



MEDNARODNA
PODIPLOMSKA ŠOLA
JOŽEFA ŠTEFANA

INFORMATION AND COMMUNICATION TECHNOLOGIES
Master study programme

Data and Text Mining

Petra Kralj Novak

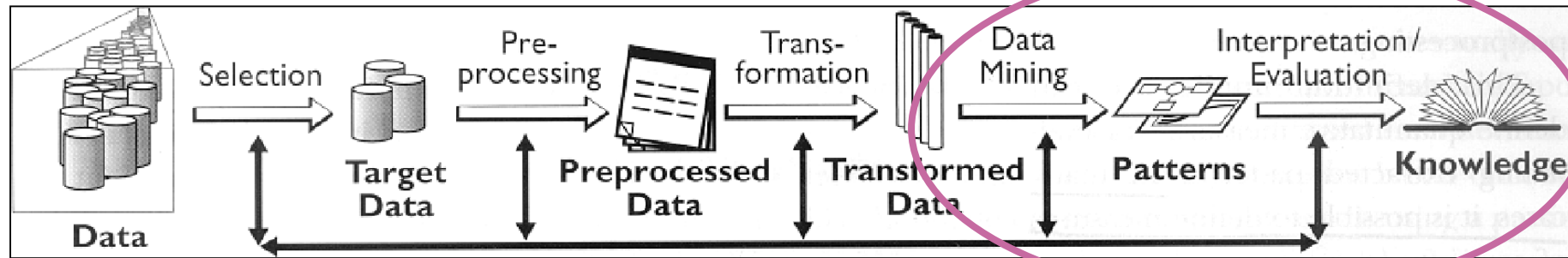
November 25, 2020

http://kt.ijs.si/petra_kralj/dmkd.html

Previously

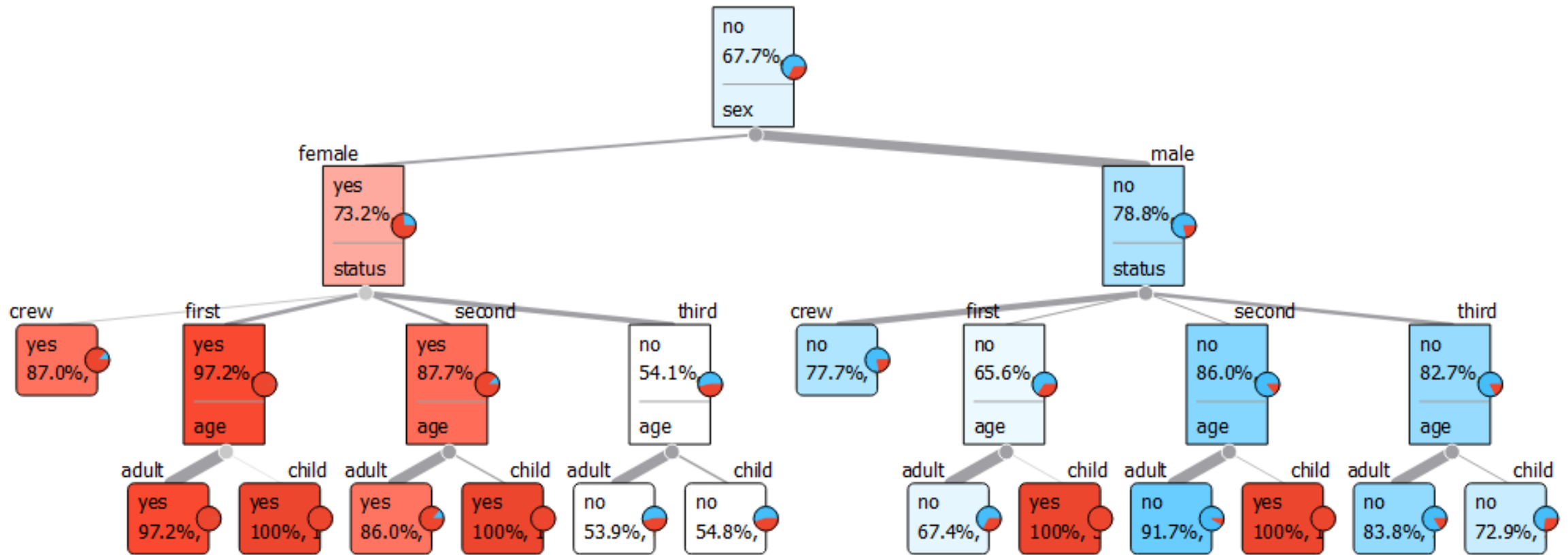
- **Data**, data types
- **Classification** with decision trees (root, leaves, rules, entropy, info gain, TDIDT, ID3)
- Classification: train – test (evaluate) - apply
- **Decision tree** example (on blackboard)
- Decision tree language bias (Orange workflow)
- **Evaluation:**
 - Methods: train-test, leave-one-out, randomized sampling,...
 - Metrics: accuracy, confusion matrix, precision, recall, F1,...

Keywords



- Data
 - Attribute, example, attribute-value data, target variable, class, discretization, market basket data
- Algorithms
 - Decision tree induction, ID3, entropy, information gain, overfitting, Occam's razor, model pruning, naïve Bayes classifier, KNN, association rules, support, confidence, classification rules, Laplace estimate, numeric prediction, regression tree, model tree, hierarchical clustering, dendrogram, k-means clustering, centroid, DB-scan, silhouette coefficient, Apriori, heuristics vs. exhaustive search, predictive vs. descriptive DM, language bias, artificial neural networks, deep learning, backpropagation,...
- Evaluation
 - Train set, test set, accuracy, confusion matrix, cross validation, true positives, false positives, ROC space, AUC, error, precision, recall, F1, MSE, RMSE, rRMSE, support, confidence

High precision and/or high recall?



Probabilistic classification

A **probabilistic** classifier is a classifier that is able to predict, given an observation of an input, a **probability** distribution over a set of classes, rather than only outputting the most likely class that the observation should belong to.

$$p(C_k \mid x_1, \dots, x_n)$$



Naïve Bayes Classifier

Basic probability refresh

- Probability of A

$$P(A)$$

- Independence

$$P(A \cap B) = P(A)P(B)$$

$$P(A|B) = P(A)$$

$$P(B|A) = P(B)$$

- Conditional probability

$$P(A|B) = P(A, B)/P(B)$$

- Bayes' Rule

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(A|B, C) = \frac{P(B|A, C)P(A|C)}{P(B|C)}$$

The idea behind the Naïve Bayes Classifier

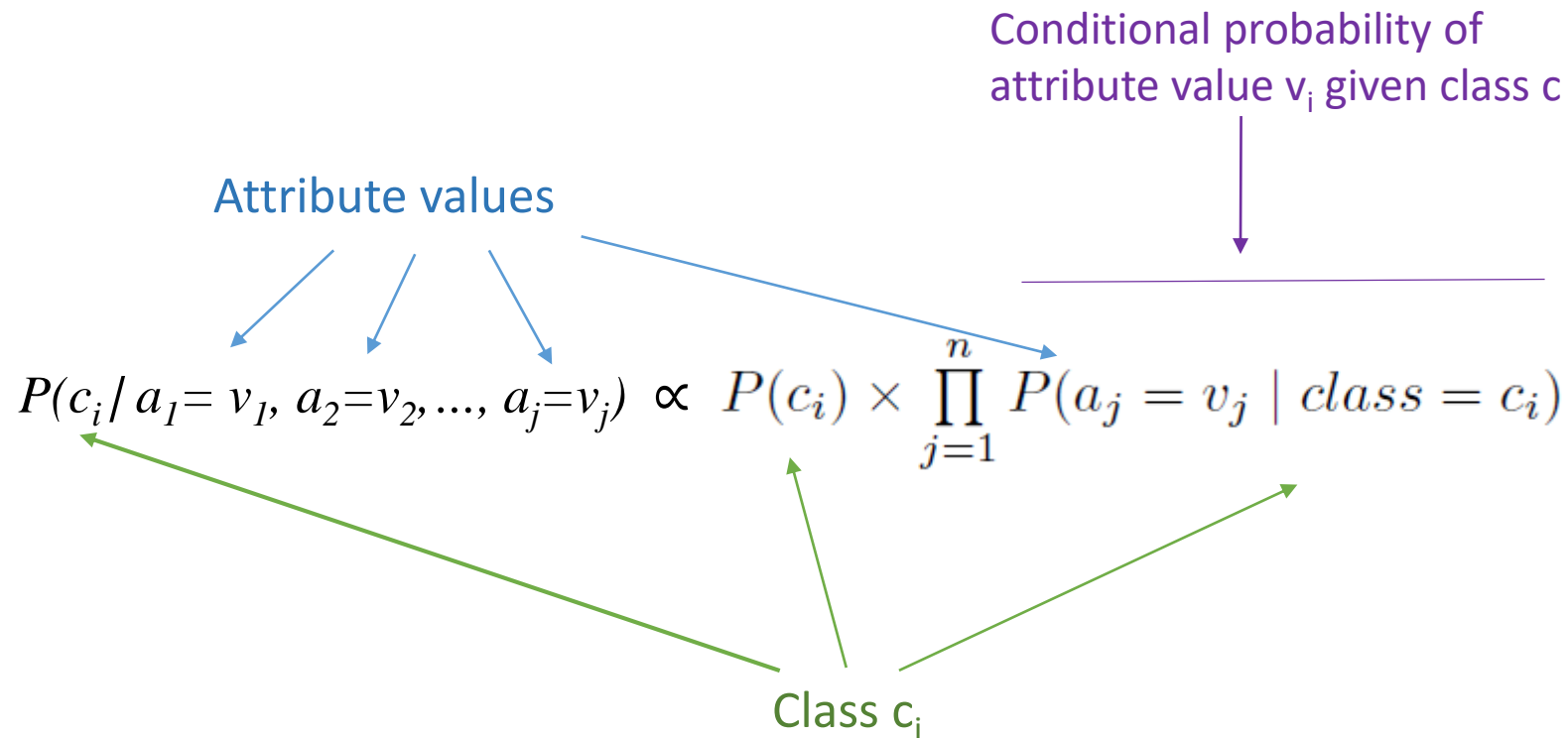
- We are interested in the probability of the class C given the attribute values $X_1, X_2, X_3, \dots, X_n$

$$P(C|X_1X_2 \dots X_n)$$

- We „**naively**“ assume that all attribute values $X_1, X_2, X_3, \dots, X_n$ are mutually independent, conditional on the category C

$$P(X_1X_2 \dots X_n|C) \approx P(X_1|C) \cdot P(X_2|C) \cdot \dots \cdot P(X_n|C)$$

Naïve Bayes Classifier



* where \propto denotes proportionality

* The results are not probabilities (they do not sum up to 1). The formula is simplified for easy implementation (and time complexity), while the results are proportional to the estimates of the probabilities of a class given the attribute values.

Exercise: Naïve Bayes Classifier

Color	Size	Time	Caught
black	large	day	YES
white	small	night	YES
black	small	day	YES
red	large	night	NO
black	large	night	NO
white	large	night	NO

$$P(c_i | a_1=v_1, a_2=v_2, \dots, a_j=v_j) \propto P(c_i) \times \prod_{j=1}^n P(a_j = v_j | class = c_i)$$

- Does the spider catch a white ant during the night?
- Does the spider catch the big black ant at daytime?

Exercise: Naïve Bayes Classifier

Does the spider catch a white ant during the night?

Color	Size	Time	Caught
black	large	day	YES
white	small	night	YES
black	small	day	YES
red	large	night	NO
black	large	night	NO
white	large	night	NO

$$P(c_i | a_1=v_1, a_2=v_2, \dots, a_j=v_j) \propto P(c_i) \times \prod_{j=1}^n P(a_j = v_j | class = c_i)$$

$$v_1 = \text{“Color = white”}$$

$$v_2 = \text{“Time = night”}$$

$$c_1 = YES$$

$$c_2 = NO$$

$$\begin{aligned} P(C_1|v_1, v_2) &= \\ &= P(YES|C = w, T = n) \\ &= P(YES) \cdot P(C = w|YES) \cdot P(T = n|YES) \\ &= \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} \\ &= \frac{1}{18} \end{aligned}$$

$$\begin{aligned} P(C_2|v_1, v_2) &= \\ &= P(NO|C = w, T = n) \\ &= P(NO) \cdot P(C = w|NO) \cdot P(T = n|NO) \\ &= \frac{1}{2} \cdot \frac{1}{3} \cdot 1 \\ &= \frac{1}{6} \end{aligned}$$

Exercise: Naïve Bayes Classifier

Does the spider catch the big black ant at daytime?

Color	Size	Time	Caught
black	large	day	YES
white	small	night	YES
black	small	day	YES
red	large	night	NO
black	large	night	NO
white	large	night	NO

$$P(c_i | a_1=v_1, a_2=v_2, \dots, a_j=v_j) \propto P(c_i) \times \prod_{j=1}^n P(a_j = v_j | class = c_i)$$

Ant 2: Color = black, Size = large, Time = day

$$v_1 = \text{"Color = black"} = \text{"C = b"}$$

$$v_2 = \text{"Size = large"} = \text{"S = l"}$$

$$v_3 = \text{"Time = day"} = \text{"T = d"}$$

$$c_1 = \text{YES}$$

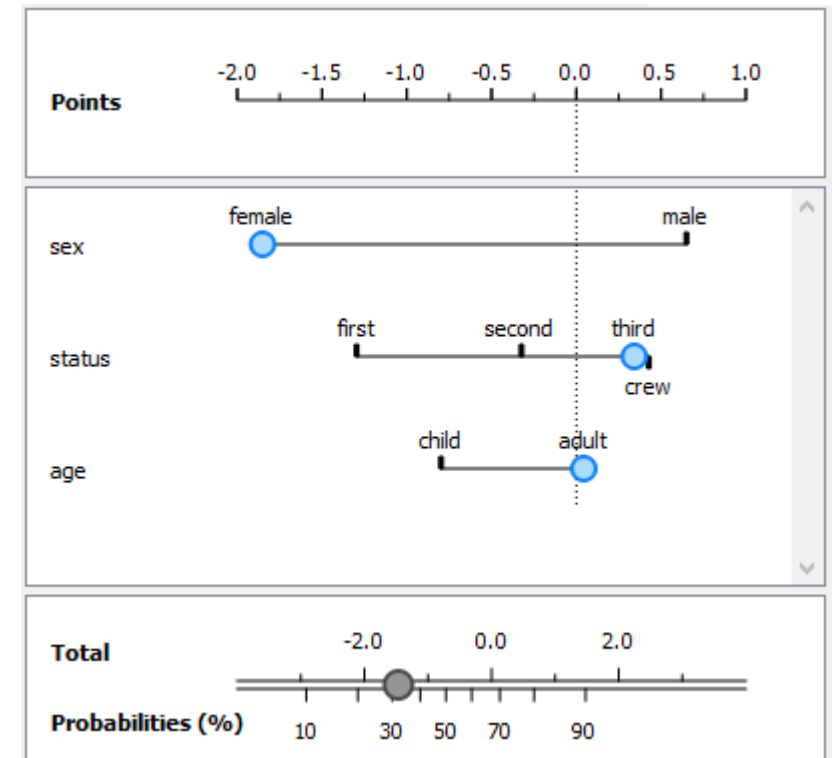
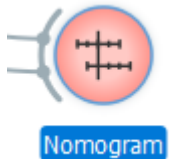
$$c_2 = \text{NO}$$

$$\begin{aligned} P(C_1 | v_1, v_2, v_3) &= \\ &= P(\text{YES} | C = b, S = l, T = d) \\ &= P(\text{YES}) \cdot P(C = b | \text{YES}) \cdot P(S = l | \text{YES}) \cdot P(T = d | \text{YES}) \\ &= \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{1}{3} \cdot \frac{2}{3} \\ &= \frac{4}{54} = \frac{2}{27} \end{aligned}$$

$$\begin{aligned} P(C_2 | v_1, v_2, v_3) &= \\ &= P(\text{NO} | C = b, S = l, T = d) \\ &= P(\text{NO}) \cdot P(C = b | \text{NO}) \cdot P(S = l | \text{NO}) \cdot P(T = d | \text{NO}) \\ &= \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{3}{3} \cdot 0 \\ &= 0 \end{aligned}$$

