Lecture 14: Summary

Nils Olovsson
nils.olofsson@igp.uu.se

Introduction to data science and Python programming for medical research
Uppsala University

2025

UPPSALA
UNIVERSITET

Content

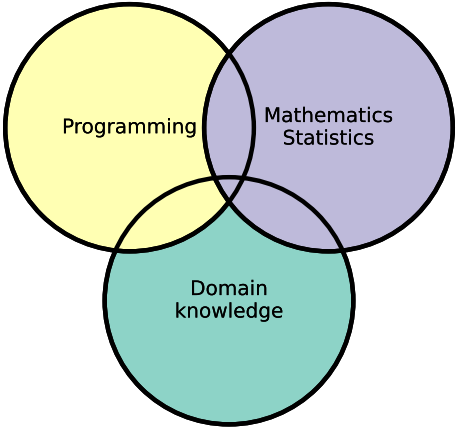
Introduction

Introduction

Introduction: Course plan learning outcomes

- Introductory knowledge in python programming.
- Calculate descriptive statistics and perform statistical tests.
- Plot data.
- Perform linear and non-linear regression.
- Work with high dimensional data and perform clustering.
- Dimensionality reduction with PCA, t-SNE and UMAP.
- History of AI and artificial neural networks (ANN).
- Explain how a small ANN can discriminate between two categories.
- Explain how machine learning models learn from data.
- Work with digital images.
- Use pre-trained deep learning models to solve problems in medical image analysis.

Introduction: What is data science?



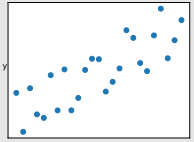
Adapted from "Datascience handbook".

Introduction: Peak under the hood

Data

$x = \{x_1, \cdots, x_N\}$

$y = \{y_1, \cdots, y_N\}$

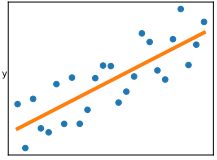


→

Model

(Linear regression)

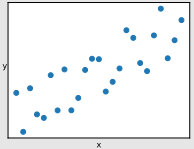
→



Introduction: Peak under the hood

Data

$$x = \{x_1, \cdots, x_N\}$$
$$y = \{y_1, \cdots, y_N\}$$

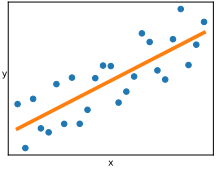


→

$$k = \frac{\sum_i y_i - \frac{N}{\sum_i x_i} \sum_i x_i y_i}{\sum_i x_i - \frac{N}{\sum_i x_i} \sum_i x_i^2}$$

```
1 import numpy as np
2 N = len(x)
3 sum_x = np.sum(x)
4 sum_y = np.sum(y)
5 sum_x2 = np.sum(np.power(x,2))
6 sum_xy = np.sum(np.multiply(x,
      y))
7 k = (sum_y - N/sum_x * sum_xy)
      /
8     (sum_x - N/sum_x * sumx2)
9
```

→



Python

Python

Python: Types and operations

Numbers	numpy
Strings	matplotlib
Lists	pandas
Dictionaries	scipy
Boolean	scipy
Numpy arrays	sklearn
Other classes	pytorch

Python: Program flow

Conditional

```
a = 1
b = 2
if b > a:
    print("b is larger than a")
else:
    print("a could be larger than b.")

```

Repetitions

```
N = 10
for i in range(N):
    print(f"{i}: Hello!")

```

Exceptions

```
v = 0
try:
    v = input("Input a number: ")
    v = float(v)
except Exception as e:
    print(e)
```

Python: Structure

Functions

Classes

Modules

Python: Packages

Course content in python packages.

Linear algebra	numpy
Plotting	matplotlib
Tabular data	pandas
Statistics	scipy
Computing	scipy
Machine learning	sklearn
Image processing	skimage
Medical image processing	Simple ITK
Deep learning	pytorch

`pip`
Python packaging index.

`conda`
Python packaging index.

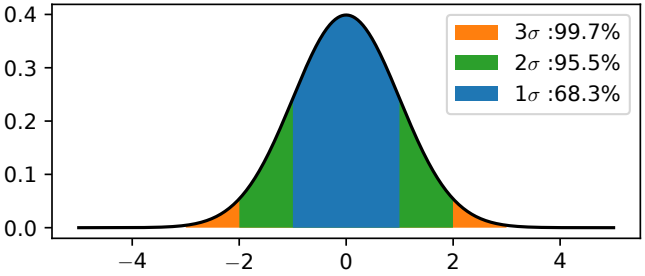
Statistics

Statistics

Statistics: Summary statistics

- Mean
- Variance
- Standard deviation

- Median
- Percentiles



Statistics: Tests

Single group

- One sample t-test

Two related

- Paired t-test
- Wilcoxon signed rank test

Two unrelated

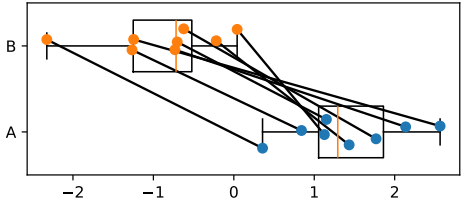
- Two sampled t-test
- Wilcoxon rank sum test
- Mann-Whitney U rank test

Multiple groups

- ANOVA
- Kruskal-Wallis

Correlation

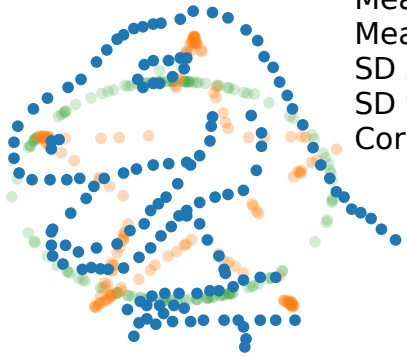
- Pearson
- Spearman



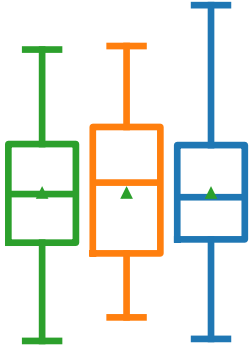
Statistics: Datasaurus dozen

We need to be **careful** when only looking at **summary statistics and correlation coefficients**!

What is the *shape* of the data?



Mean x: 54.26
Mean y: 47.83
SD x: 16.71
SD y: 26.84
Correlation: -0.06



Datasaurus dozen
https://www.research.autodesk.com/publications/same-stats-different-graphs/

Regression

Regression

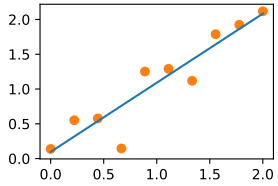
Regression: Linear, nonlinear, logistic

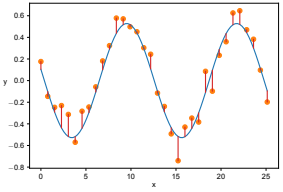
Supervised

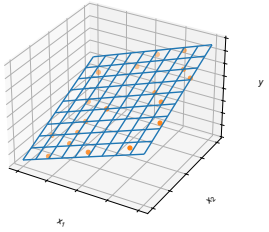
We want to **fit** a **function**, **curve** or a **model** to our data in some **optimal** manner.

