \frac{n}{1 + df(t)}$$

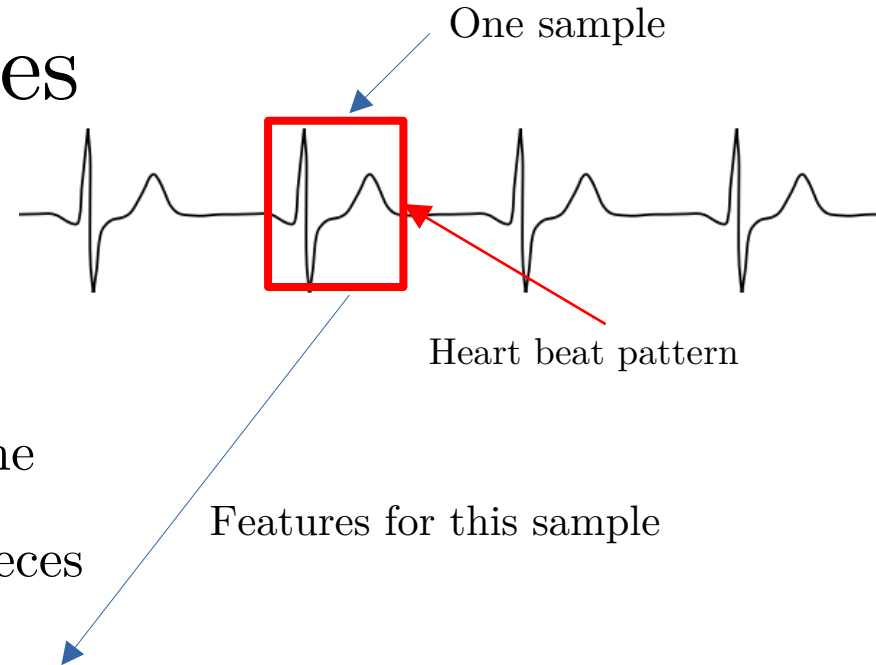
$$tf-idf(t,d) = tf(t,d) \times idf(t)$$

High score for words appearing often

Low score for words appearing often **everywhere**



Time series



Idea:

- 1) neighborhood in time is important
- 2) pattern invariant by translation along time

Locality and windowing: cut the signal in pieces

Example of features for a time window:

mean, variance, zero-crossing, autocorrelation, Fourier transform coefficients

More examples:

<https://tsfresh.readthedocs.io/>

Images

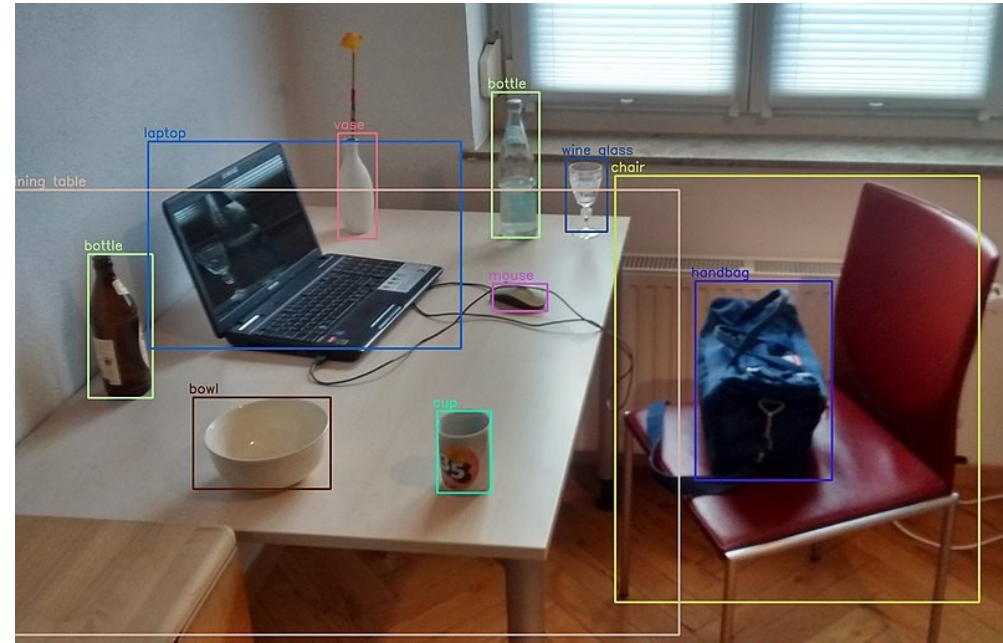
Idea:

- 1) neighborhood in **space** is important
- 2) pattern invariant by translation along **space**

Locality and windowing. Cut the image in patches, with or without overlap

“Convolution” special architecture

“Convolutional neural networks”, learn filters / elementary shapes.



Wikimedia CC-BY-SA-4.0 by MTheiler

<https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks>

Summary for data with geometry

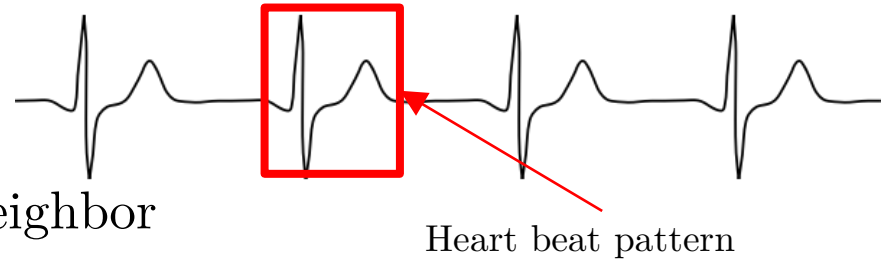
Information is in patterns

Patterns are particular combinations of neighbor values.

→ Patterns spread across features.

It is important to encode the geometry.

Not all ML methods expect a matrix of samples and features!



More recent feature selection / generation

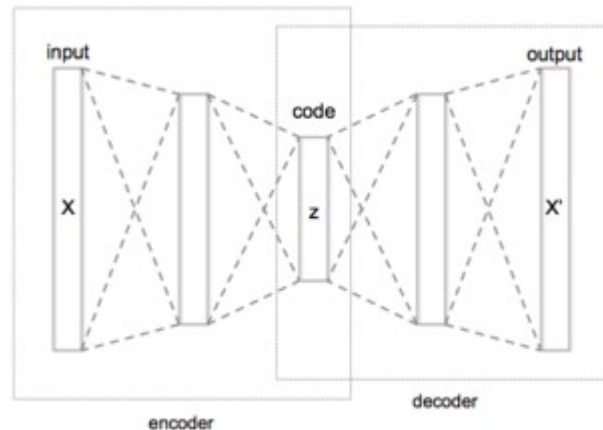
The bitter lesson strikes back

Self-supervised learning

- Examples:
- Autoencoders
- BERT (text and language)
- Other tasks <https://ai.googleblog.com/2021/09/discovering-anomalous-data-with-self.html>