Use of Naïve Bayes

- Frequently used in practice
 - Medical diagnosis
 - The attributes are inherently chosen to be as independent as possible
 - NB is not sensitive to missing data
 - Simple text classification(features are words)
 - Classification of news into categories
 - Spam detection
 -
- Why?
 - Simple
 - Not sensitive to missing values
 - Uses all the available data
 - Very few parameters
 - Visualization with nomograms



Probability Estimation



Estimating probability

- In machine learning we often estimate probabilities from small samples of data and their subsets:
 - In the 5th depth of a decision tree we have just about 1/32 of all training examples.
- Estimate the probability based on the amount of evidence and of the prior probability
 - Coin flip: prior probability 50% - 50%
 - One coin flip does not make us believe that the probability of heads is 100%
 - More evidence can make us suspect that the coin is biased

Estimating probability

Relative frequency

- $P(c) = n(c) / N$
- A disadvantage of using relative frequencies for probability estimation arises with small sample sizes, especially if the probabilities are either very close to zero, or very close to one.
- In our spider example:
$$P(\text{Time}=\text{day} \mid \text{caught}=\text{NO}) =$$
$$= 0/3 = 0$$

$n(c)$... number of examples where c is true

N ... number of all examples

k ... number of possible events

Relative frequency vs. Laplace estimate

Relative frequency

- $P(c) = n(c) / N$
- A disadvantage of using relative frequencies for probability estimation arises with small sample sizes, especially if the probabilities are either very close to zero, or very close to one.
- In our spider example:
$$P(\text{Time}=\text{day} \mid \text{caught}=\text{NO}) = 0/3 = 0$$

$n(c)$... number of examples where c is true

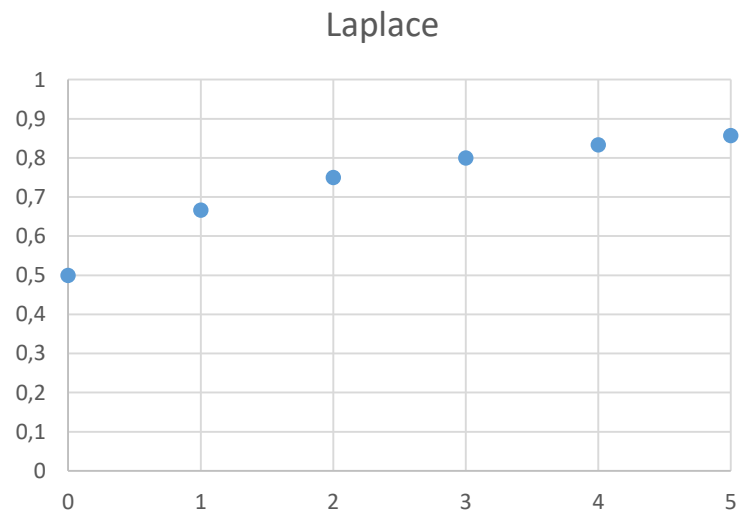
N ... number of all examples

k ... number of possible events

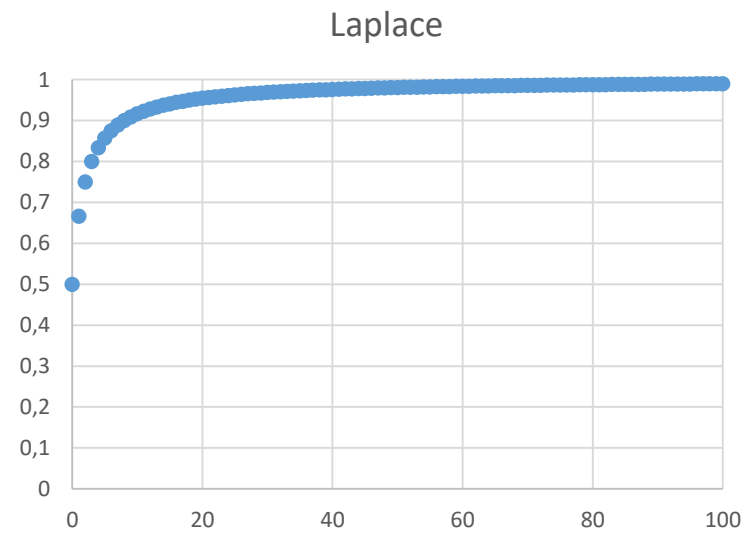
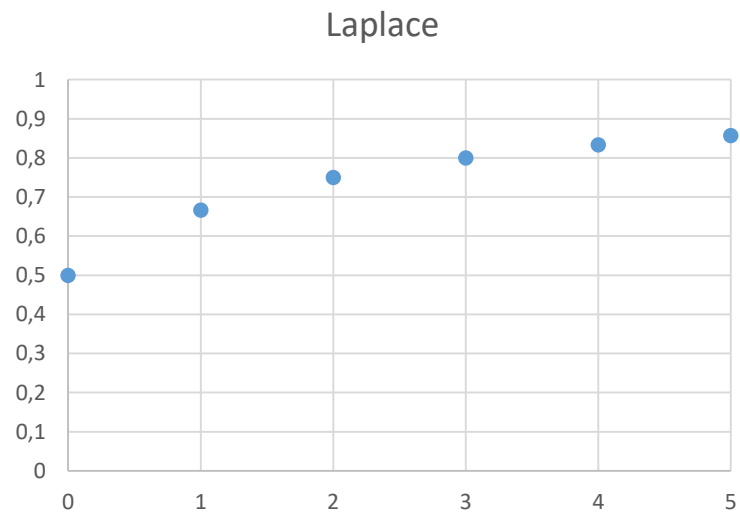
Laplace estimate

- Assumes uniform prior distribution over the probabilities for each possible event
- $P(c) = (n(c) + 1) / (N + k)$
- In our spider example: $P(\text{Time}=\text{day} \mid \text{caught}=\text{NO}) = (0+1)/(3+2) = 1/5$
- With lots of evidence it approximates relative frequency
- If there were 300 cases when the spider didn't catch ants at night: $P(\text{Time}=\text{day} \mid \text{caught}=\text{NO}) = (0+1)/(300+2) = 1/302 = 0.003$
- With Laplace estimate probabilities can never be 0.

Laplace estimate



Laplace estimate



Homework

- Compare the Naïve Bayes classifier with decision trees.
- How do we evaluate the Naïve Bayes classifier? Methods, metrics.
- Estimate the probabilities of C1 and C2 in the table below by relative frequency and Laplace estimate.

Number of events		Relative frequency		Laplace estimate	
Class C1	Class C2	P(C1)	P(C2)	P(C1)	P(C2)
0	2				
12	88				
12	988				
120	880				

Literature

- Max Bramer: Principles of data mining (2007)

- 2. Introduction to Classification: Naive Bayes and Nearest Neighbour

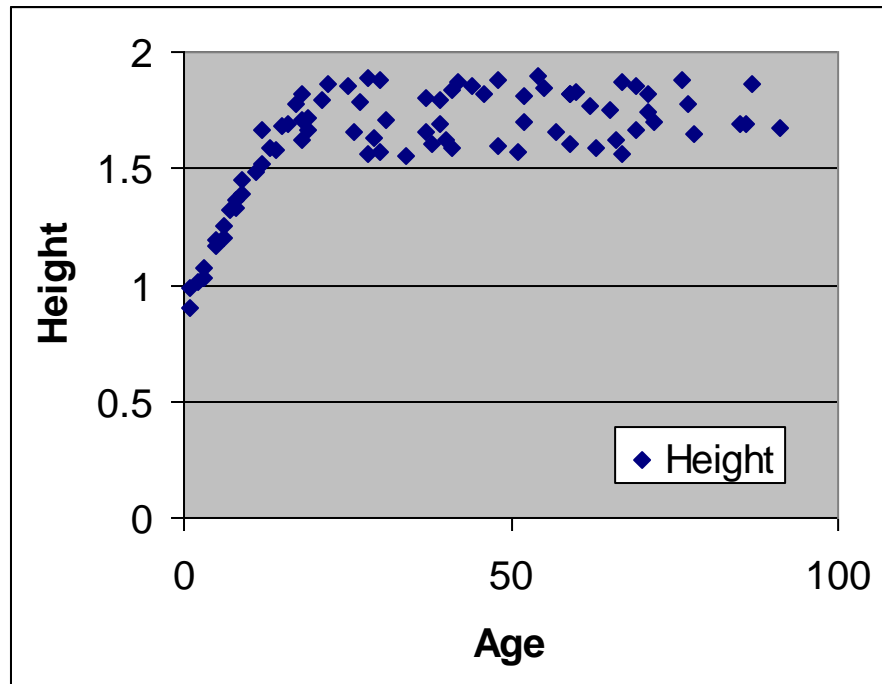
On pg. 30, there is a mistake where it says “making the assumption that the attributes are independent” ... it should be “conditionally independent given the class”. Refer to https://en.wikipedia.org/wiki/Naive_Bayes_classifier



Numeric prediction

Example

- data about 80 people: Age and Height



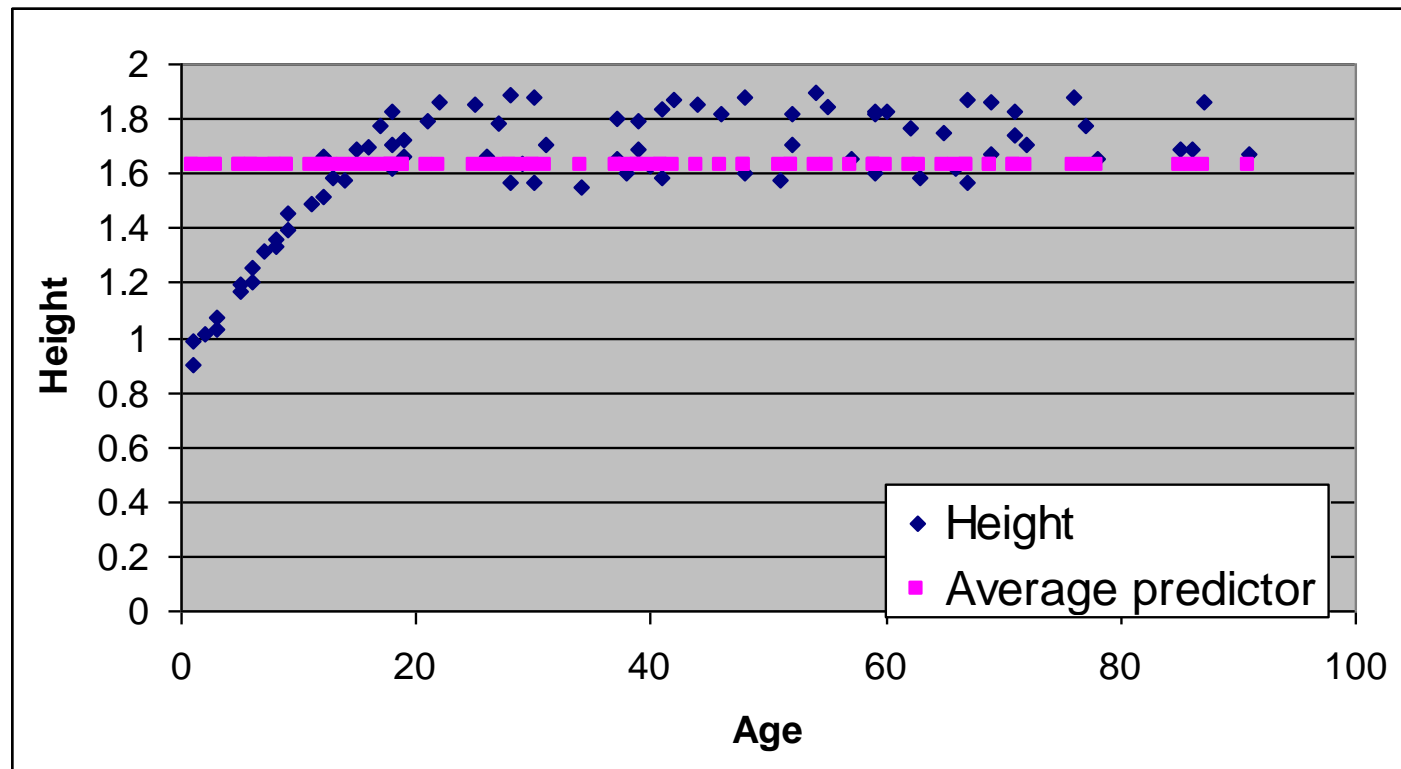
Age	Height
3	1.03
5	1.19
6	1.26
9	1.39
15	1.69
19	1.67
22	1.86
25	1.85
41	1.59
48	1.60
54	1.90
71	1.82
...	...