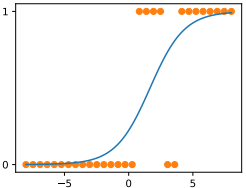
Regression methods:

- Linear regression
- Multi-linear regression
- Non-linear regression
- Logistic regression









Regression: Problems

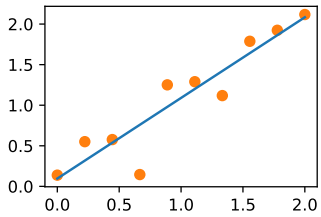
Two families of problems and methods:

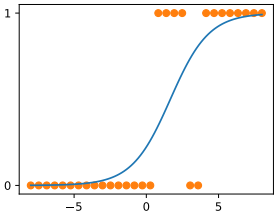
Regression

Continuous input $\rightarrow$ *continuous output*
Example: Medication dosage (mg) and blood pressure (hg).

Classification

Continuous input $\rightarrow$ *discrete/binary output*
Example: Medication dosage (mg) and treatment outcome (successful or not)



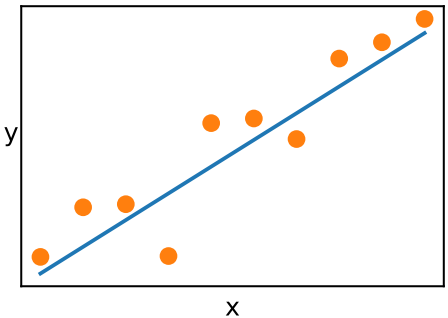


Regression: Linear

Assume a **linear relationship** between x and y.

We want to **fit a straight line** to data such that we can **predict** y from x.

$$y = kx + m$$

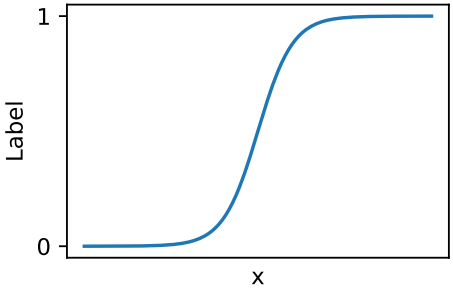


Regression: Logistic

Looked at continuous input → continuous output.

Now look at **continuous input** → **binary output**, *i.e.* **classification**.

These are often modeled with some s-shaped **logistic** function.



https://en.wikipedia.org/wiki/Logistic_function

High dimensional data

High dimensional data

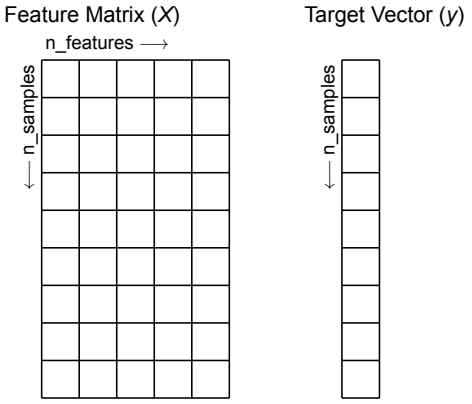
High dimensional data: Data matrix

Also called the **feature matrix** contains all our sampled feature vectors where **one sample** is represented by one **row**.

If the problem is a **regression or classification** problem we also have **target vector**.

```
#import numpy as np
from sklearn.datasets import load_breast_cancer
X, y = load_breast_cancer(return_X_y=True)
print(f"Shape of X: {X.shape}")
print(f"Shape of y: {y.shape}")
```

```
>> Shape of X: (569, 30)
>> Shape of y: (569,)
```



Adapted from Python Data Science Handbook (VanderPlas)

High dimensional data: Covariance matrix

Normal distributions in 1D determined by two scalar values, mean and variance.

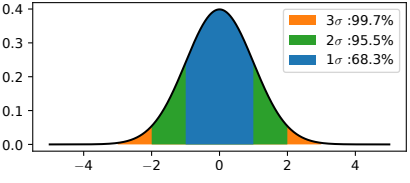
Multivariate normal distributions determined by mean, vector, and covariance matrix, C.

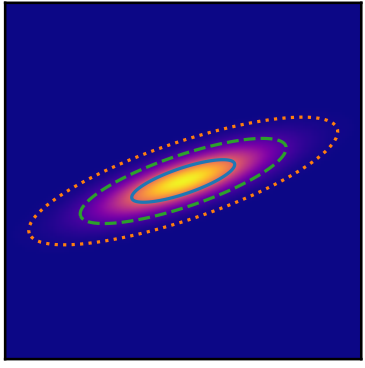
$N \times N$, where N is dimension of the space.

Covariance matrices are square and symmetric.

$$C_{2D} = \begin{pmatrix} v_x & a \\ a & v_y \end{pmatrix} \quad C_{3D} = \begin{pmatrix} v_x & a & b \\ a & v_y & c \\ b & c & v_z \end{pmatrix}$$

```
1 import numpy as np
2 from sklearn.datasets import load_diabetes
3 X, y = load_diabetes(return_X_y=True)
4 C = np.cov(X)
```





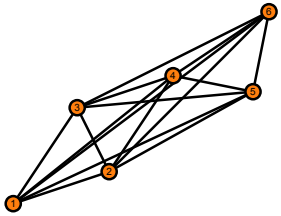
High dimensional data: Distance matrix

Pairwise distances between points, *e.g.* in matrix X. $M \times M$, where M is samples in the data. Distance matrices are square and, typically, symmetric.

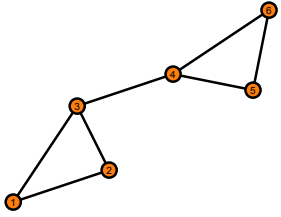
Can connect all points (dense) or subset of points (sparse).

$$D = \begin{pmatrix} 0.0 & 0.6 & 0.7 & 1.3 & 1.7 & 2.0 \\ 0.6 & 0.0 & 0.4 & 0.7 & 1.0 & 1.4 \\ 0.7 & 0.4 & 0.0 & 0.6 & 1.1 & 1.3 \\ 1.3 & 0.7 & 0.6 & 0.0 & 0.5 & 0.7 \\ 1.7 & 1.0 & 1.1 & 0.5 & 0.0 & 0.5 \\ 2.0 & 1.4 & 1.3 & 0.7 & 0.5 & 0.0 \end{pmatrix} \quad D = \begin{pmatrix} 0.0 & 0.6 & 0.7 & 0.0 & 0.0 & 0.0 \\ 0.6 & 0.0 & 0.4 & 0.0 & 0.0 & 0.0 \\ 0.7 & 0.4 & 0.0 & 0.6 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.6 & 0.0 & 0.5 & 0.7 \\ 0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.5 \\ 0.0 & 0.0 & 0.0 & 0.7 & 0.5 & 0.0 \end{pmatrix}$$

```
1 from sklearn.datasets import load_diabetes
2 import sklearn.metrics
3 X, y = load_diabetes(return_X_y=True)
4 D = sklearn.metrics.pairwise_distances(X)
```



Fully connected graph.