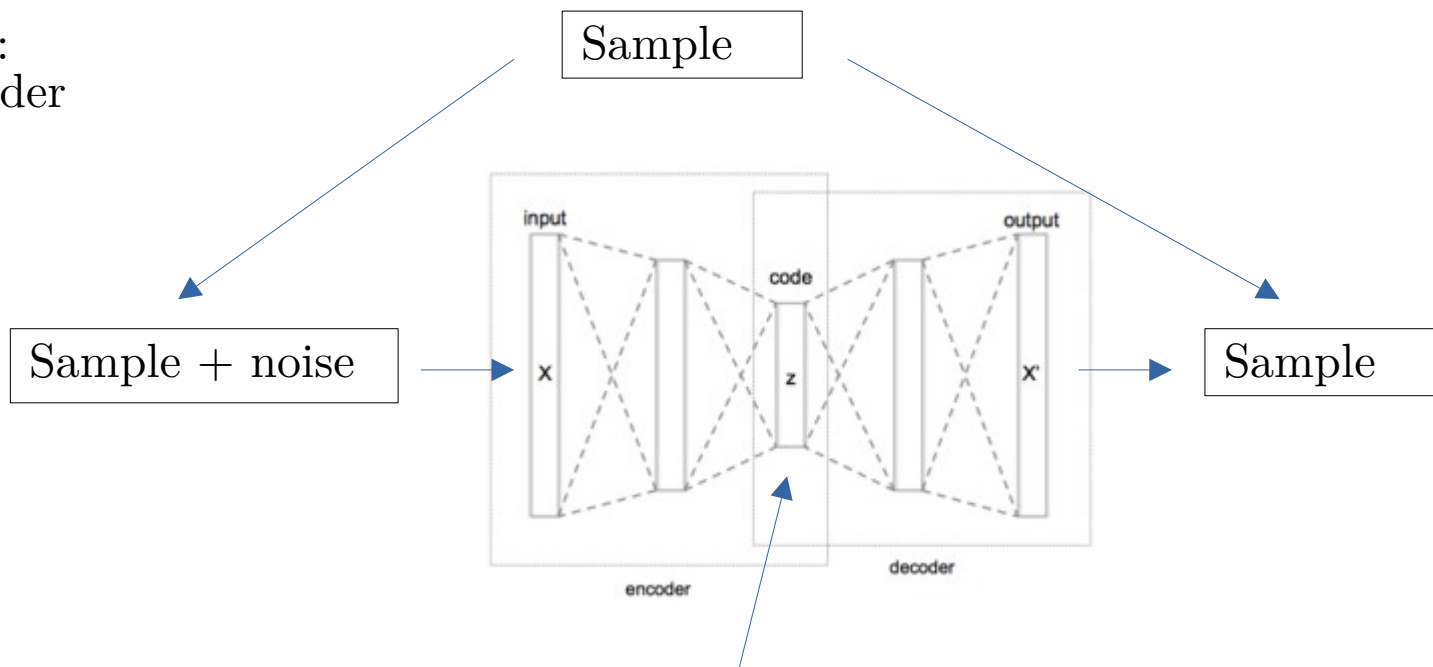
Principle: Modify the data and train to detect the modification or reconstruct the original data

Embedding or latent representation:
Machine internal representation of the input



Self-supervised learning

Example:
autoencoder



Embedding or latent representation:
Machine internal representation of the input

Text data

What about word similarity? And encoding with real numbers?

Word2vec: Associate a word with a feature vector in some latent space.