Test set

Age	Height
2	0.85
10	1.4
35	1.7
70	1.6

Baseline numeric predictor

- Average of the target variable



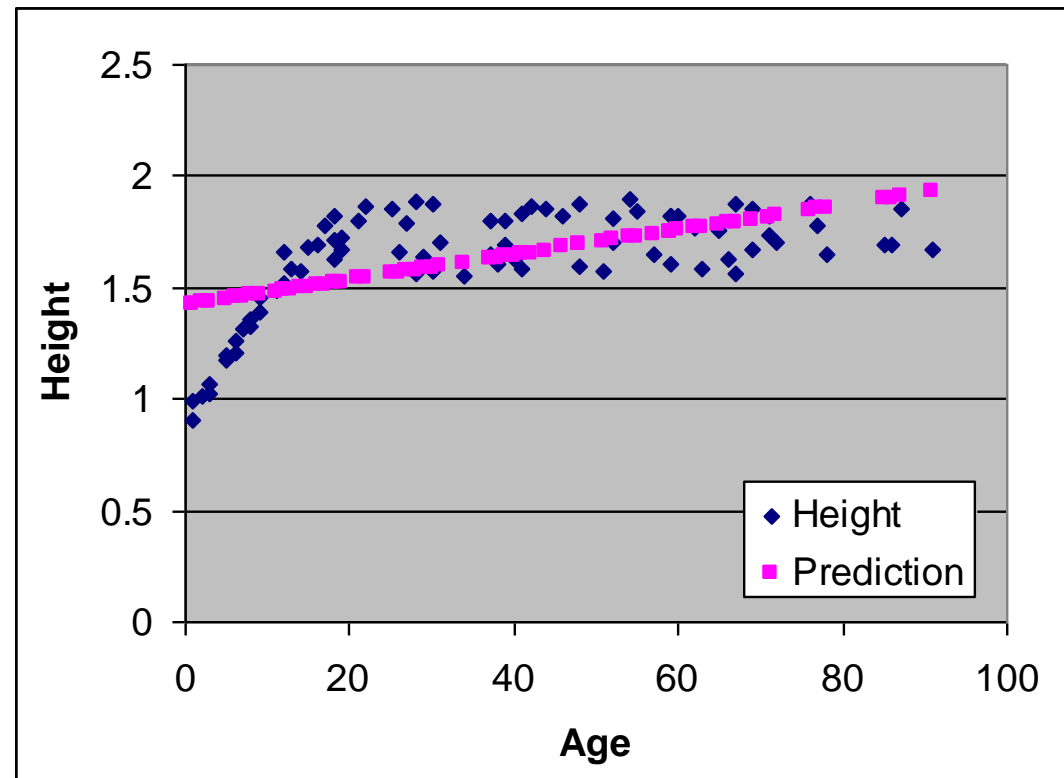
Baseline predictor: prediction

Average of the target variable is 1.63

Age	Height	Baseline
2	0.85	
10	1.4	
35	1.7	
70	1.6	

Linear Regression Model

$$\text{Height} = 0.0056 * \text{Age} + 1.4181$$

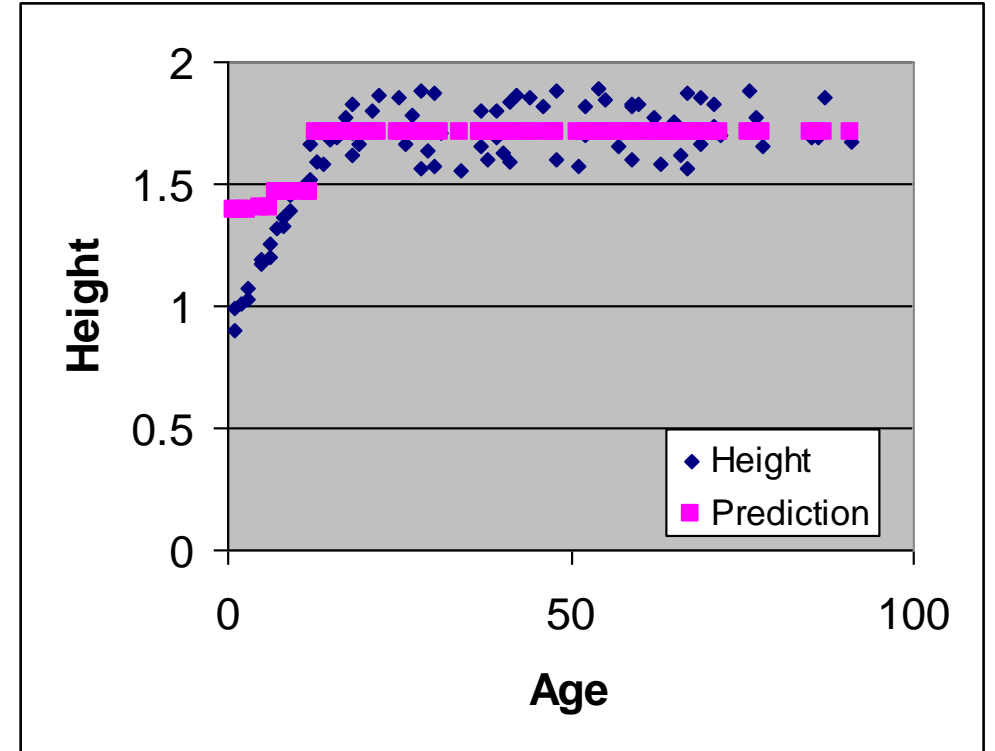
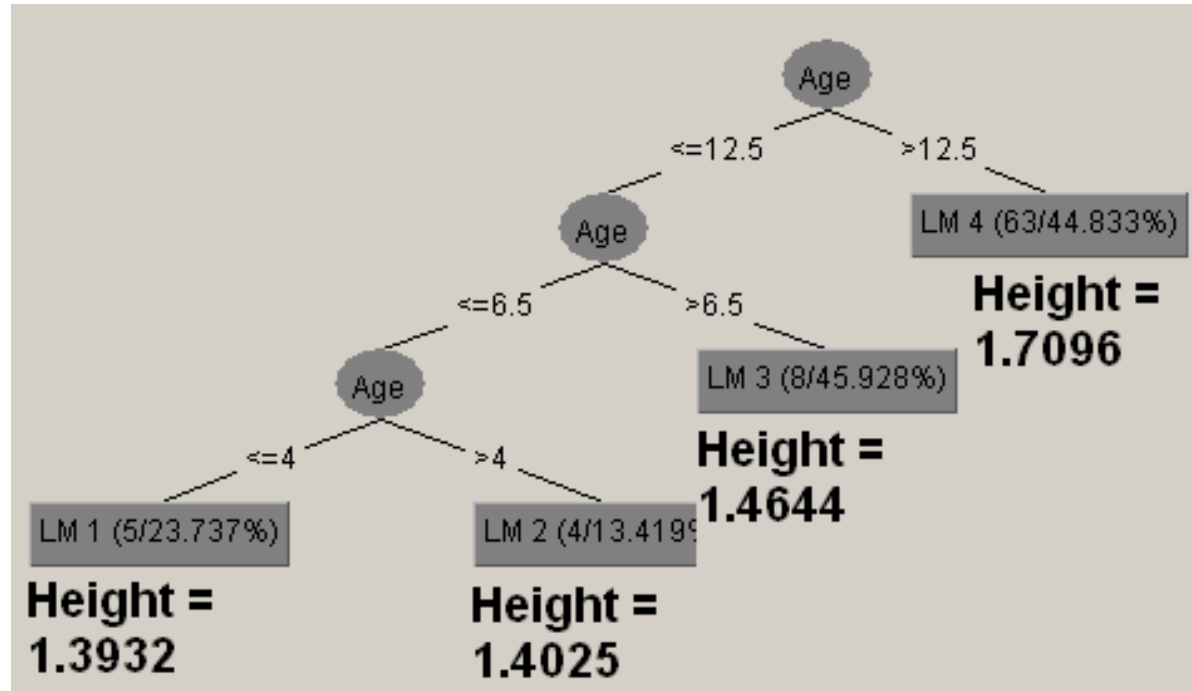


Linear Regression: prediction

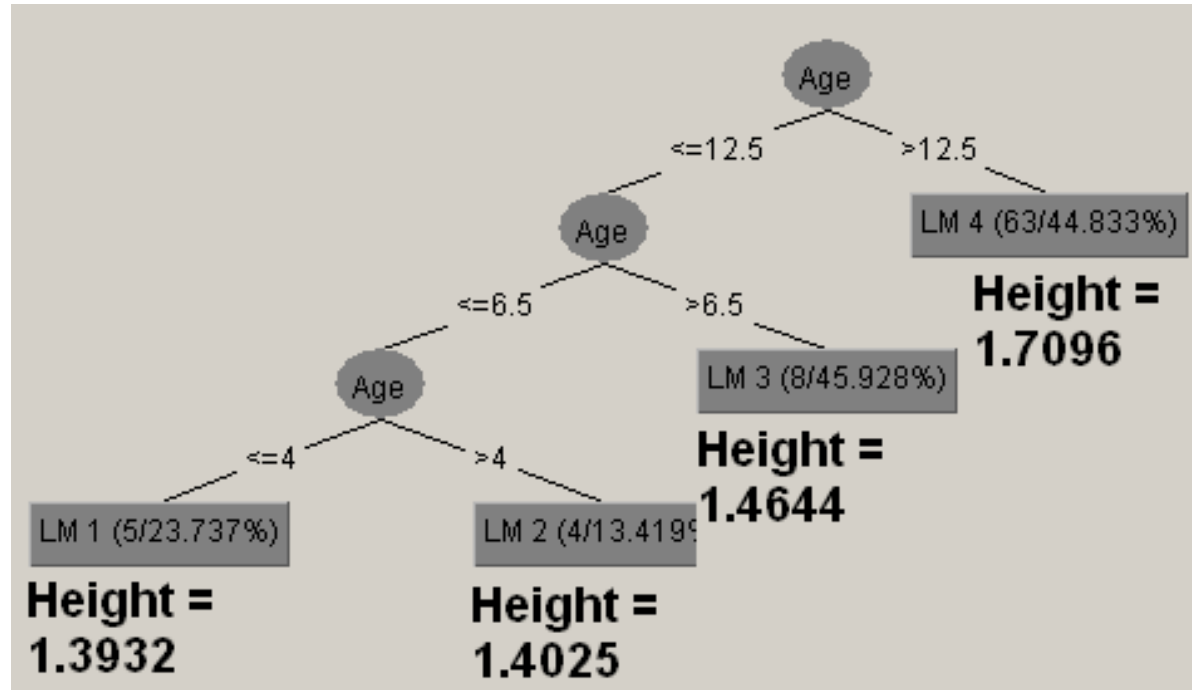
$$\text{Height} = 0.0056 * \text{Age} + 1.4181$$

Age	Height	Linear regression
2	0.85	
10	1.4	
35	1.7	
70	1.6	

Regression tree

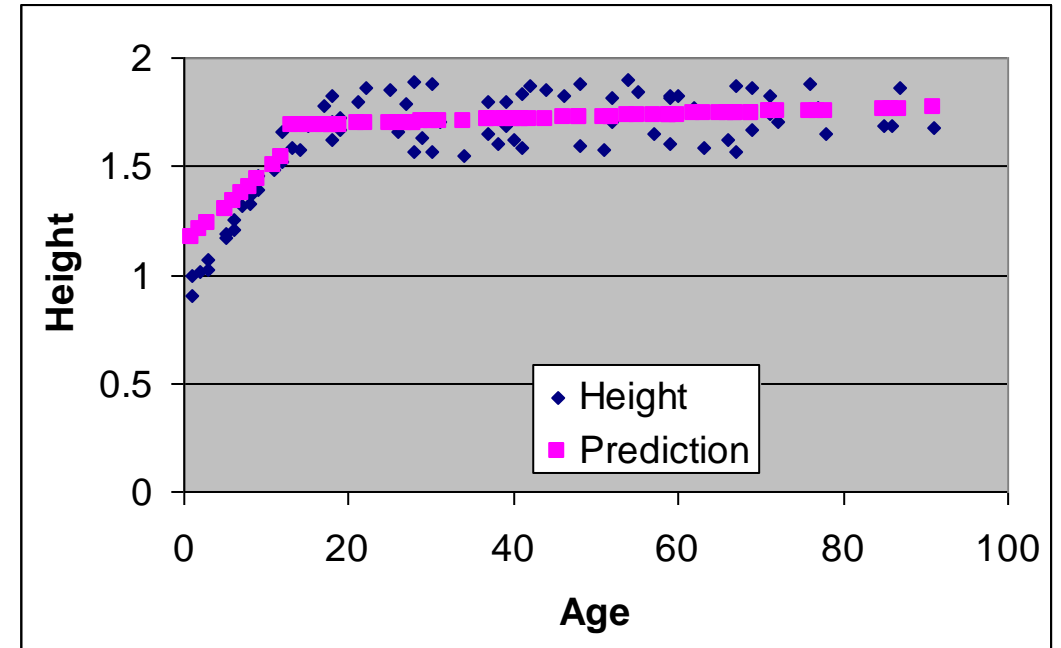


Regression tree: prediction

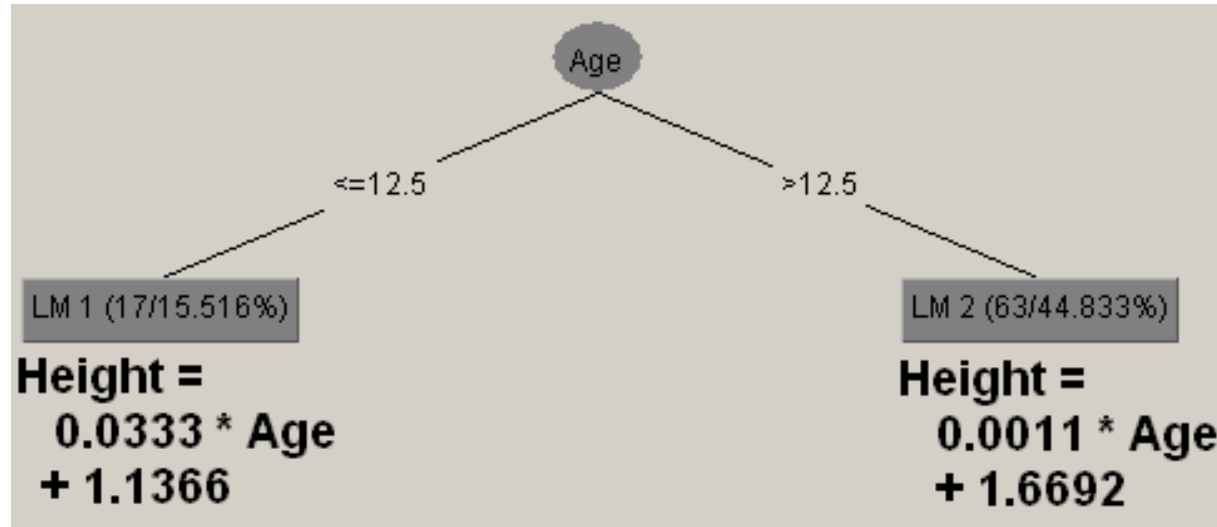


Age	Height	Regression tree
2	0.85	
10	1.4	
35	1.7	
70	1.6	

Model tree



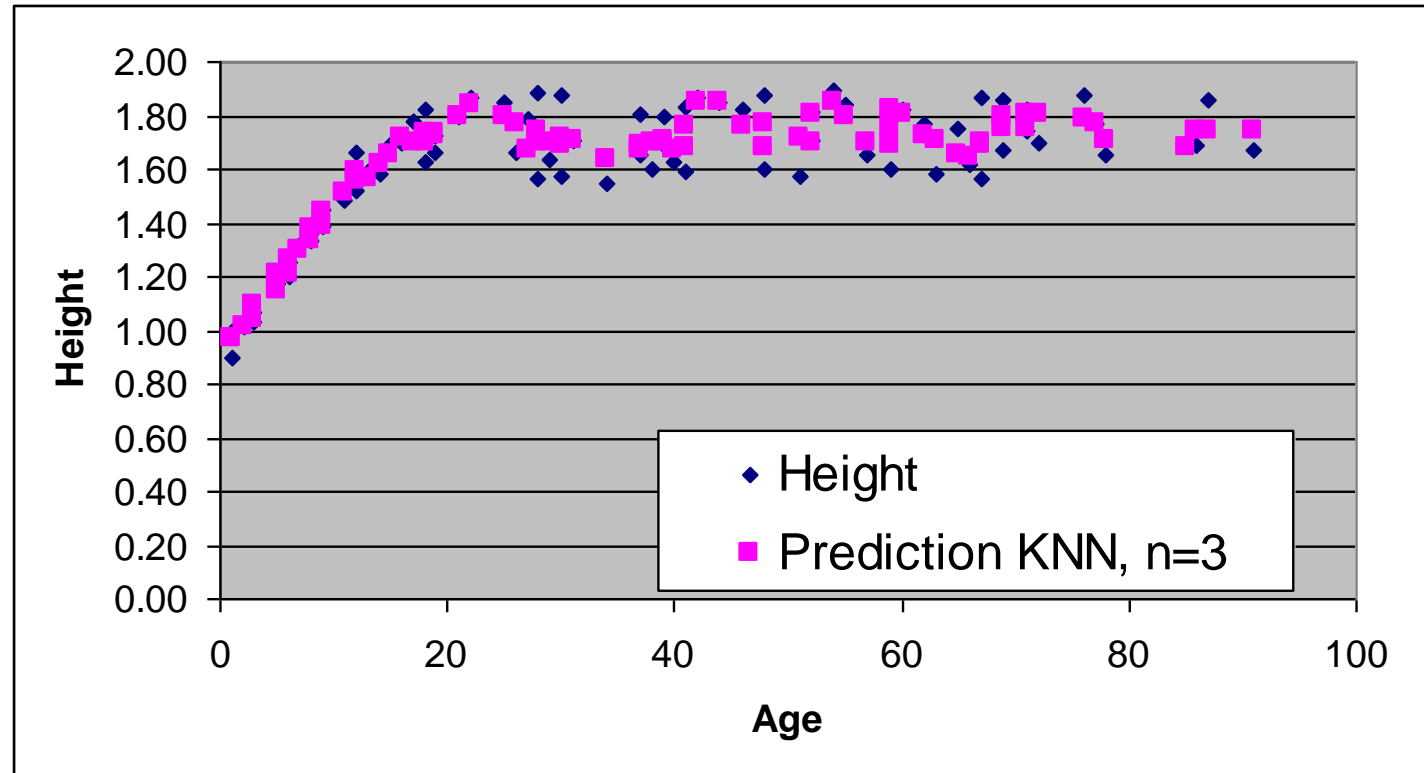
Model tree: prediction



Age	Height	Model tree
2	0.85	
10	1.4	
35	1.7	
70	1.6	

KNN – K nearest neighbors

- Looks at K closest examples (by non-target attributes) and predicts the average of their target variable
- In this example, K=3



KNN prediction

Age	Height
1	0.90
1	0.99
2	1.01
3	1.03
3	1.07
5	1.19
5	1.17

Age	Height	kNN
2	0.85	
10	1.4	
35	1.7	
70	1.6	

KNN prediction

Age	Height
8	1.36
8	1.33
9	1.45
9	1.39
11	1.49
12	1.66
12	1.52
13	1.59
14	1.58

Age	Height	kNN
2	0.85	
10	1.4	
35	1.7	
70	1.6	

KNN prediction

Age	Height
30	1.57
30	1.88
31	1.71
34	1.55
37	1.65
37	1.80
38	1.60
39	1.69
39	1.80