Connects only neighboring points.

Clustering

Clustering

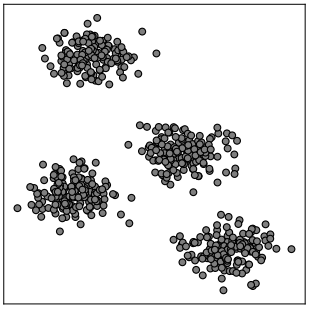
Clustering: Kmeans

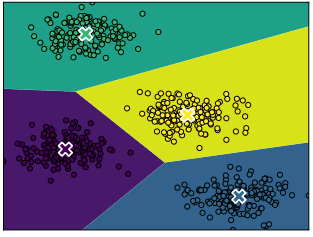
Unsupervised

Simple and commonly used method.

Can't properly cluster elongated or nonlinear data.

Need to specify k, the number of clusters.





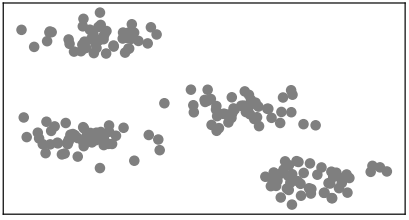
Clustering: Gaussian mixture models

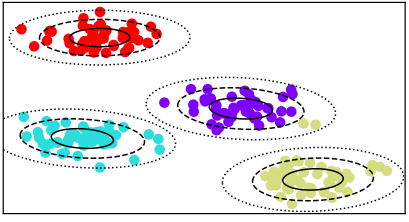
Method for probability density estimation.

Can cluster elongated data in any direction.

Can't properly cluster nonlinear data.

Need to specify the number of clusters.



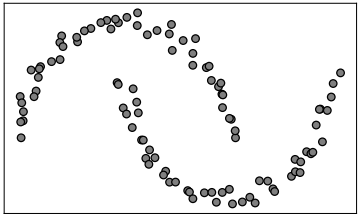


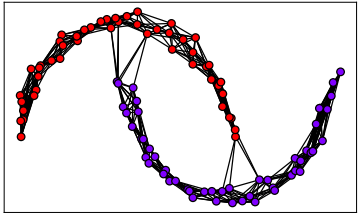
Clustering: Spectral

Graph and distance/affinity matrix used to separate data into clusters based on the strength of their connections..

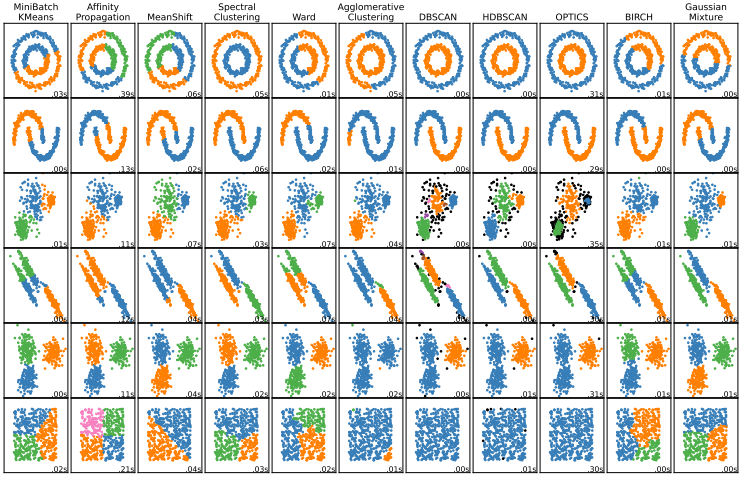
Can cluster nonlinear data.

Need to specify the number of clusters.





Clustering: Overview



https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

Principal component analysis

Principal component analysis: Dimensionality reduction

Vectors

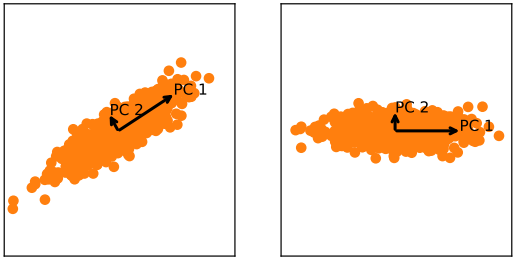
Principal components are **vectors**, $\mathbf{v}_i$, that create a data-oriented coordinate system.

Linear transform

Can be seen as a **rotation or re-orientation** of the data points.

Variance

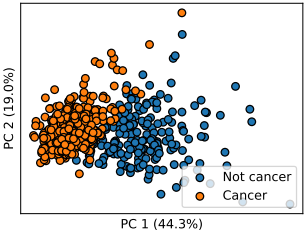
Each **vector** is associated with a **variance** perpendicular to its direction.



Principal component analysis: Visualization

Cancer cell nuclei morphology

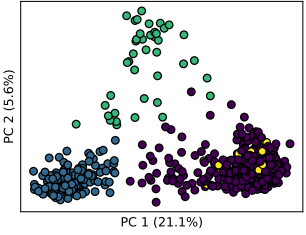
30 $\rightarrow$ 2 **dimensions**



CAF data

Cancer associated fibroblast transcriptomics.

557 $\rightarrow$ 2 **dimensions**

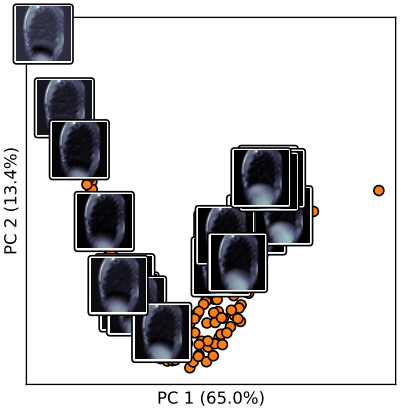


Principal component analysis: Latent space

PCA can find a latent, low dimensional, space that represents the data.

E.g. respiration is a one dimensional process, your breath in and out.

As such we could hypothesize that the breathing process should be well represented with one or two principal components!



Manifold learning

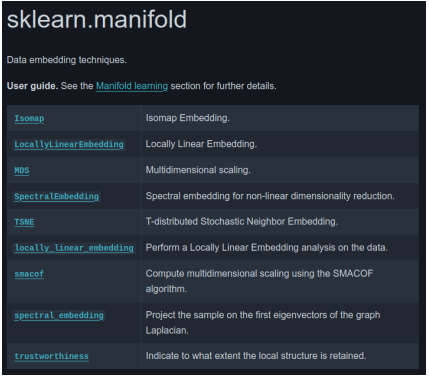
Manifold learning

Manifold learning: Unwrap data

Find mapping such that points close in original data are close after mapping. (Locally flat.)