- Learning process with a neural network

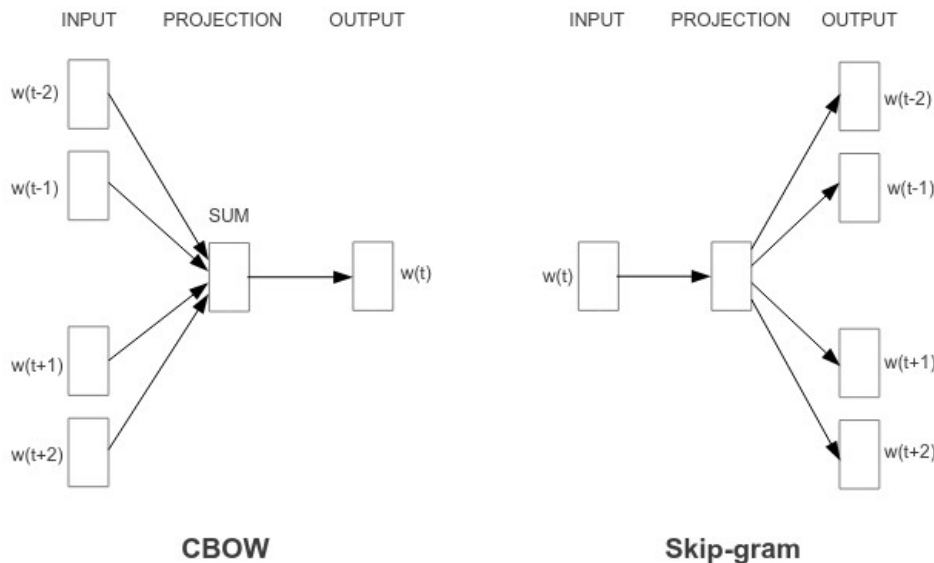


Figure from the original Word2vec paper, Efficient Estimation of Word Representations in Vector Space, Mikolov et al. (2013)

- Words are similar if they are often surrounded by the same words
- Operation on embeddings: king-man+women = queen, Paris - France + Italy = Rome

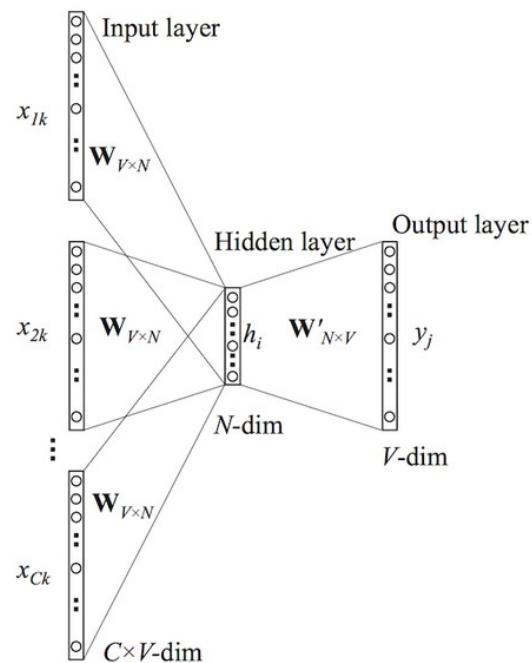
Word2vec

Links for more info (optional):

<https://jaketae.github.io/study/word2vec/>

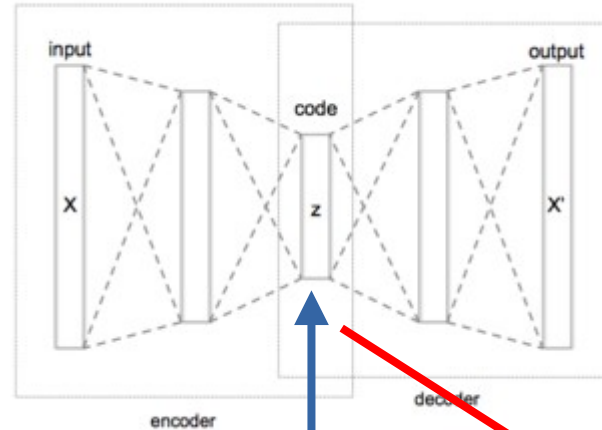
<http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/>

Matrices W have learnable weights (entries)



Self-supervised learning

Auto encoder:
Reconstruct the input



Embedding space
Or latent space

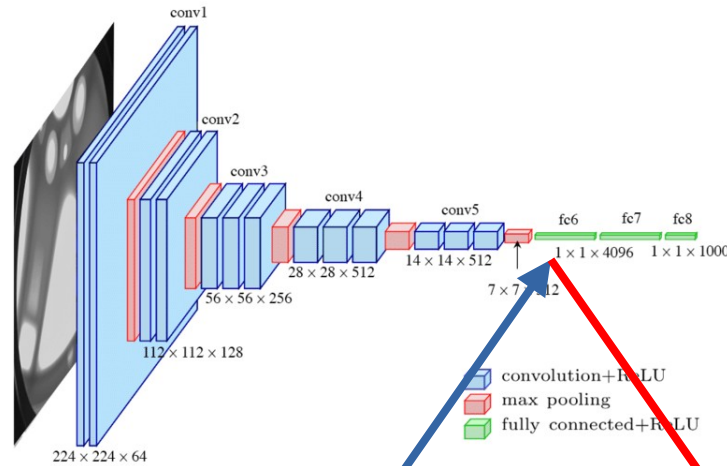
Feature vector!