Age	Height	kNN
2	0.85	
10	1.4	
35	1.7	
70	1.6	

KNN prediction

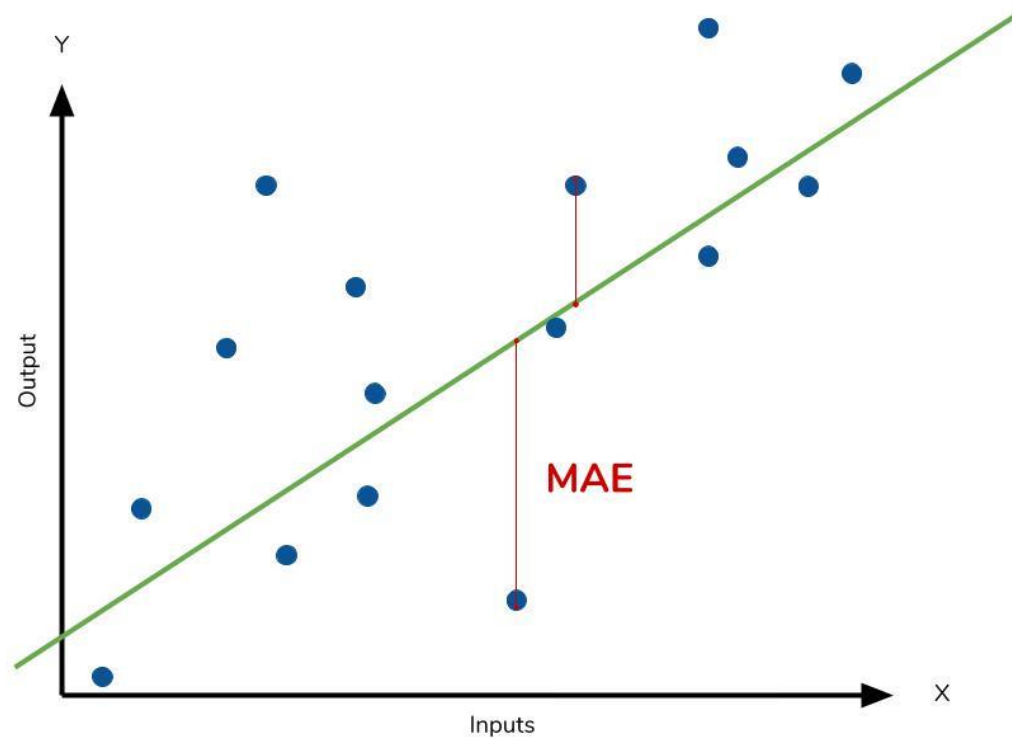
Age	Height
67	1.56
67	1.87
69	1.67
69	1.86
71	1.74
71	1.82
72	1.70
76	1.88

Age	Height	kNN
2	0.85	
10	1.4	
35	1.7	
70	1.6	

Which predictor is the best?

Age	Height	Baseline	Linear regression	Regression tree	Model tree	kNN
2	0.85	1.63	1.43	1.39	1.20	1.00
10	1.4	1.63	1.47	1.46	1.47	1.44
35	1.7	1.63	1.61	1.71	1.71	1.67
70	1.6	1.63	1.81	1.71	1.75	1.77

MAE: Mean absolute error



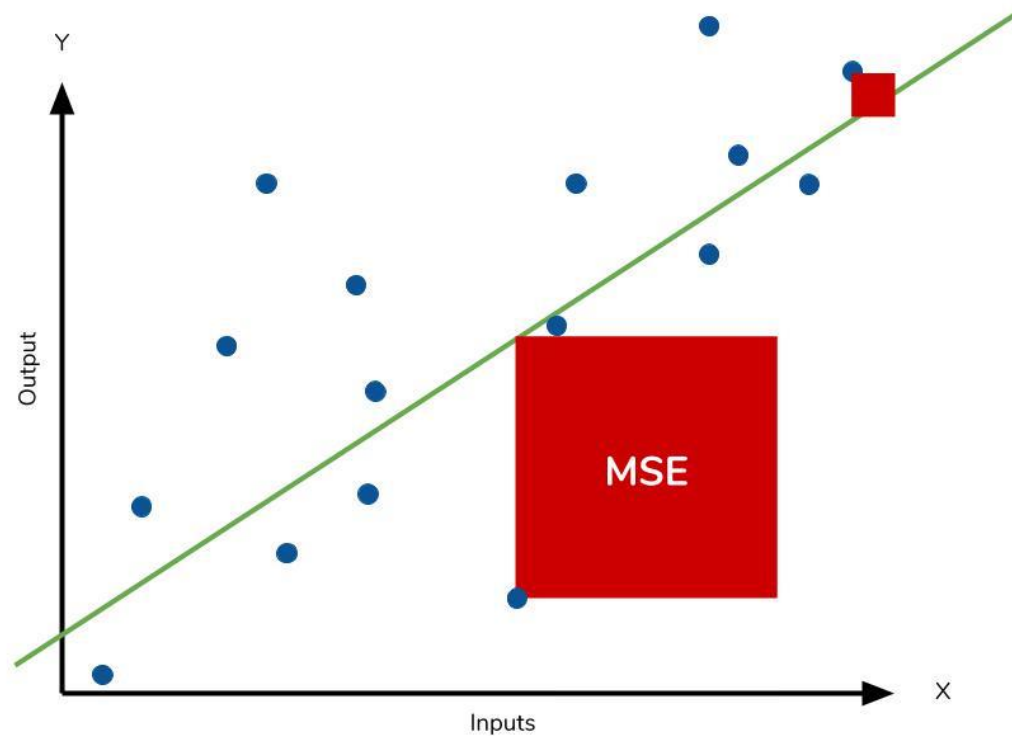
$$MAE = \frac{1}{n} \sum \left| y - \hat{y} \right|$$

Annotations for the equation:

- Divide by the total number of data points (points to $\frac{1}{n}$)
- Actual output value (points to y)
- Predicted output value (points to \hat{y})
- Sum of (points to \sum)
- The absolute value of the residual (points to $|y - \hat{y}|$)

The average difference between the predicted and the actual values.
The units are the same as the units in the target variable.

MSE: Mean squared error



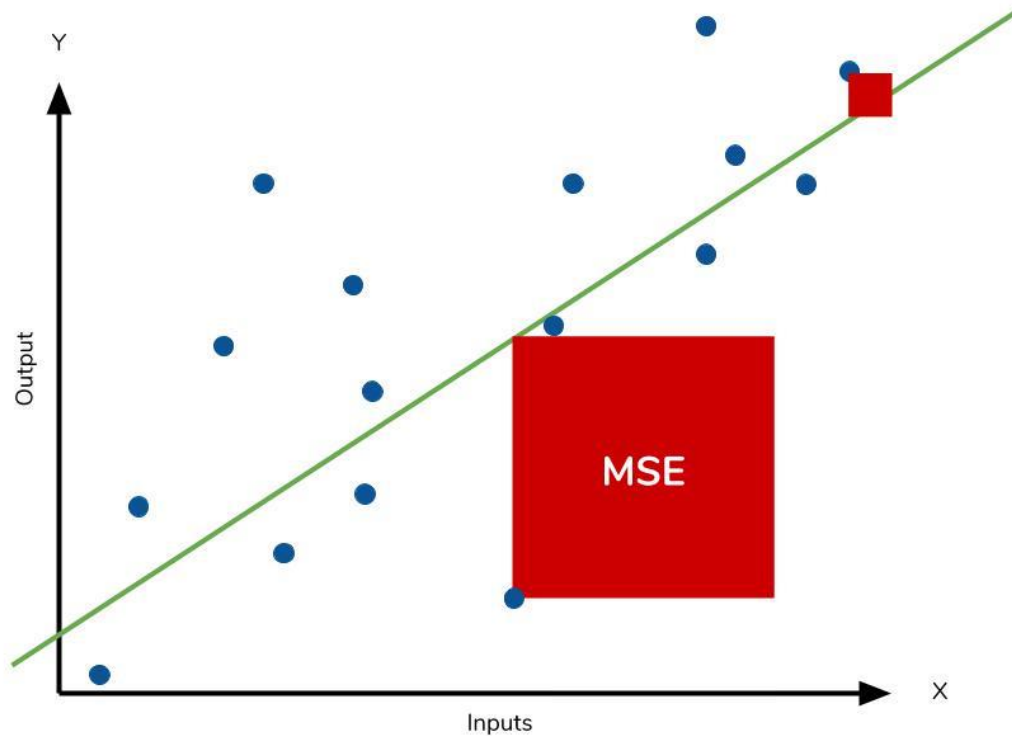
$$MSE = \frac{1}{n} \sum \underbrace{\left(y - \hat{y} \right)^2}_{\substack{\text{The square of the difference} \\ \text{between actual and} \\ \text{predicted}}}$$

Mean squared error measures the average squared difference between the estimated values and the actual value.

Weights large errors more heavily than small ones.

The units of the errors are squared.

RMSE: Root mean square error

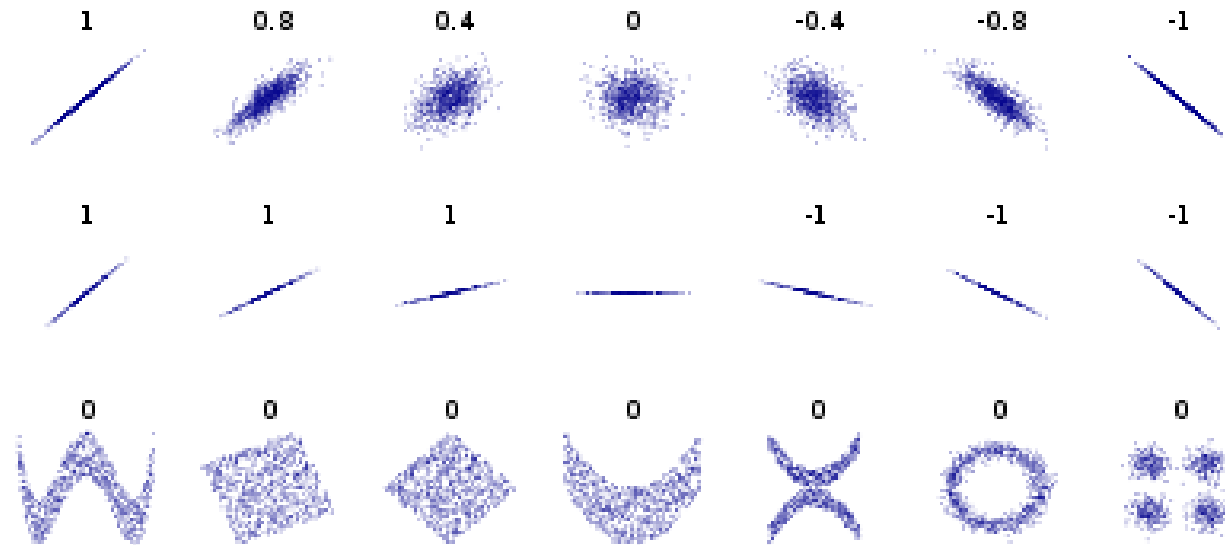


$$RMSE = \sqrt{MSE}$$

Taking the square root of MSE yields the root-mean-square error (RMSE), which has the same units as the quantity being estimated.

Correlation coefficient

- Pearson correlation coefficient is a statistical formula that measures the strength between variables and relationships.



Similar to confusion matrix in the classification case.
No unit.

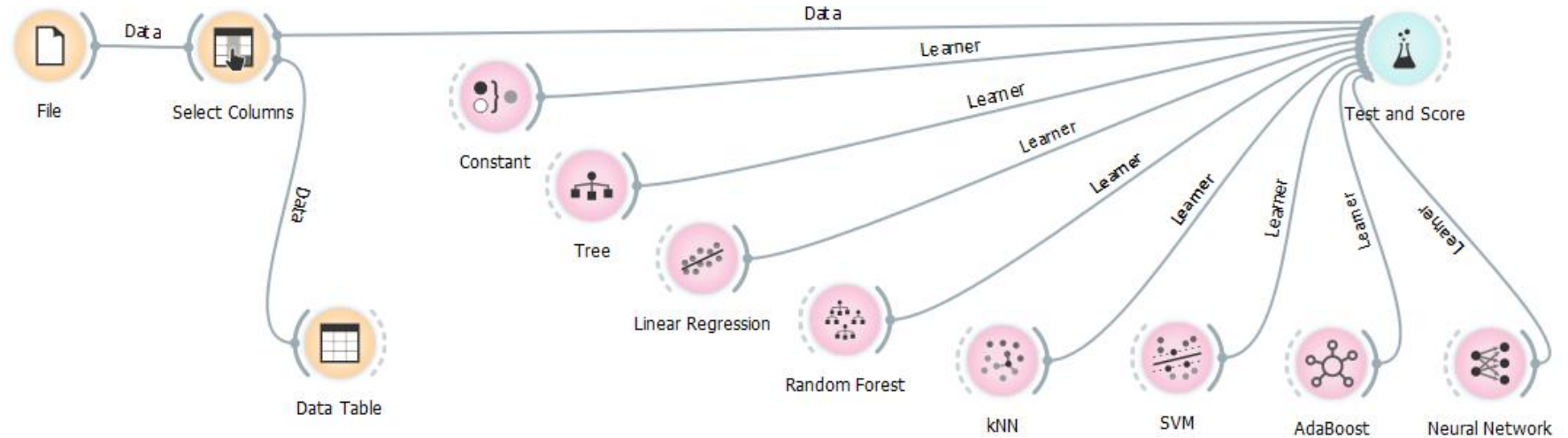
Performance measures for numeric prediction

Performance measure	Formula
mean-squared error	$\frac{(p_1 - a_1)^2 + \dots + (p_n - a_n)^2}{n}$
root mean-squared error	$\sqrt{\frac{(p_1 - a_1)^2 + \dots + (p_n - a_n)^2}{n}}$
mean absolute error	$\frac{ p_1 - a_1 + \dots + p_n - a_n }{n}$
relative squared error	$\frac{(p_1 - a_1)^2 + \dots + (p_n - a_n)^2}{(a_1 - \bar{a})^2 + \dots + (a_n - \bar{a})^2}, \text{ where } \bar{a} = \frac{1}{n} \sum_i a_i$
root relative squared error	$\sqrt{\frac{(p_1 - a_1)^2 + \dots + (p_n - a_n)^2}{(a_1 - \bar{a})^2 + \dots + (a_n - \bar{a})^2}}$
relative absolute error	$\frac{ p_1 - a_1 + \dots + p_n - a_n }{ a_1 - \bar{a} + \dots + a_n - \bar{a} }$
correlation coefficient	$\frac{S_{PA}}{\sqrt{S_P S_A}}, \text{ where } S_{PA} = \frac{\sum_i (p_i - \bar{p})(a_i - \bar{a})}{n-1},$ $S_P = \frac{\sum_i (p_i - \bar{p})^2}{n-1}, \text{ and } S_A = \frac{\sum_i (a_i - \bar{a})^2}{n-1}$

* p are predicted values and a are actual values.

Numeric prediction in Orange

Models



Metrics

- MSE – mean squared error
- RMSE – root mean squared error
- MAE – mean absolute error
- R^2 – correlation coefficient

Evaluation Results				
Model	MSE	RMSE	MAE	R2
Constant	0.055	0.236	0.175	-0.005
Linear Regression	0.033	0.181	0.142	0.405
SVM	0.032	0.179	0.128	0.423
Neural Network	0.026	0.161	0.118	0.533
kNN	0.011	0.107	0.086	0.794
Tree	0.010	0.100	0.073	0.817
AdaBoost	0.004	0.066	0.057	0.922
Random Forest	0.003	0.057	0.048	0.940

Numeric prediction	Classification
Data: attribute-value description	
Target variable: Continuous	Target variable: Categorical (nominal)
Evaluation: cross validation, separate test set, ...	
Error: MSE, MAE, RMSE, ...	Error: 1-accuracy
Algorithms: Linear regression, regression trees,...	Algorithms: Decision trees, Naïve Bayes, ...
Baseline predictor: Mean of the target variable	Baseline predictor: Majority class

Homework

- Read

Loh, Wei-Yin. "Classification and regression trees." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1.1 (2011): 14-23.