Will look at several methods that can be divided into two families.

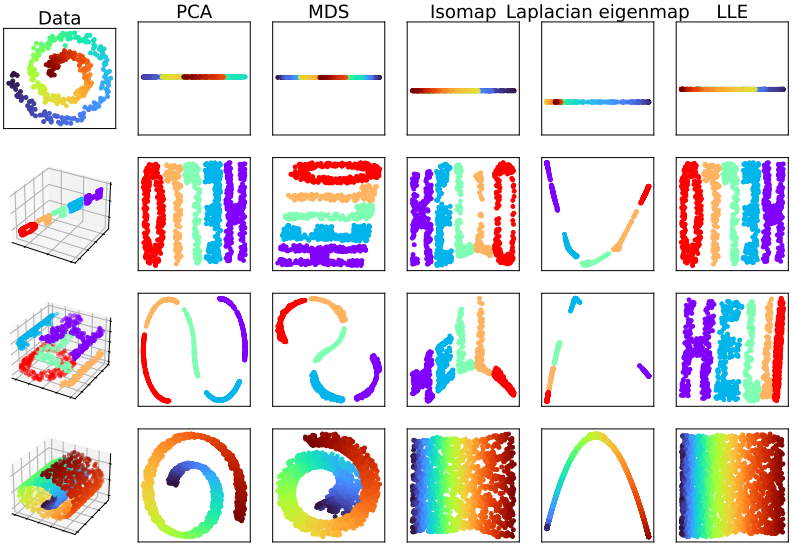
- MDS, Isomap, Eigenmap, LLE } Spectral methods
- t-SNE, UMAP } Optimization (Gradient descent)



https://scikit-learn.org/stable/api/sklearn.manifold.html

https://umap-learn.readthedocs.io/en/latest/

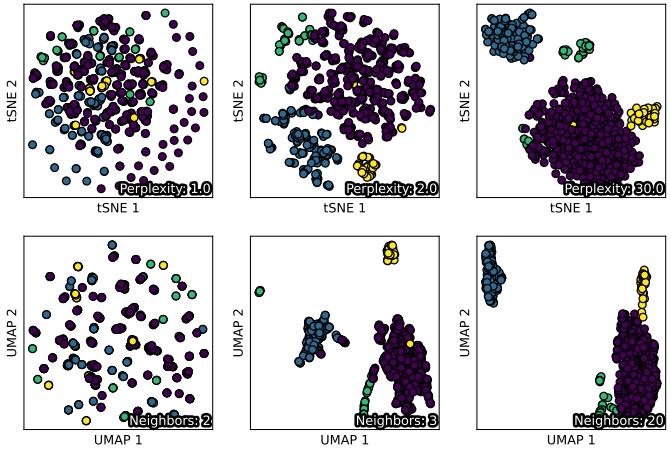
Manifold learning: Comparison



Manifold learning: t-SNE and UMAP

t-SNE

UMAP



Support vector machines

Support vector machines: Flexible regression and classification

Supervised

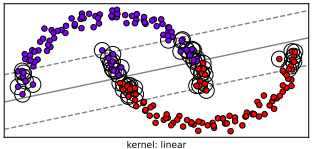
Support vector regression.

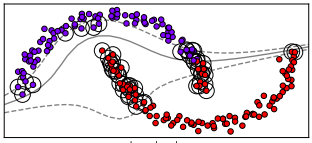
Support vector classification.

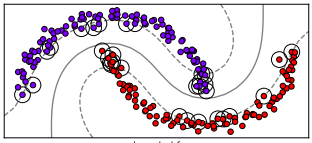
Kernels

- Linear
- Polynomial
- RBF

Radial basis function SVC is a very powerful non-linear classifier! Beware of overfitting!







Decision trees and random forests

Decision trees and random forests: Trees and ensembles

Successive splitting for regression and classification models.

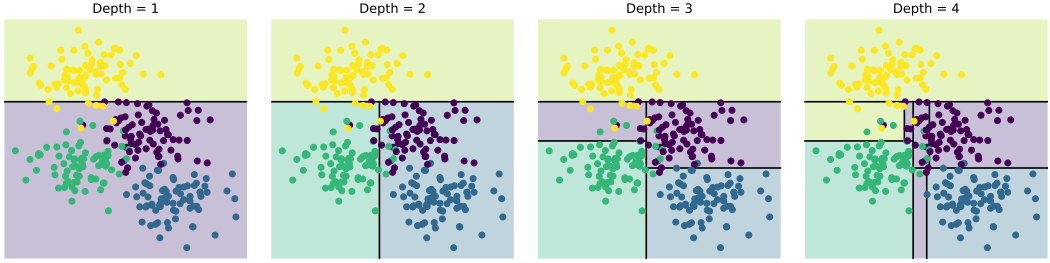
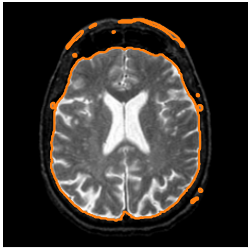
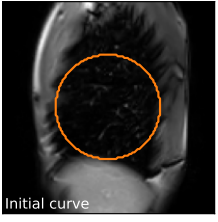


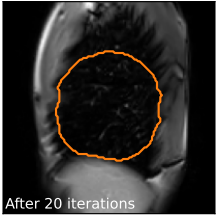
Image analysis

Image analysis

Image analysis: Segmentation







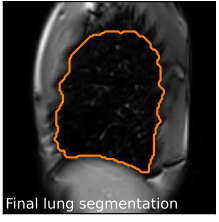


Image analysis: Mask processing and comparisons

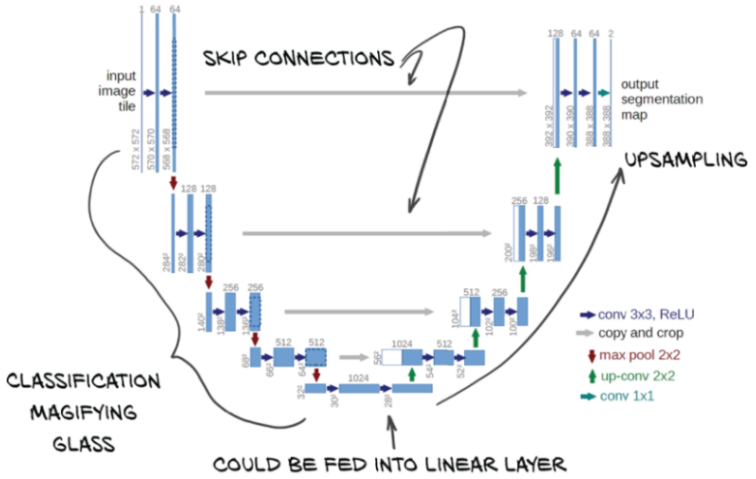
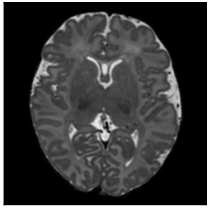
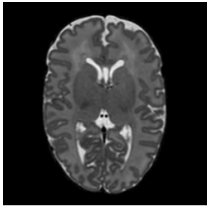