ML model 2

- * unsupervised
- * semi-supervised
- * supervised

Transfer learning

Train on task 1
Supervised



Example:
Image classifier

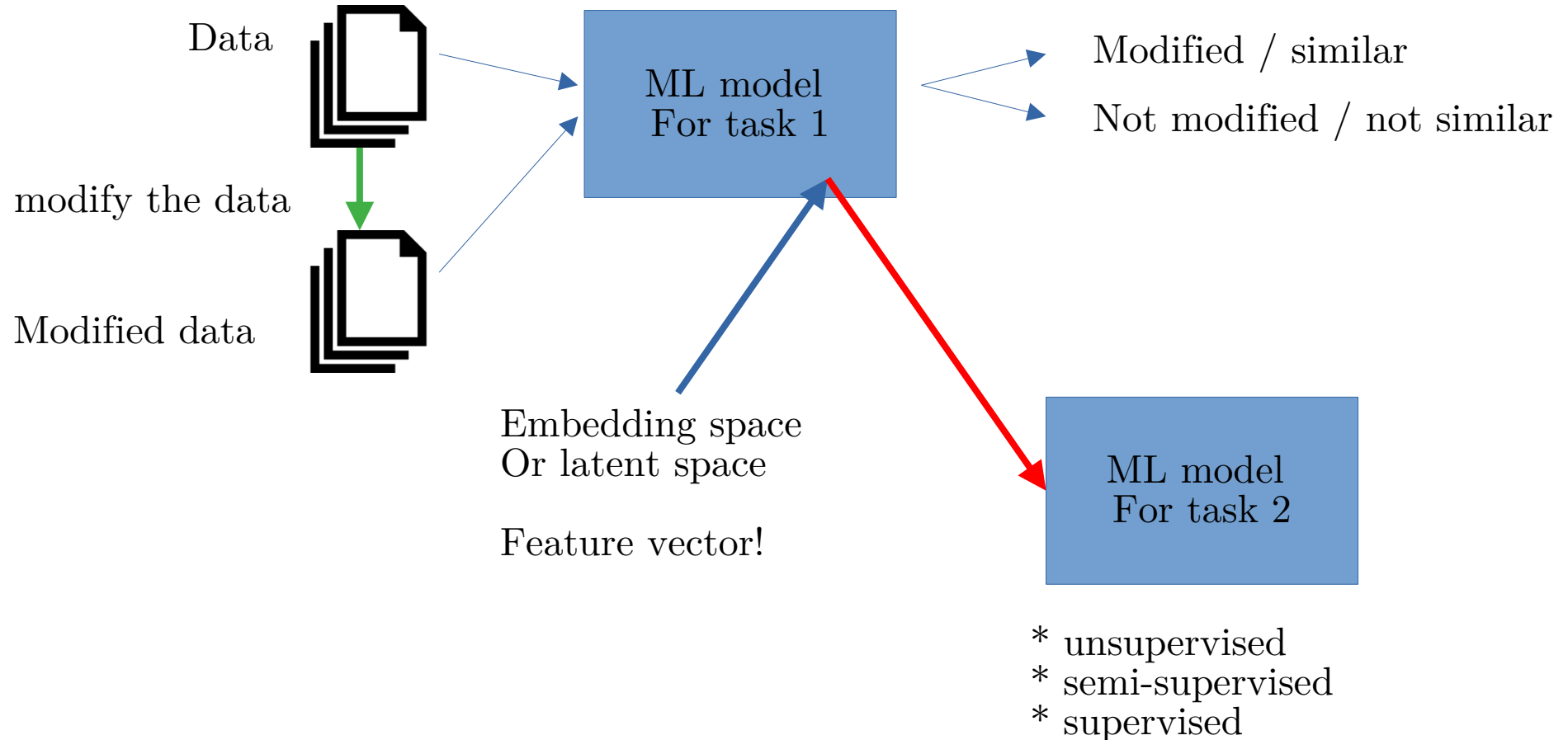
Embedding space
Or latent space

Feature vector!