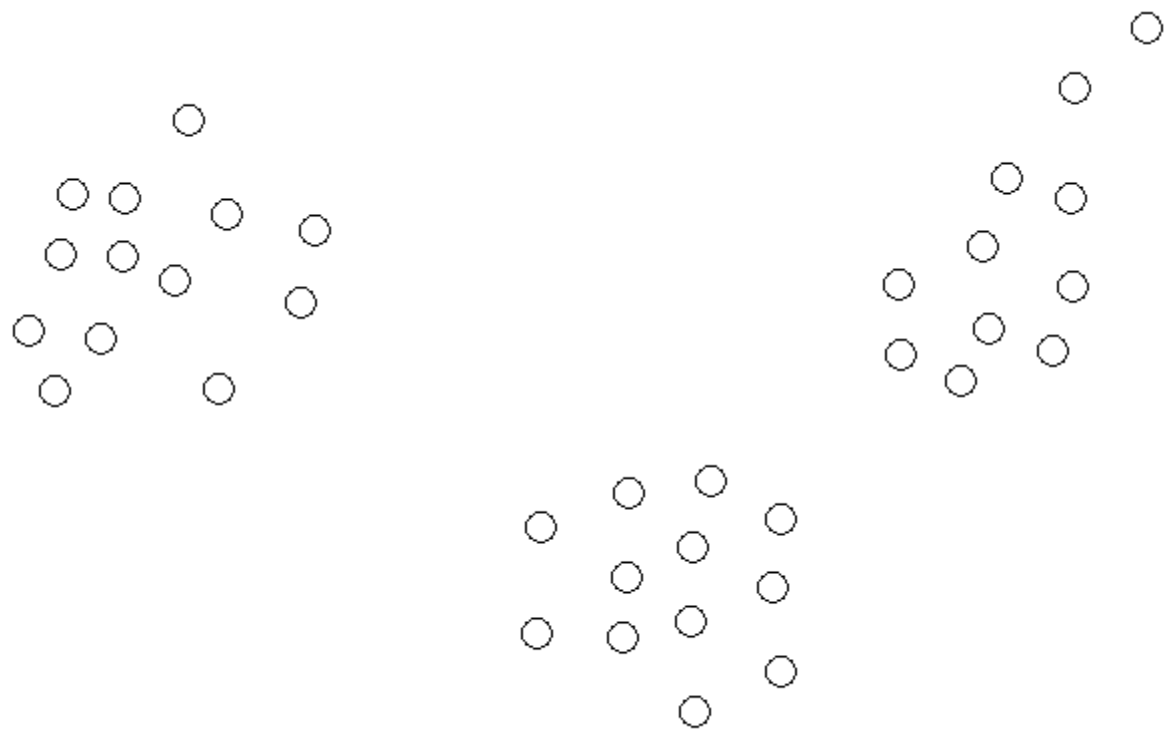
<https://onlinelibrary.wiley.com/doi/full/10.1002/widm.8>

- Compare decision and regression trees.
- Rules of thumb for choosing the k parameter of KNN.

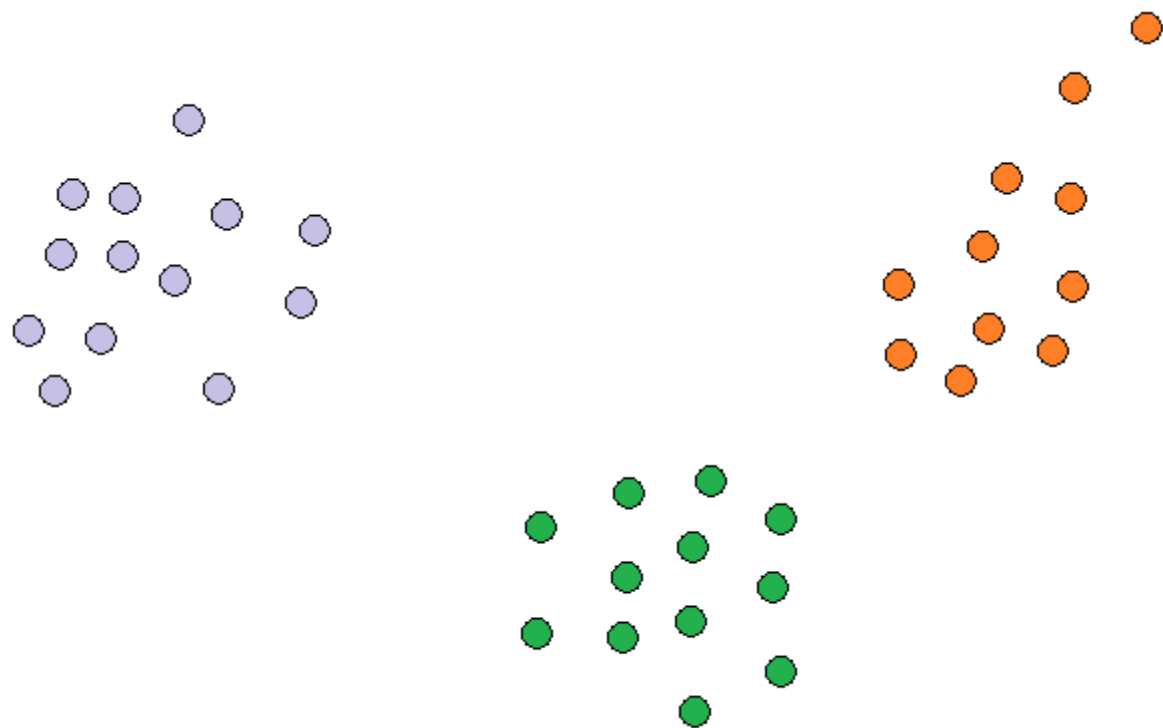
The background consists of a repeating pattern of circular portraits of Stefan Jovanović. Each portrait is set within a circular frame that contains the name 'Stefan Jovanović' and the mathematical formula $J = \sigma \cdot T^4$. The portraits and text are rendered in a light, semi-transparent style against a light blue background.

Clustering

Clustering



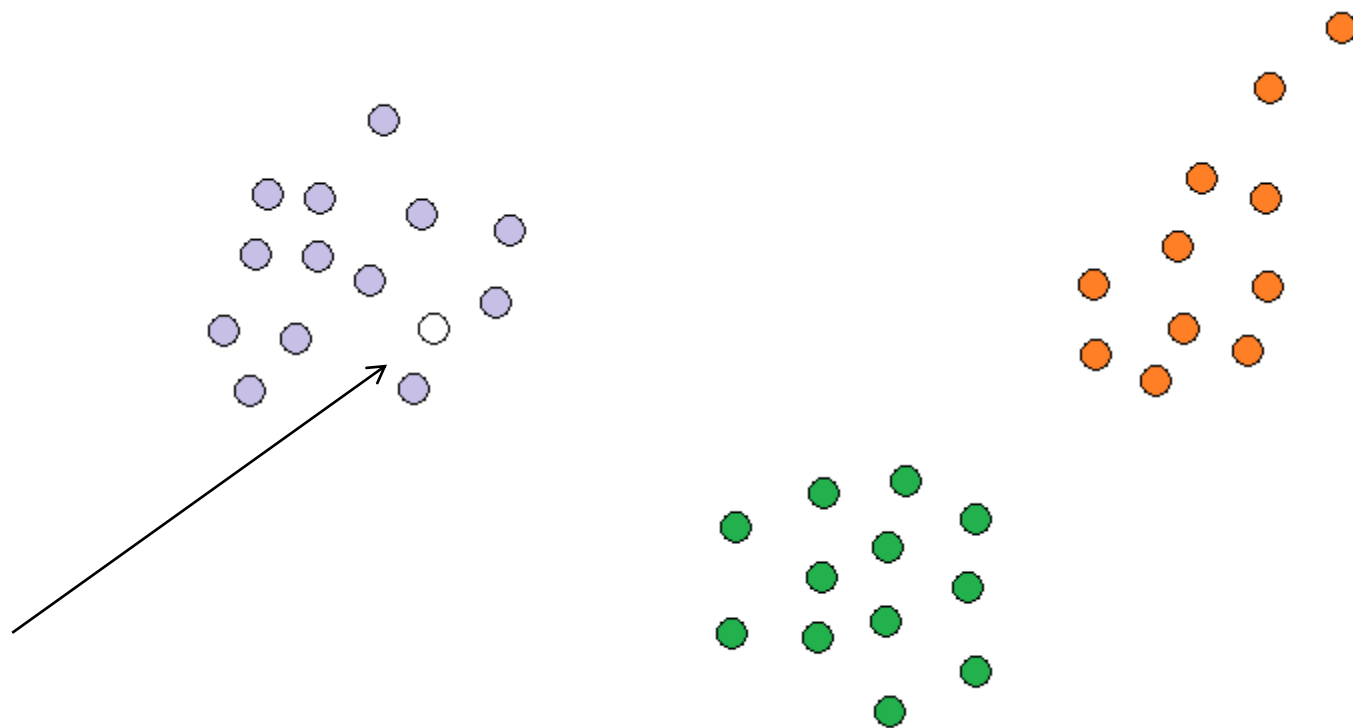
Clustering



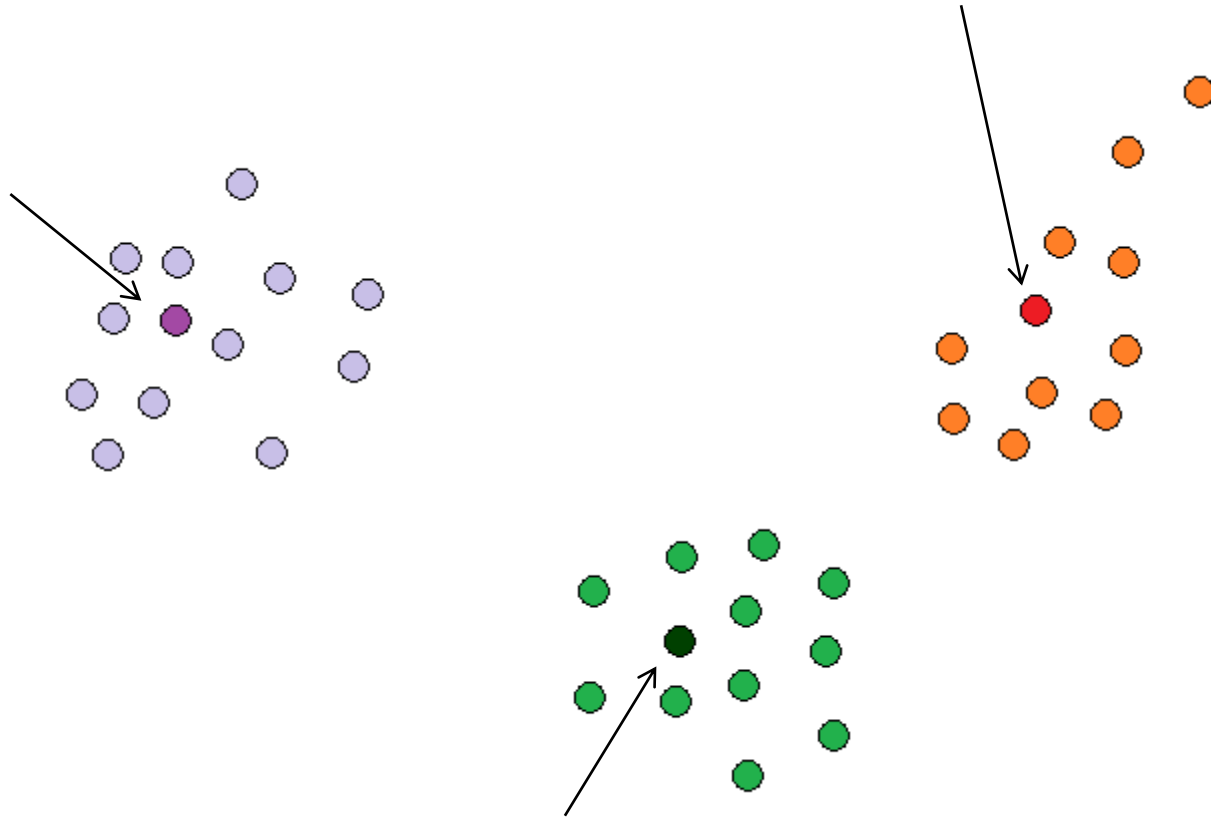
Clustering

- ... is the process of grouping the data instances into clusters so that objects within a cluster have high similarity but are very dissimilar to objects in other clusters.
- Wish list:
 - Identity clusters irrespective of their shapes
 - Scalability
 - Ability to deal with noisy data
 - Insensitivity to the order of input records