Image analysis: Registration

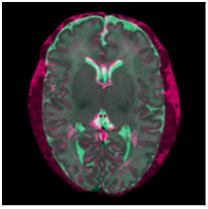
1. Transform
2. Similarity
3. Optimization



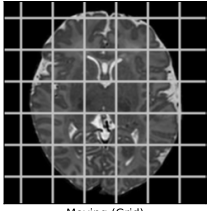
Moving



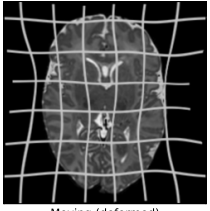
Fixed



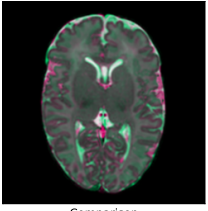
Comparison
before registration



Moving (Grid)



Moving (deformed)



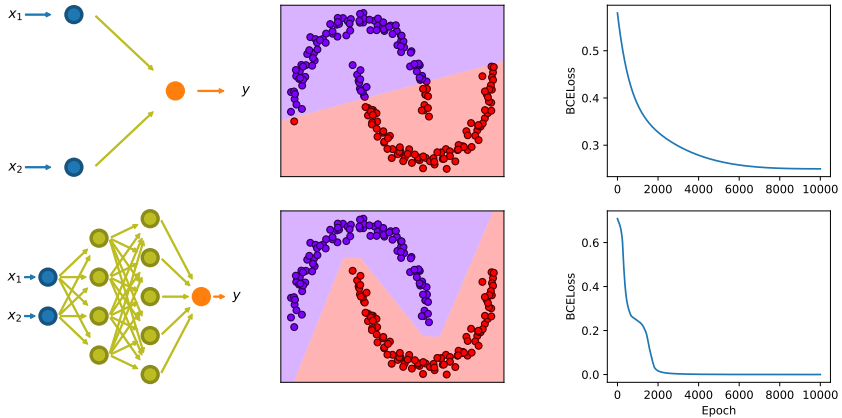
Comparison
after registration

Artificial neural networks

Artificial neural networks: Multiple layers

Supervised

Multiple layers enables nonlinear regression or classification.



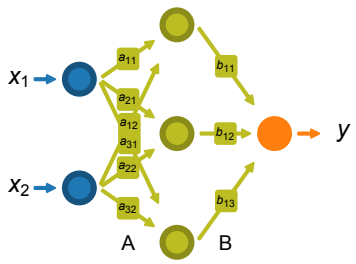
Artificial neural networks: Backpropagation

The output from the network is calculated by feeding values forward in the network.

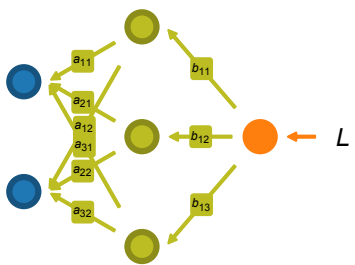
The output is compared to the expected output using a loss function.

The loss, L, is then backpropagated through the network and the weights are updated.

Forward



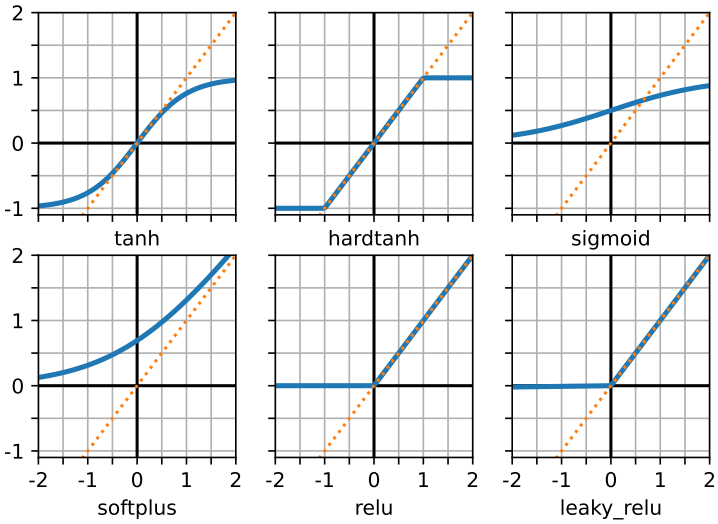
Backpropagation



Artificial neural networks: Activation functions

The limited active region of the sigmoid like functions (`tanh`, `hardtanh` and `sigmoid`) make them less popular for hidden layers.
They are useful in the output layer for classification.

Rectifying linear unit, `relu`, and its variants are popular in hidden layers.
(All have derivatives! Important!)



https://pytorch.org/docs/stable/nn.functional.html#non-linear-activation-functions

Artificial neural networks: Loss functions

Regression

Mean squared error

Targets y should be numbers on continuous scale.

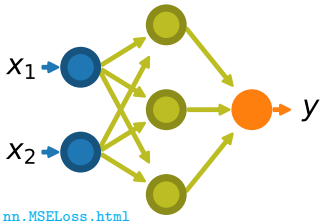
```
from torch import nn
loss_func = nn.MSELoss()
```

Single class output

Binary cross entropy

Note that the targets y should be numbers between 0 and 1.

```
from torch import nn
loss_func = nn.BCELoss()
```

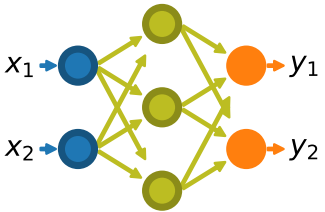


Multiple class output

Cross entropy

The output from network expected to contain the unnormalized logits, continuous values, for each class.

```
from torch import nn
loss_func = nn.CrossEntropyLoss()
```

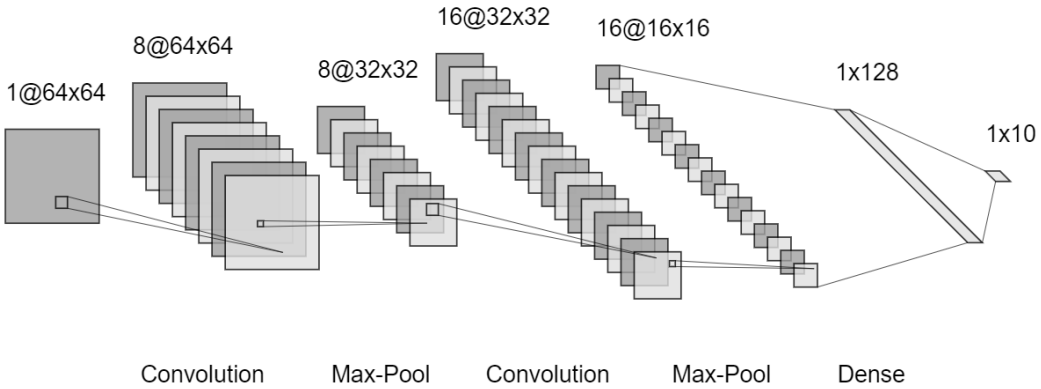


https://pytorch.org/docs/stable/generated/torch.nn.MSELoss.html
https://pytorch.org/docs/stable/generated/torch.nn.BCELoss.html
https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html
https://pytorch.org/docs/stable/nn.html#loss-functions

Deep learning image analysis

Deep learning image analysis: Classification

E.g. does the patient in the image have cancer.