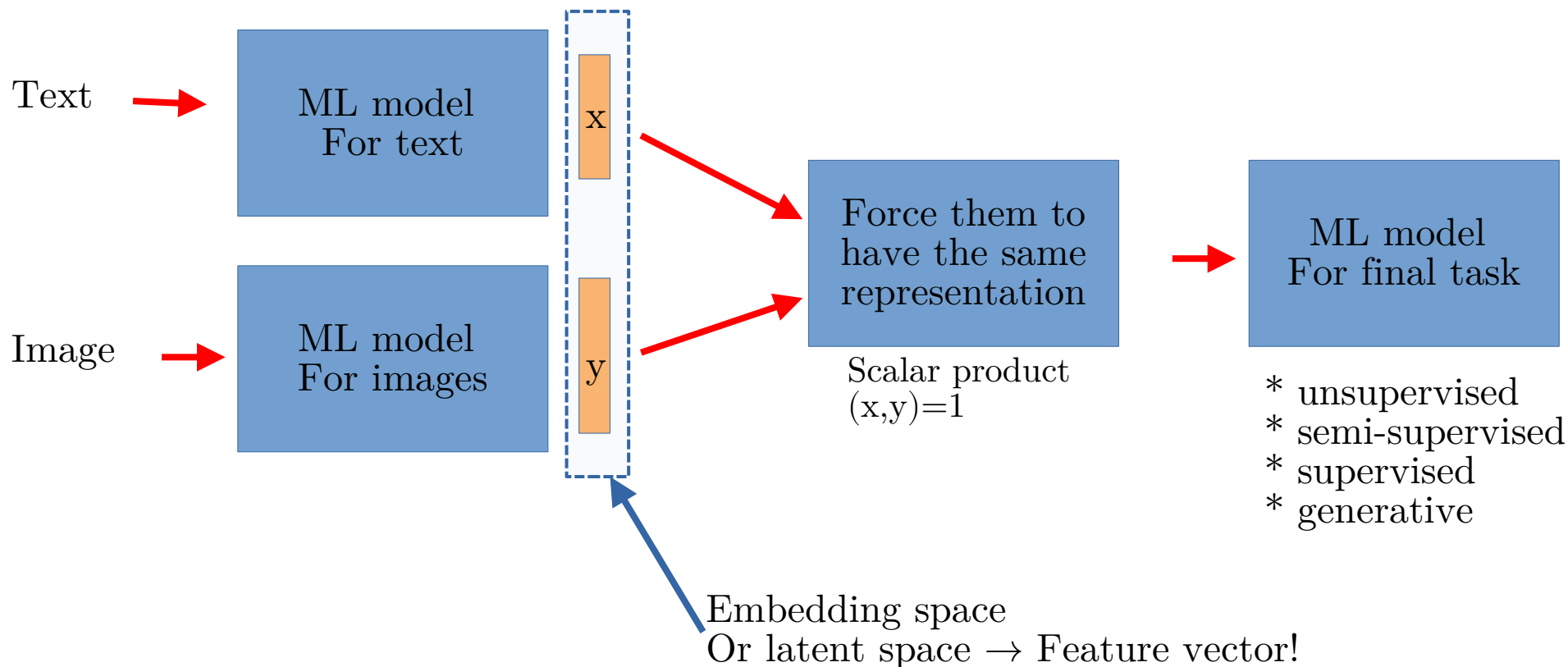
ML model
For task 2

- * unsupervised
- * semi-supervised
- * supervised

Self-supervised learning



Multi-modal



Example: CLIP for associating image and text, <https://openai.com/research/clip>