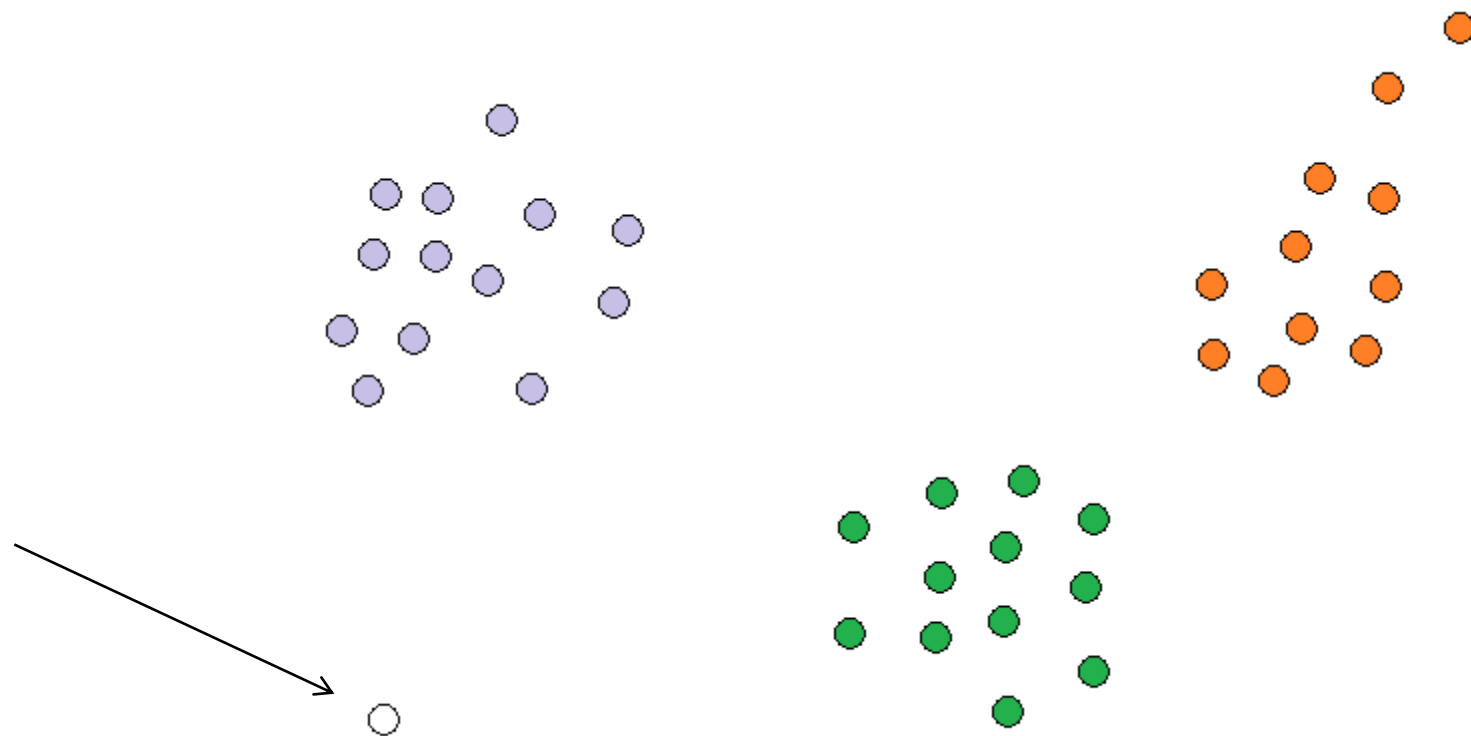
Unsupervised classification



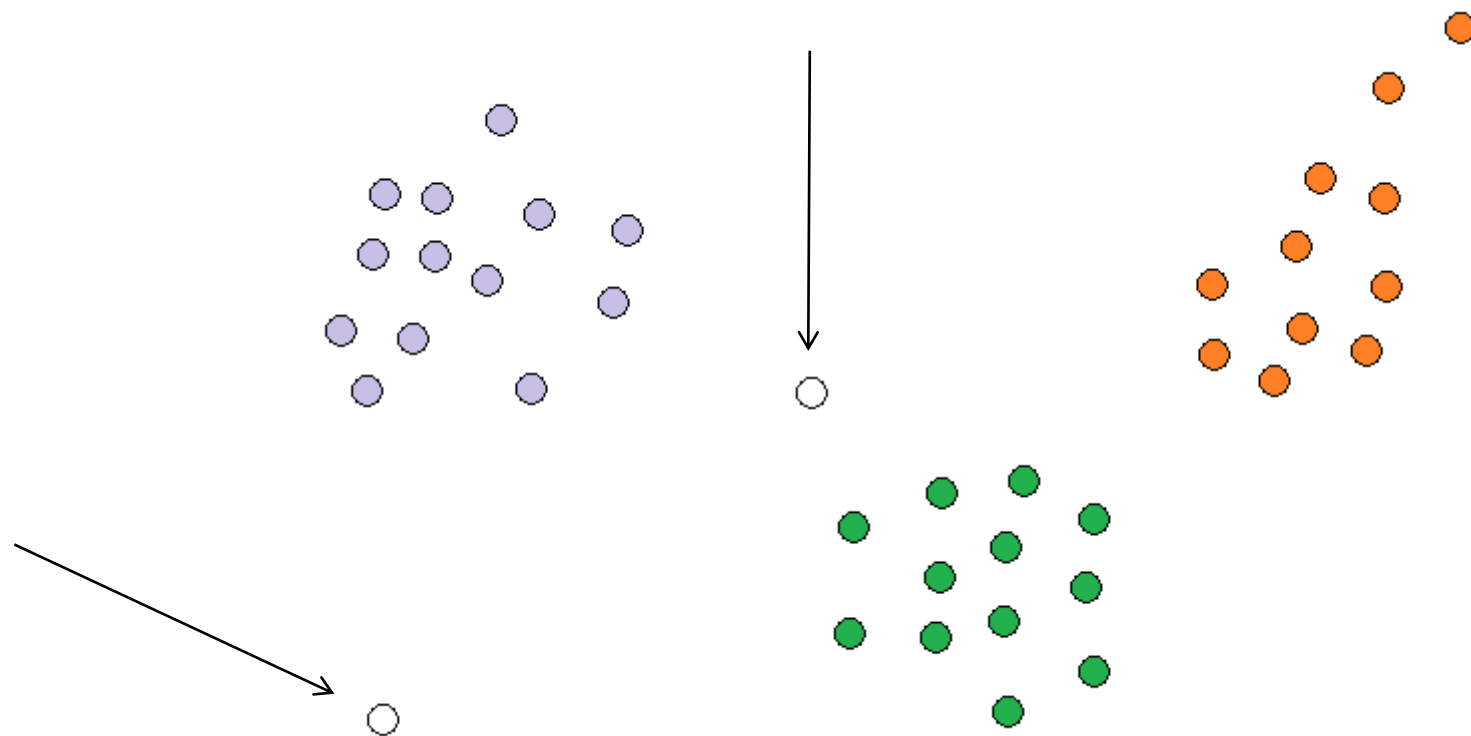
Data summarization: centroid, medoid



Outlier detection



Outlier detection

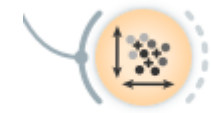


Applications

- Data mining
 - Unsupervised classification
 - Data summarization
 - Outlier analysis
 - ...
- Customer segmentation and collaborative filtering
- Text applications
- Social network analysis

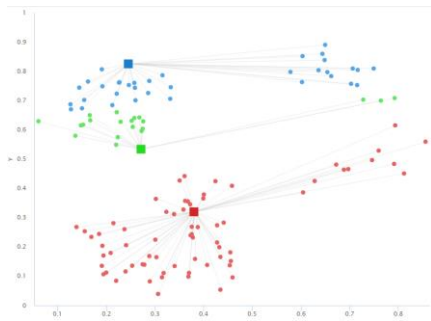
Clustering types

- Partitioning
 - k-means, k-medoids, k-modes
- Hierarchical
 - Agglomerative
- Grid-based
 - Multi-resolution grid structure
 - Efficient and scalable
- Density-based
 - A cluster is a dense region of points, which is separated by low density regions, from other regions of high density
 - Algorithms: DBSCAN, OPTICS, DenClue

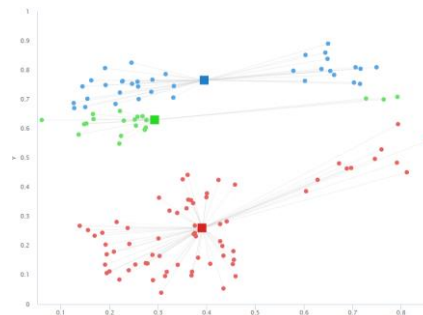


K-Means example

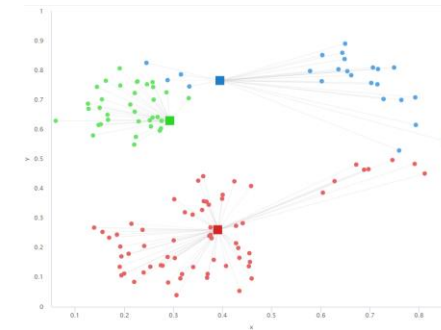
Random initialization



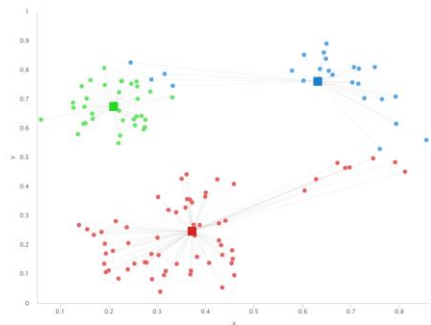
Centroid computation



Assignment of points to the nearest centroid



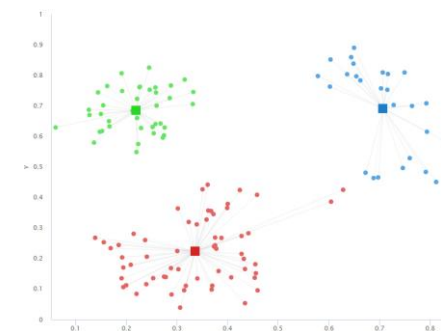
Centroid computation




Assignment of points to the nearest centroid



Centroid computation



K-means

1. Choose k random instances as cluster centers
 2. Assign each instance to its closest cluster center
 3. Recompute cluster centers by computing the average (aka *centroid*) of the instances pertaining to each cluster
 4. If cluster centers have moved, go back to Step 2
- 

(Equivalent termination criterion: stop when assignment of instances to cluster centers has not changed)

Alternatives: K-medoids, K-modes

- Might get stuck in local minima
- Silhouette for finding the optimal K

Clustering evaluation

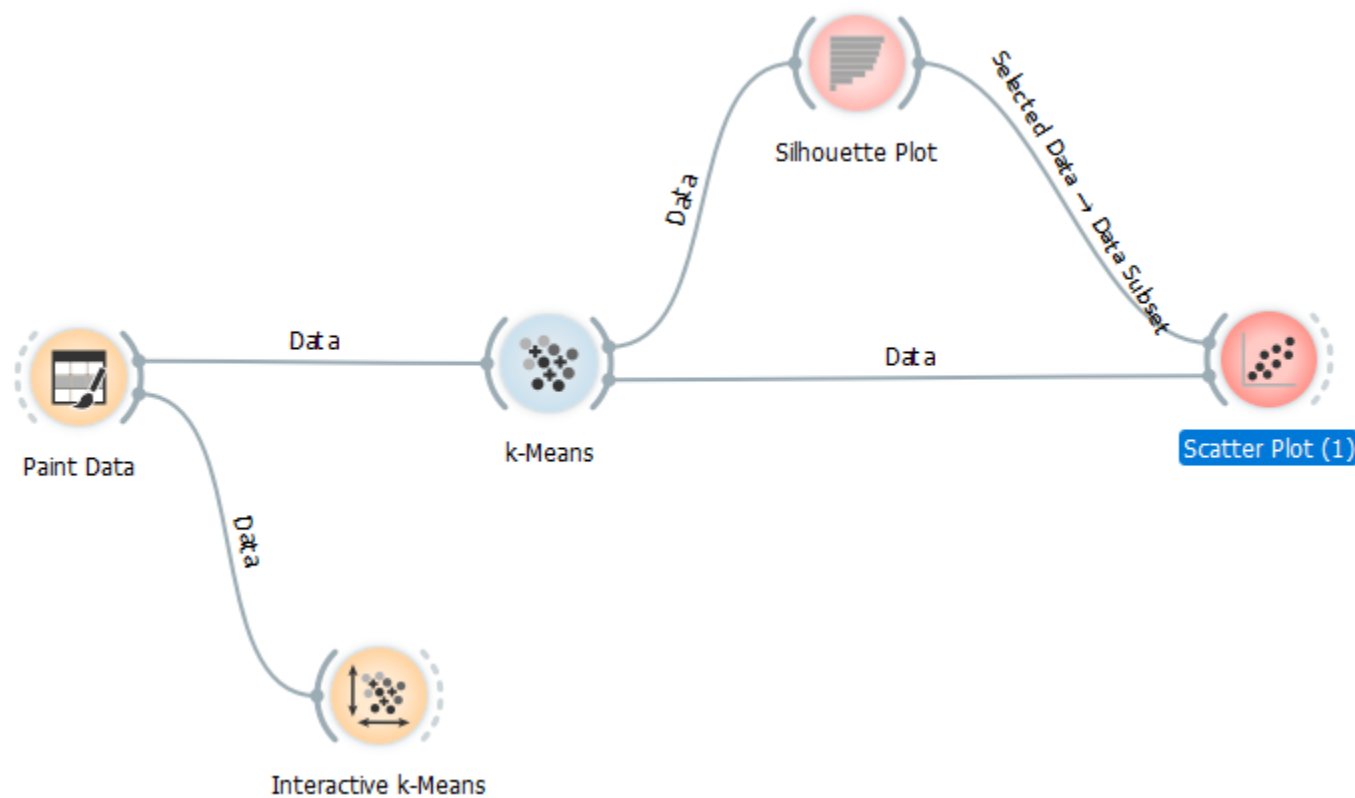
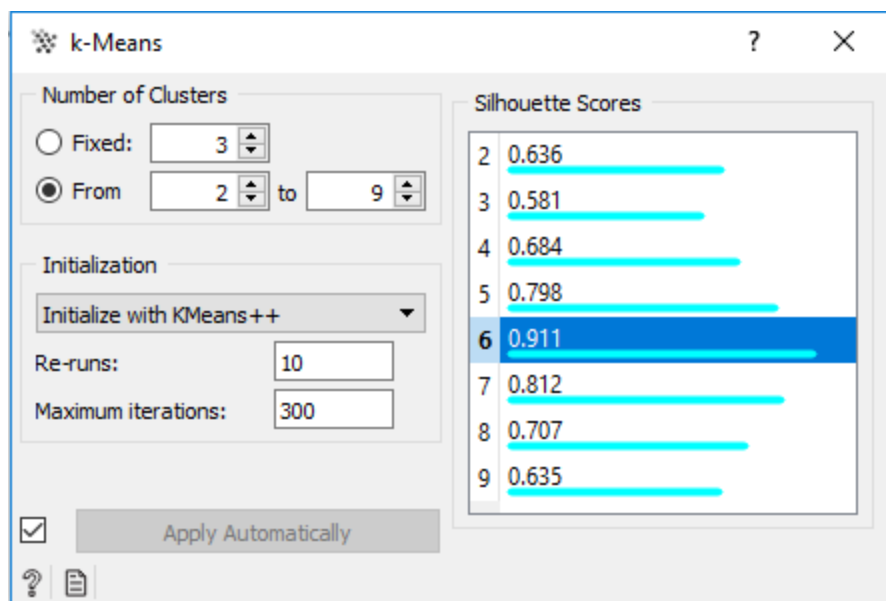
- Clustering analysis doesn't have a solid evaluation metric
 - External validation criteria
 - Using the ground truth to evaluate to evaluate the clustering result
 - Internal validation criteria
 - Sum of distances to centroids
 - Intracluster to intercluster distance ratio
 - Silhouette coefficient
-
- Parameter tuning – the “elbow” method

Silhouette coefficient

- The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation).
- For example x_i , its silhouette coefficient is
$$s_i = (b_i - a_i) / \max(a_i, b_i)$$
 - a_i average distance between x_i to all other examples in its cluster.
 - b_i average distance between x_i to the examples in the “closest neighboring” cluster
- The overall silhouette coefficient is the average of the data point-specific coefficients.

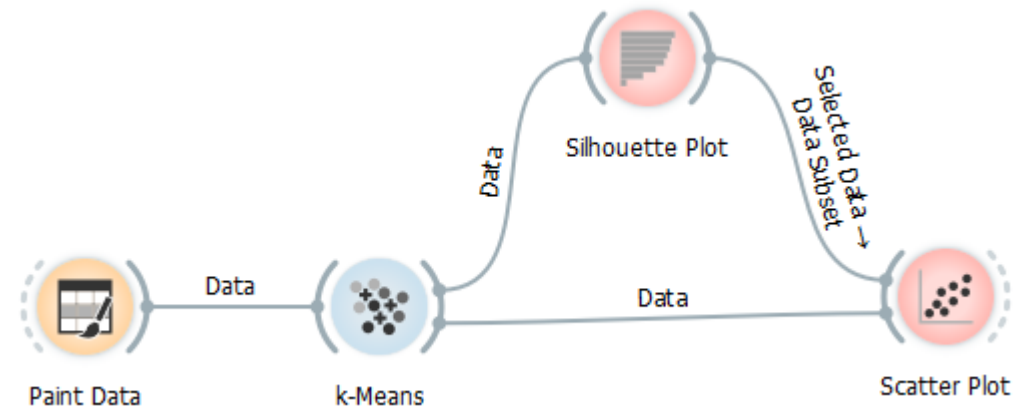


k-Means + Silhouette + „reruns“



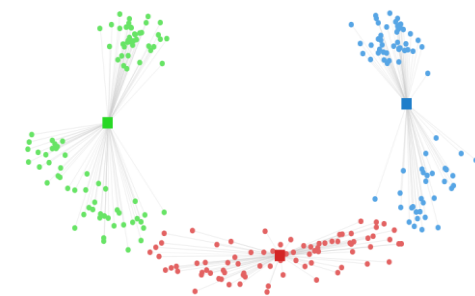
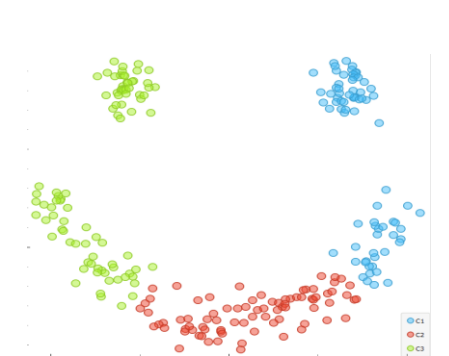
Lab exercise: clustering a smiley face