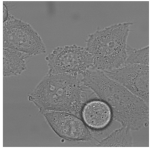
https://github.com/MachineLearningBiomedicalApplications
https://github.com/MachineLearningBiomedicalApplications/notebooks/tree/main/Chapter%2011%20-%20Convolutional%20networks

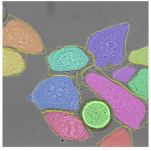
Deep learning image analysis: Segmentation

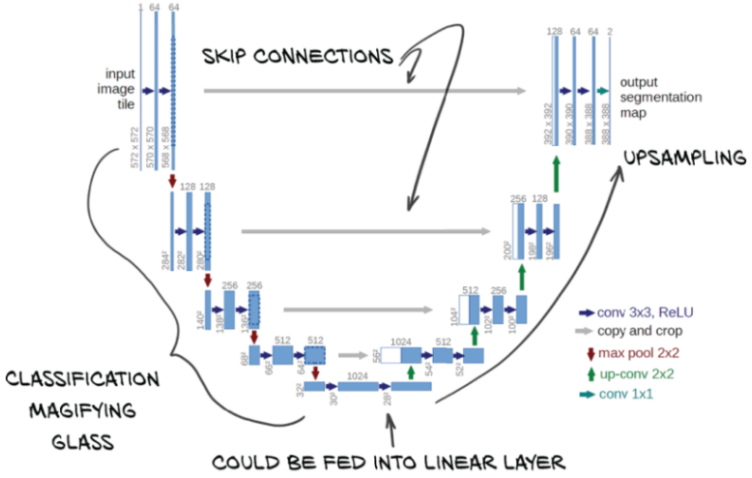
Unet architecture introduced for the segmentation of cells.

Has since proven very successful for other types of images as well.

Skip connections help parts of the network see both a bigger picture and fine details at the same time.





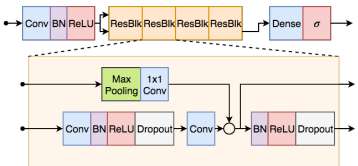


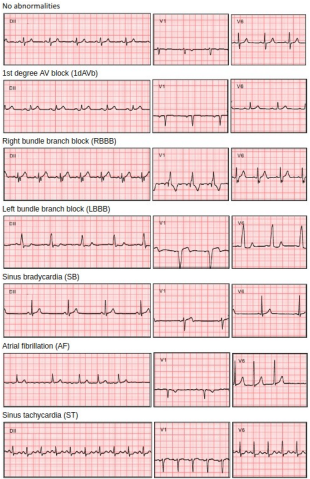
Ronneberger et al. 2015
http://arxiv.org/abs/1505.04597

Deep learning image analysis: Architectures

Ex machine learning/AI Medicine - ECG diagnosis

Historical data from 2010-2017.
1.6M patients





UPPSALA UNIVERSITET

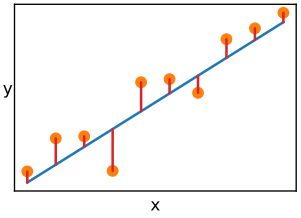
Key concepts

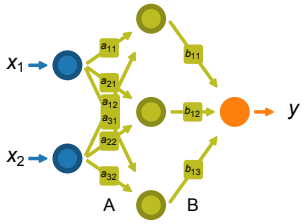
Key concepts

Key concepts: Model parameters

Linear regression	$y = kx + m$
Logistic regression	Also k and m
Kmeans	Centers of all clusters
PCA	Mean value and vectors
Manifold	Low dim points
SVM	Data points representing support vectors.
Trees	Values where domain is split.
Neural networks	Weights in layers.
Convolutional networks	Weights in kernels.

Complexity is determined by number of parameters in model.





Key concepts: Supervised and unsupervised

Unsupervised

No target vector. Only data.
Can be used for data exploration, visualizations or models that recreate data from original distribution.

- Clustering
- PCA
- Manifold learning
- (Neural networks not seen in this course)

Supervised

Target vector we want to predict from data.
Regression and classification problems.

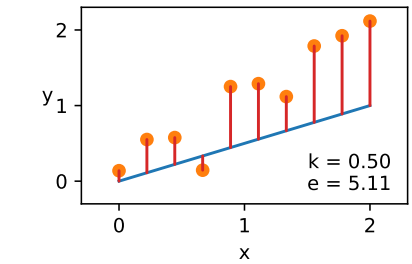
- Linear regression
- Logistic regression
- Support vector machines
- Random forests
- Neural networks

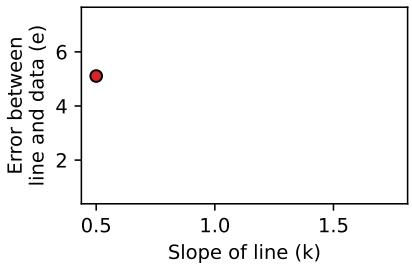
Key concepts: Error / loss function

Simpler linear regression problem. Have a line with fixed intercept of $m = 0$. Try to find the optimal slope k to fit it to our data.

$$\arg\min_k e(k; x, y)$$

By looking at several possible values for the slope we can see that there is an optimal value at the bottom of the error function!



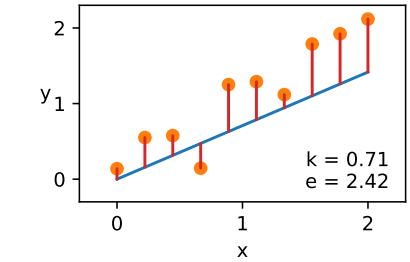


Key concepts: Error / loss function

Simpler linear regression problem. Have a line with fixed intercept of $m = 0$. Try to find the optimal slope k to fit it to our data.

$$\arg\min_k e(k; x, y)$$

By looking at several possible values for the slope we can see that there is an optimal value at the bottom of the error function!

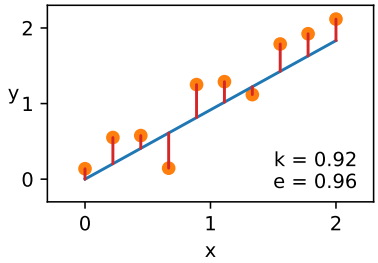


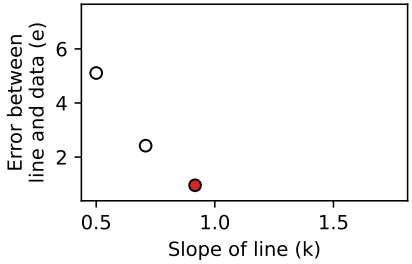
Key concepts: Error / loss function

Simpler linear regression problem. Have a line with fixed intercept of $m = 0$. Try to find the optimal slope k to fit it to our data.

$$\arg\min_k e(k; x, y)$$

By looking at several possible values for the slope we can see that there is an optimal value at the bottom of the error function!



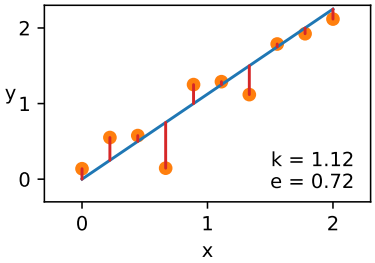


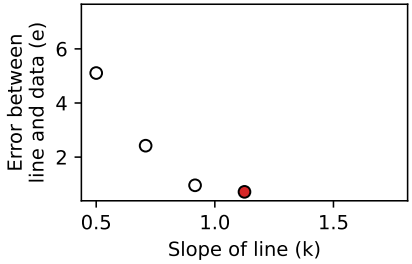
Key concepts: Error / loss function

Simpler linear regression problem. Have a line with fixed intercept of $m = 0$. Try to find the optimal slope k to fit it to our data.

$$\arg\min_k e(k; x, y)$$

By looking at several possible values for the slope we can see that there is an optimal value at the bottom of the error function!



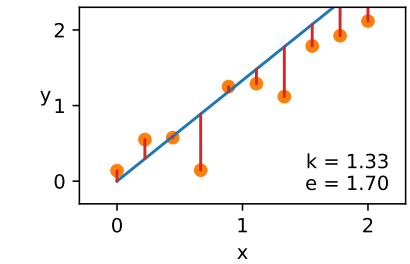


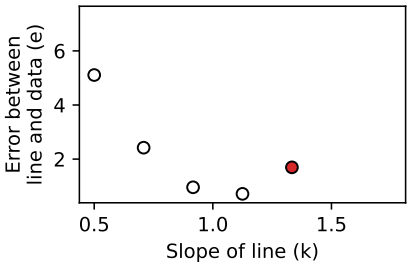
Key concepts: Error / loss function

Simpler linear regression problem. Have a line with fixed intercept of $m = 0$. Try to find the optimal slope k to fit it to our data.

$$\arg\min_k e(k; x, y)$$

By looking at several possible values for the slope we can see that there is an optimal value at the bottom of the error function!



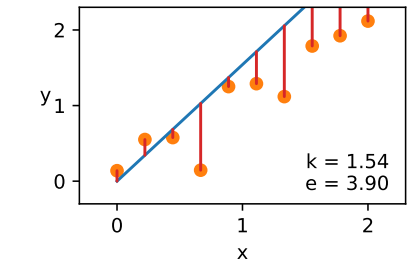


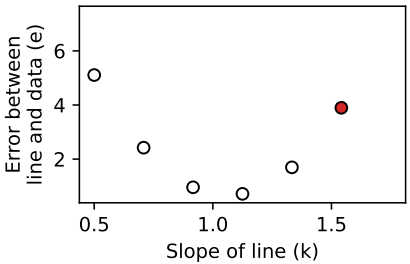
Key concepts: Error / loss function

Simpler linear regression problem. Have a line with fixed intercept of $m = 0$. Try to find the optimal slope k to fit it to our data.

$$\arg\min_k e(k; x, y)$$

By looking at several possible values for the slope we can see that there is an optimal value at the bottom of the error function!



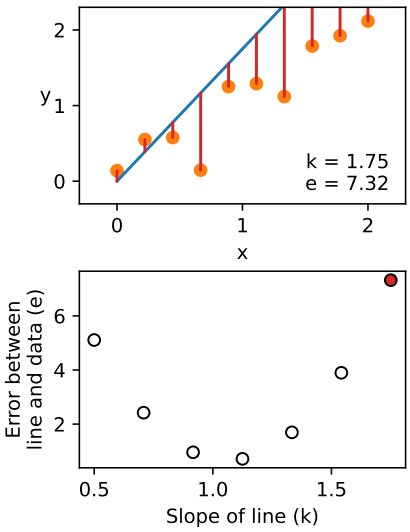


Key concepts: Error / loss function

Simpler linear regression problem. Have a line with fixed intercept of $m = 0$. Try to find the optimal slope k to fit it to our data.

$$\arg\min_k e(k; x, y)$$

By looking at several possible values for the slope we can see that there is an optimal value at the bottom of the error function!

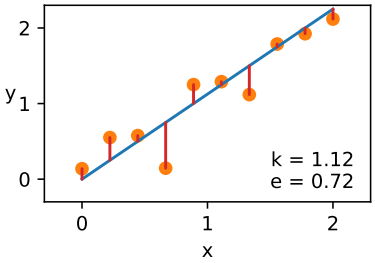


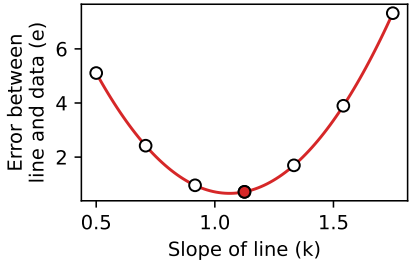
Key concepts: Error / loss function

Simpler linear regression problem. Have a line with fixed intercept of $m = 0$. Try to find the optimal slope k to fit it to our data.

$$\arg\min_k e(k; x, y)$$

By looking at several possible values for the slope we can see that there is an optimal value at the bottom of the error function!





Key concepts: Optimization

Typically define an error or loss function, e over set of parameters, $\mathbf{p}$.
Want to find parameters such that gradient, $\nabla e = 0$.

Analytical solution

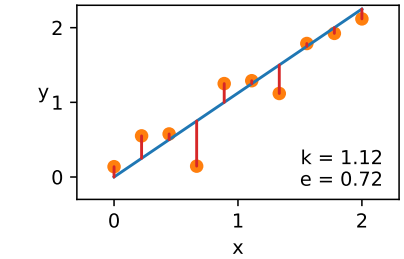
- Linear regression

Eigenvalue direct solution

- PCA
- Manifold learning
- Spectral clustering

Gradient descent (iterative)

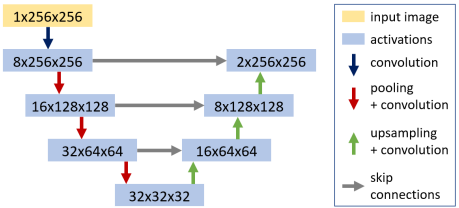
- Logistic regression
- t-SNE and UMAP
- Image registration
- Neural networks (backpropagation)

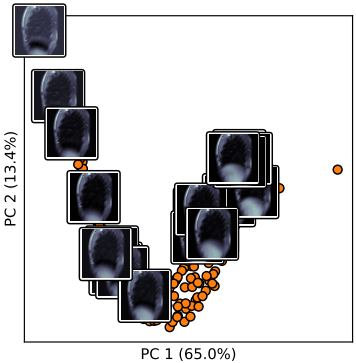


Key concepts: Latent spaces

Data can be represented in a lower dimensional space that still enables its representation or even full reconstruction.