- Paint data: A smiley face
- Cluster by k-means clustering

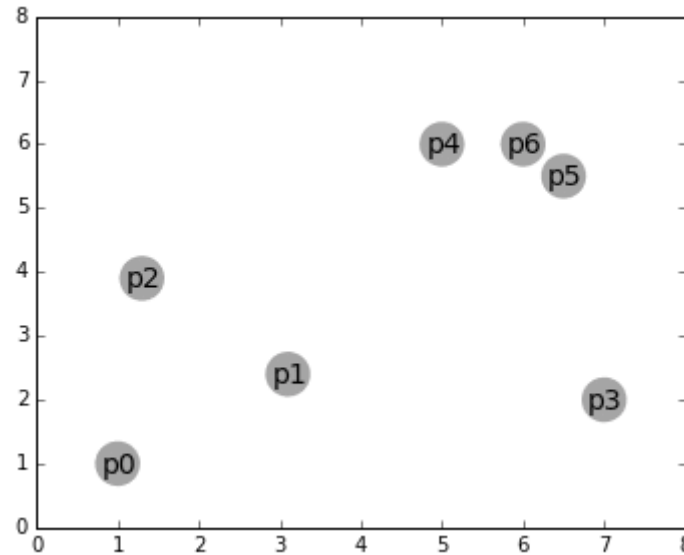


Properties of k-Means

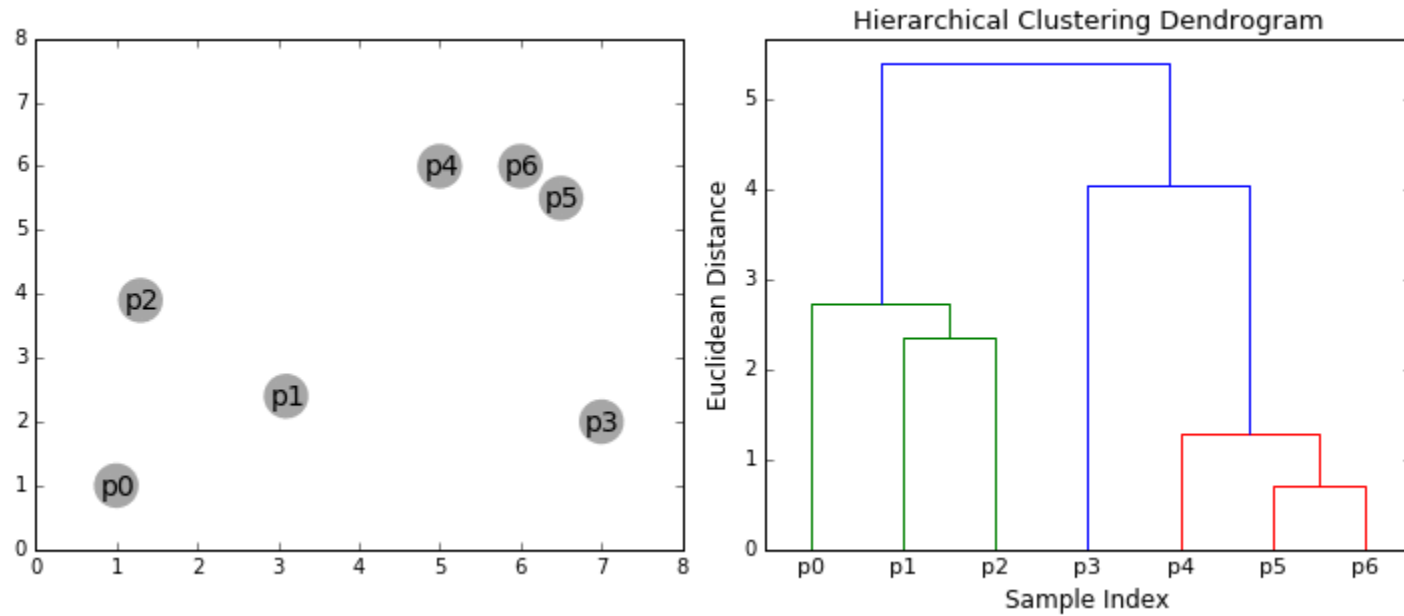
- The number of clusters k is fixed in advance
- It is fast, it always converges
- Can converge into a local minima (bad solution because of unlucky start)
- Finds “spherical” shaped clusters
- K-Means will cluster the data even if it can't be clustered (e.g. data that comes from uniform distributions)



Agglomerative clustering - example



Agglomerative clustering - dendrogram



Agglomerative clustering

1. Start with a collection \mathbf{C} of n singleton clusters
 - Each cluster contains one data point $\mathbf{c}_i = \{\mathbf{x}_i\}$
2. Repeat until only one cluster is left:
 1. Find a pair of clusters that is closest: $\min \mathbf{D}(\mathbf{c}_i, \mathbf{c}_j)$
 2. Merge the clusters \mathbf{c}_i and \mathbf{c}_j into \mathbf{c}_{i+j}
 3. Remove \mathbf{c}_i and \mathbf{c}_j from the collection \mathbf{C} , add \mathbf{c}_{i+j}

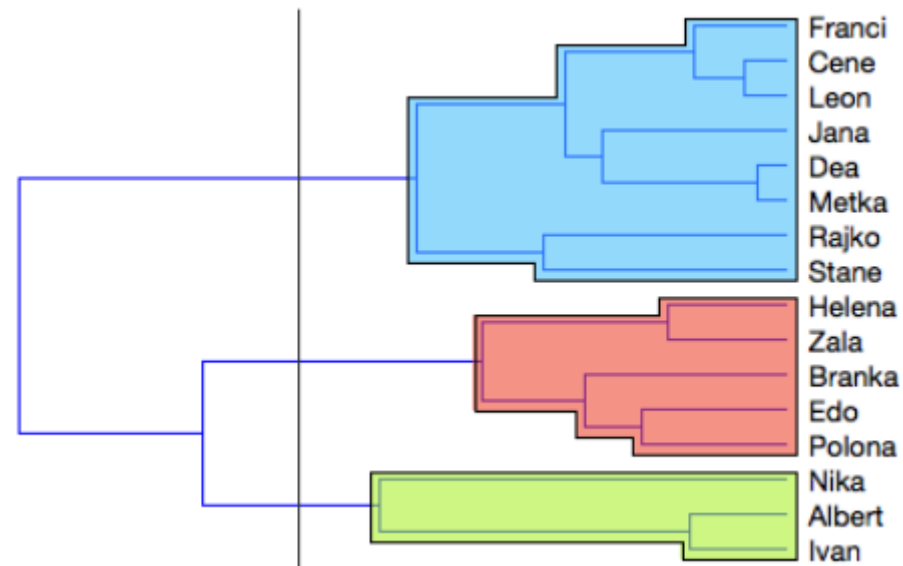


Some new index, not a sum

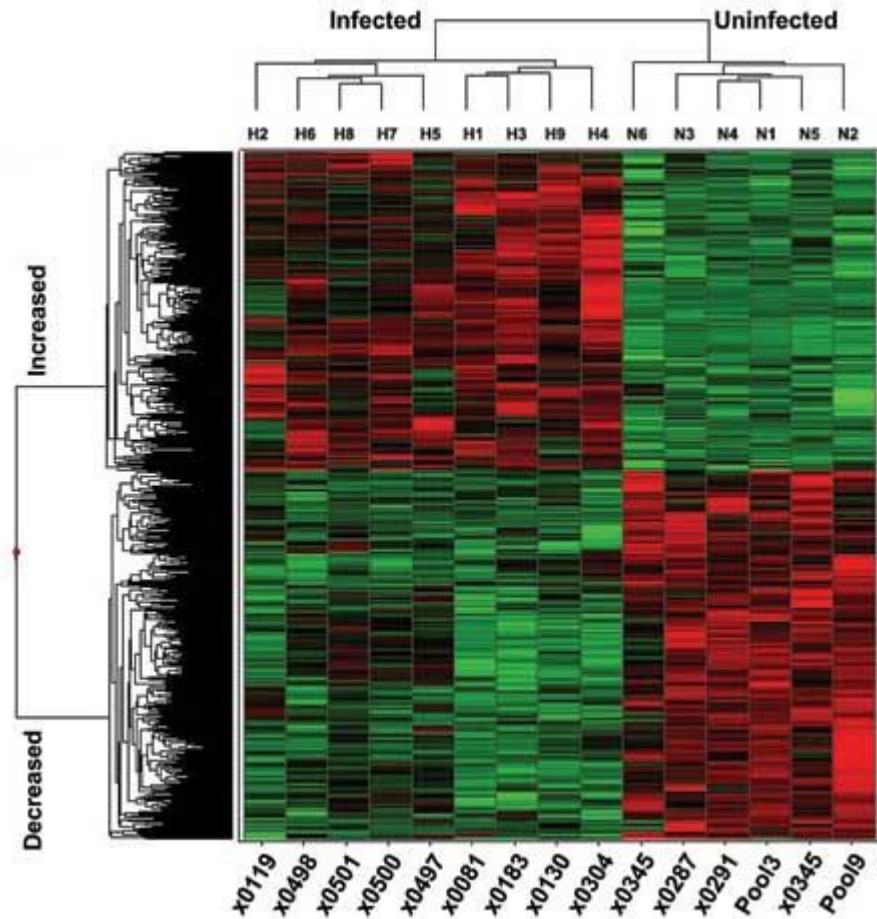
- Time and space complexity
- Sensitive to noisy data

Dendrogram

- The agglomerative hierarchical clustering algorithm's result is commonly displayed as a tree diagram called a dendrogram.
- Dendrogram is a tree diagram for showing taxonomic relationships.

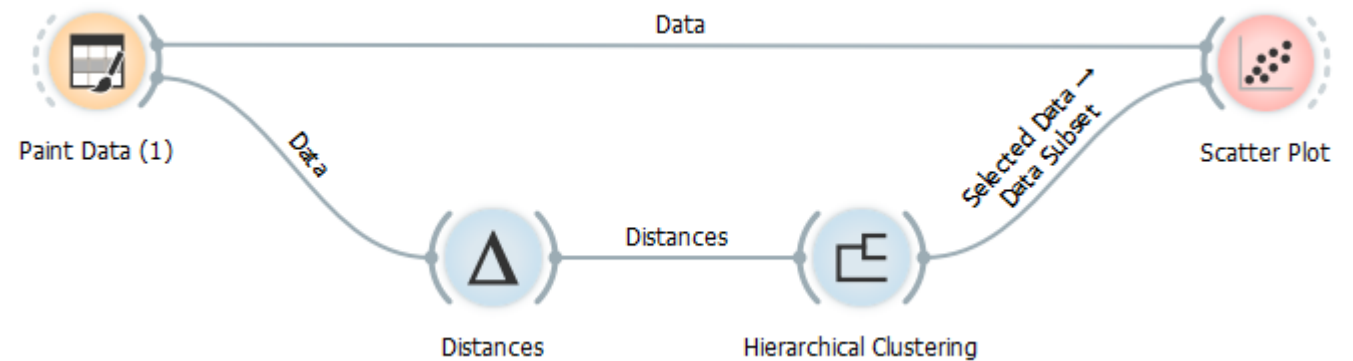


Example: Hierarchical clustering of genes



Lab exercise: clustering a smiley face

- Paint data: A smiley face
- Cluster by k-means clustering

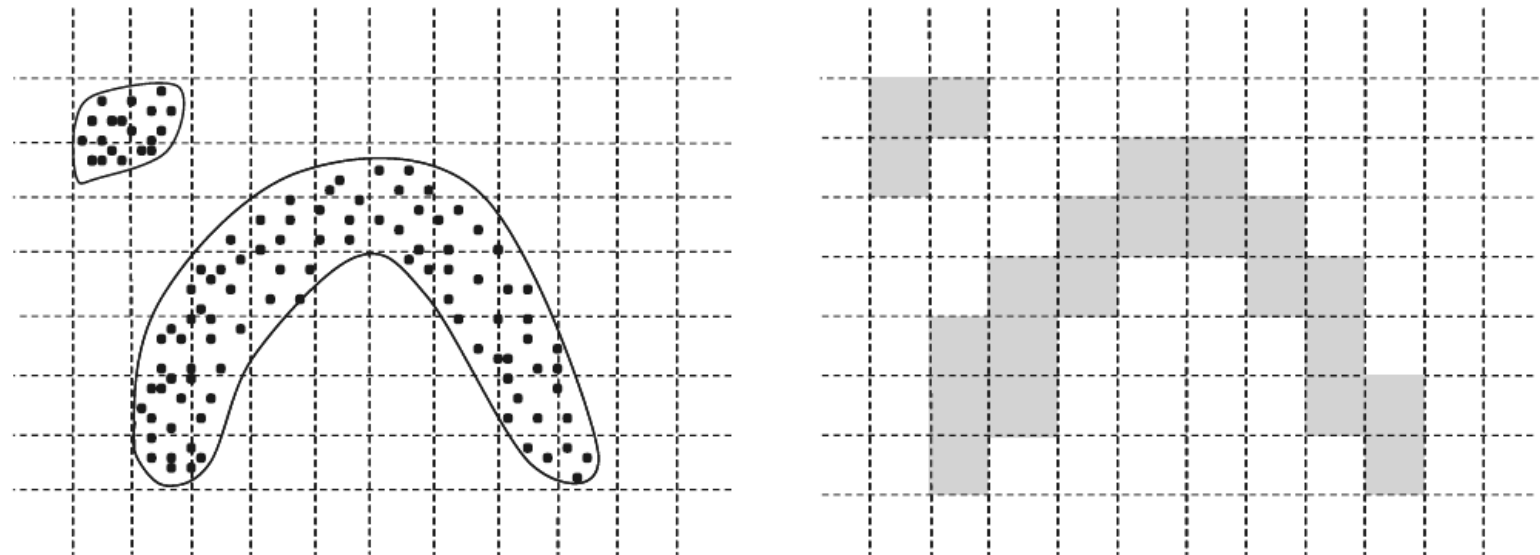


Grid-based (parameters \mathbf{p} and τ)

1. Discretize each dimension of \mathbf{D} into \mathbf{p} ranges
2. Determine dense grid cells at level τ
3. Create graph where dense grid cells are connected if they are adjacent
4. Determine connected components of graph
5. Return: points in each connected component as a cluster

Grid-based (parameters \mathbf{p} and τ)

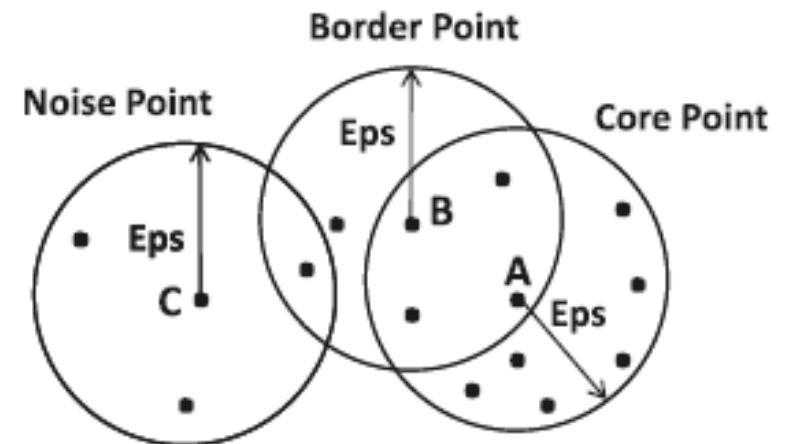
1. Discretize each dimension of \mathbf{D} into \mathbf{p} ranges
2. Determine dense grid cells at level τ
3. Create graph where dense grid cells are connected if they are adjacent
4. Determine connected components of graph
5. Return: points in each connected component as a cluster



Density based clustering

DBSCAN(Data: D , Radius: Eps , Density: τ)

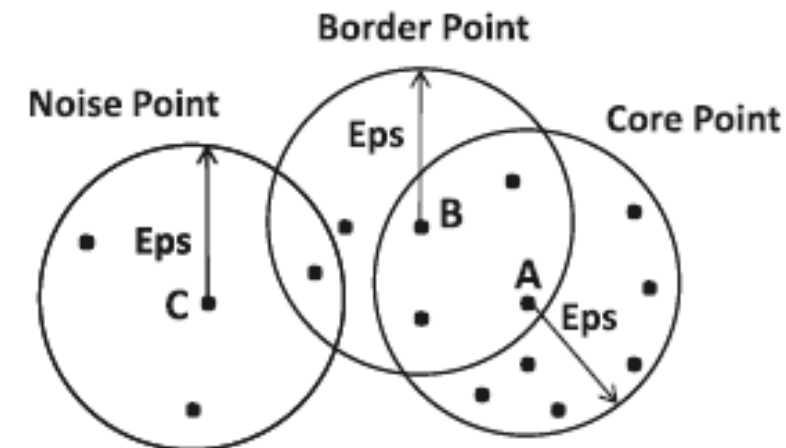
- *Core point*: A data point is defined as a *core* point, if it contains at least τ data points within a radius Eps .
- *Border point*: A data point is defined as a *border* point, if it contains less than τ points, but it also contains at least one core point within a radius Eps .
- *Noise point*: A data point that is neither a core point nor a border point is defined as a *noise* point.