- PCA
- Manifold learning
- Neural networks (encoder-decoder)





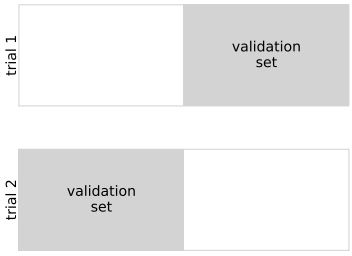
Pre-processing

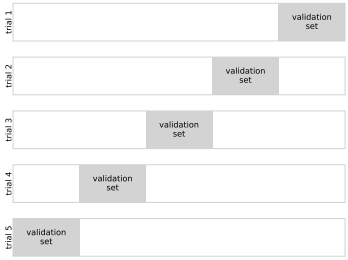
Pre-processing

Pre-processing: Holdout sets

Split data into **train** and **test** sets.

(Extreme case when we put only one data point in the test set and do this for each data point in the data set.
This way of testing is called **leave one out**.)





https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_validate.html
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.LeaveOneOut.html

Pre-processing: Missing data

Pandas has several ways of **identifying** and **handling missing values**.

`df.isnull()`	Generates Boolean mask.
`df.notnull()`	Opposite of `isnull()`.
`df.dropna()`	Get filtered df.
`df.fillna(...)`	Get copy of data with filled or imputed missing values.

```
1  import numpy as np
2  import pandas as pd
3  # Create random data and set one value to nan
4  rng = np.random.default_rng(seed=0)
5  X = rng.uniform(0, 9, size=(4, 2))
6  X[0,0] = np.nan
7  # Create dataframe
8  df = pd.DataFrame(X)
9  df.columns = ['Group 1', 'Group 2']
10 print("## Original dataframe")
11 print(df)
12 # Remove rows containting a nan
13 df_no_nan = df.dropna()
14 print("## Dataframe without nan")
15 print(df_no_nan)
```

```
1  >> ## Original dataframe
2  >>      Group 1   Group 2
3  >> 0        NaN  2.428080
4  >> 1   0.368762  0.148749
5  >> 2   7.319432  8.214800
6  >> 3   5.459722  6.565469
7  >> ## Dataframe without nan
8  >>      Group 1   Group 2
9  >> 1   0.368762  0.148749
10 >> 2   7.319432  8.214800
11 >> 3   5.459722  6.565469
```

Pre-processing: Standard scaler

Features can represent **different things** so values can be of **different ranges**.

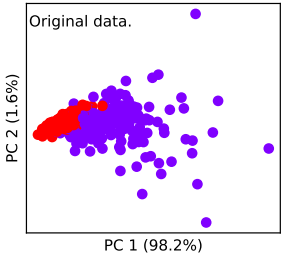
E.g. body temperature may vary between 35 and 42 degrees while blood concentration of a substance may range from 0 to $\frac{1}{1000}$.

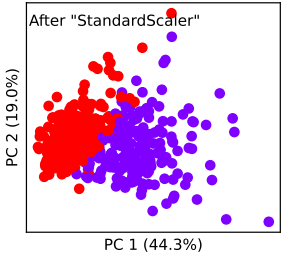
Many methods are **not scale invariant**.

Data can be normalized by *e.g.*

1. Center the mean on 0.
2. Scale such that variance is 1.

Can **improve performance** of many estimators.





https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html

Pre-processing: Log transform

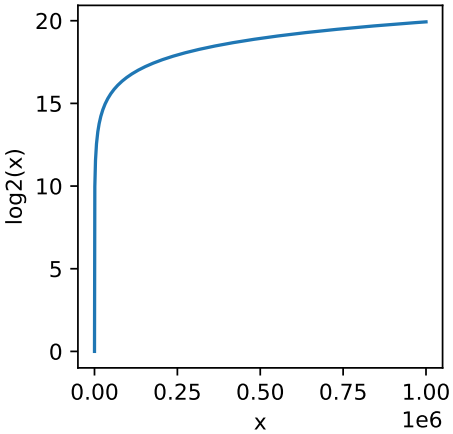
Single cell data typically **transformed** by taking a **logarithm**

$$X = \log_2(X + 1)$$

This transform **is reasonable** because:

1. All values in this type of data is ≥ 0.
2. The $+1$ makes sure that 0 in original data is 0 in transformed data.

The **logarithm compresses** the **range** of the data just as the `StandardScaler`.

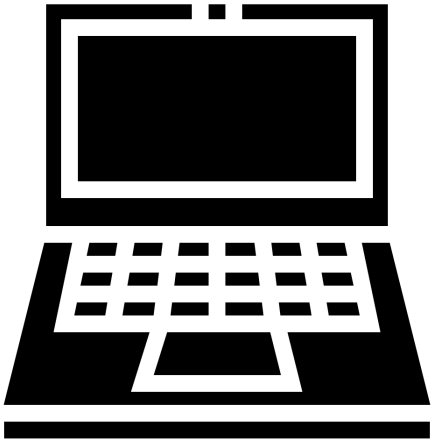


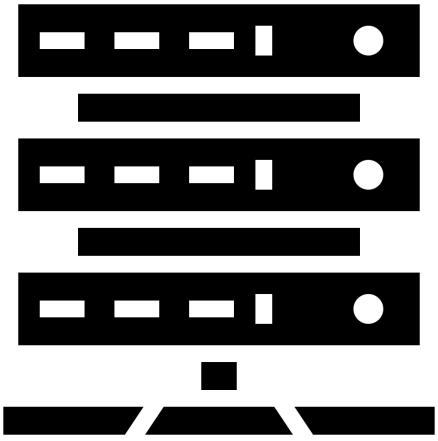
The art of using t-SNE for single-cell transcriptomics
doi:10.1038/s41467-019-13056-x

Data management

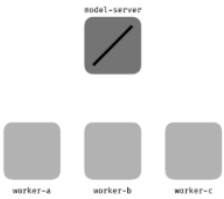
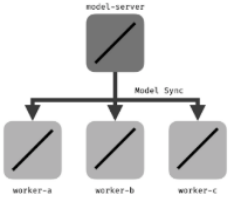
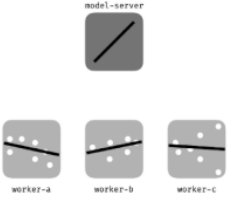
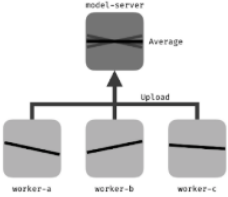
Data management

Data management: Local vs server





Data management: Federated learning

Step 1	Step 2	Step 3	Step 4
			
Central server chooses a statistical model to be trained	Central server transmits the initial model to several nodes	Nodes train the model locally with their own data	Central server pools model results and generate one global mode without accessing any data

https://en.wikipedia.org/wiki/Federated_learning