Density based clustering

DBSCAN(Data: D , Radius: Eps , Density: τ)

1. Determine core, border and noise points of D at level (Eps, τ) ;
2. Create graph in which core points are connected if they are within Eps of one another;
3. Determine connected components in graph;
4. Assign each border point to connected component with which it is best connected;
5. **Return** points in each connected component as a cluster;



DBSCAN properties

Similar to grid-based approaches, except that it uses circular regions as building blocks.

Advantages of DBSCAN:

- Can detect clusters of arbitrary shape.
- Does not require the number of clusters as an input parameter.
- Not sensitive to outliers.

Disadvantages of DBSCAN:

- Computationally expensive in the first step (Determine core, border and noise points of D at level (Eps, τ));
- Susceptible to variations in the local cluster density.
- Struggles with high dimensionality data.

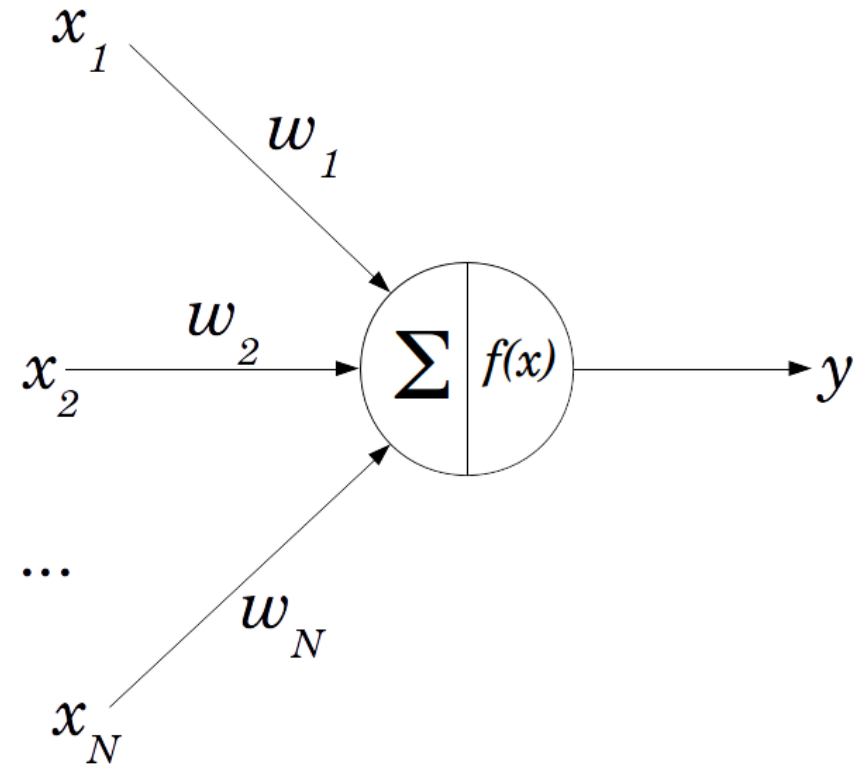
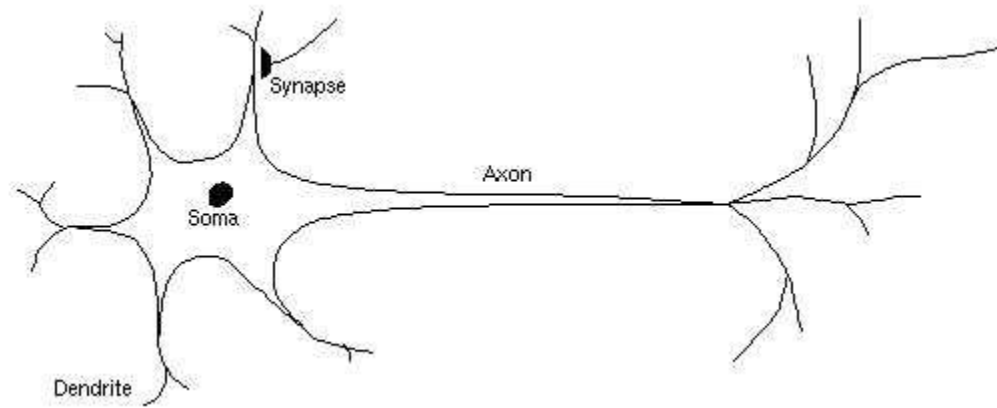
Literature

- Max Bramer: Principles of data mining (2007)
 - 14. Clustering
- Aggarwal, Charu C. *Data mining: the textbook*. Springer, 2015.
Chapter 6: cluster analysis, pgs 195 -201

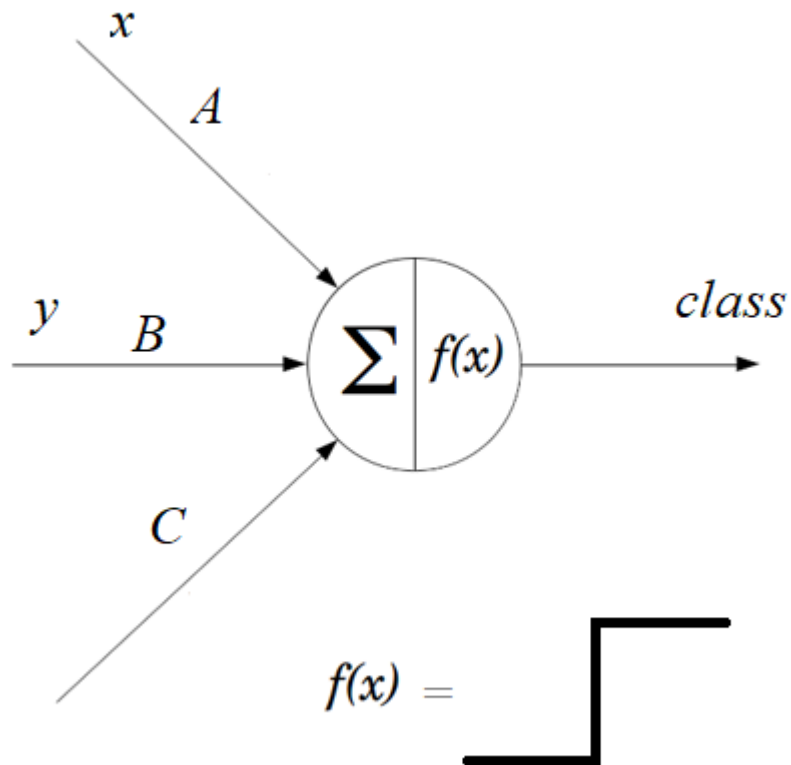
The background consists of a repeating pattern of circular portraits of Stefan Jovanović, a Serbian mathematician. Each portrait is set within a circular frame that also contains the mathematical formula $J = \sigma \cdot T^4$. The portraits and text are rendered in a light, semi-transparent style against a light blue background.

Neural networks

Neuron, perceptron



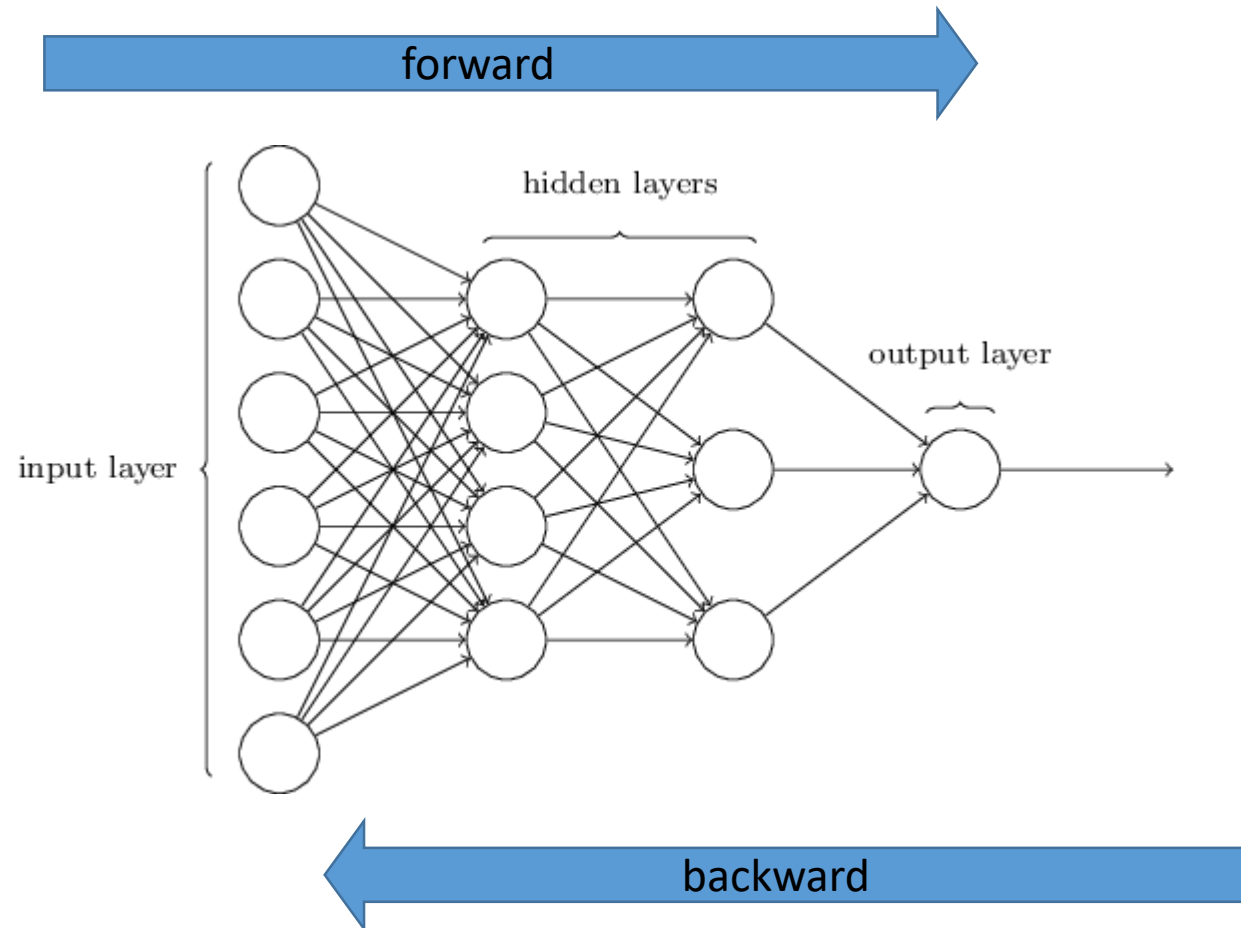
The **perceptron** is a mathematical model of a biological neuron



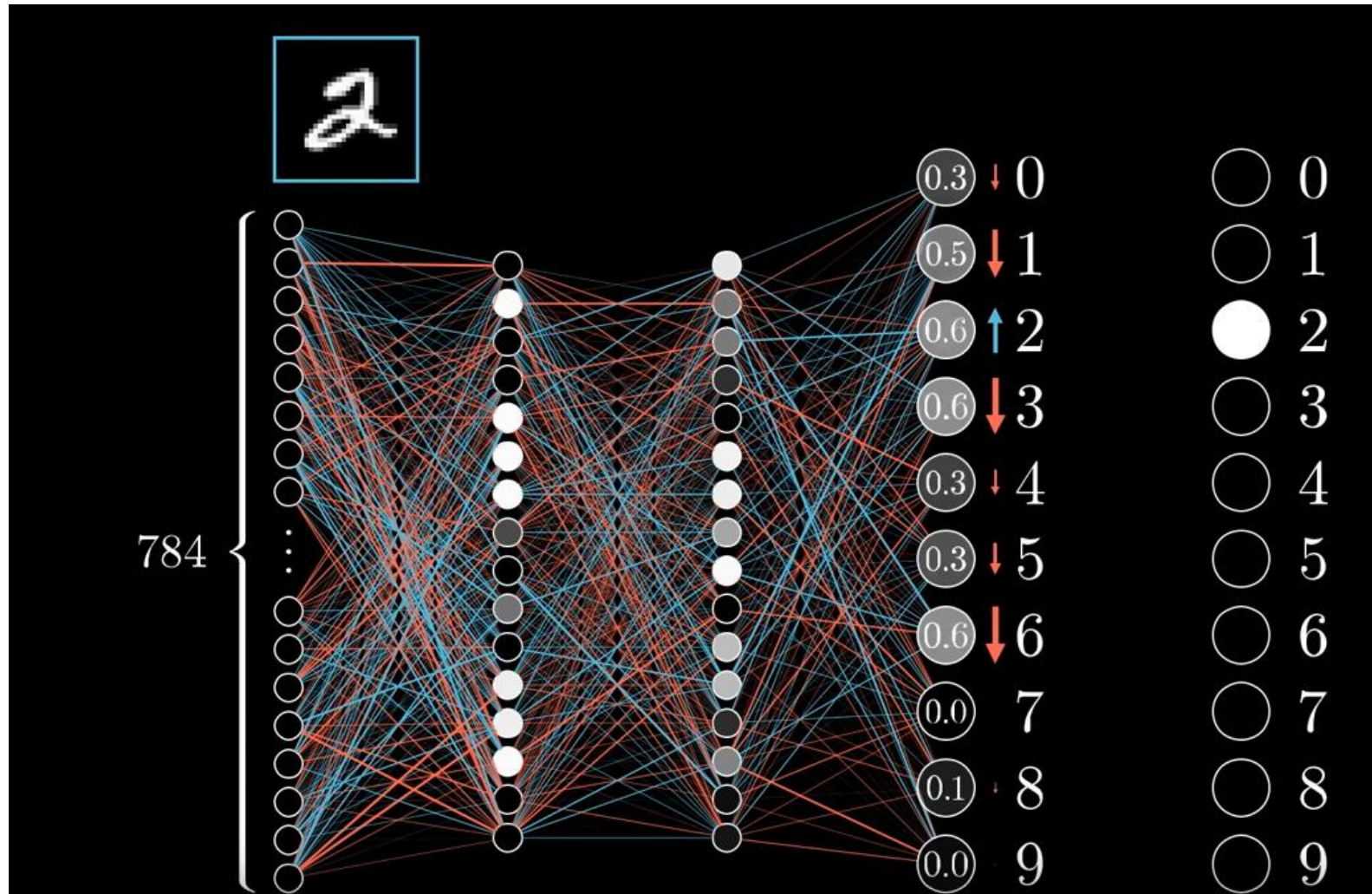
- A single perceptron can separate linearly.

$$\text{Output of P} = \begin{cases} 1 & \text{if } Ax + By > C \\ 0 & \text{if } Ax + By \leq C \end{cases}$$

Neural network



What is backpropagation really doing?



Predictive model

- Architecture
 - Define
 - Compile
- Train (fit)
 - Forward
 - Backward
 - Optimize
- Predict (evaluate)
 - Forward

Train

- **Forward propagation** (check performance)
 - **Loss function** is an error metric between actual and predicted
 - absolute error, sum of squared errors
- **Backpropagation** (direction of parameter/weight change)
 - How much the total error will change if we change the internal weight of the neural network by a certain small value **Δw (gradient)**
 - Backpropagate the errors using the derivatives of these functions: auto-differentiation
- **Optimization** (change weights based on learning rate, gradient descent)
 - New weight = old weight — Derivative Rate * learning rate
 - **Batch size** is a hyperparameter that controls the number of training samples to work through before the model's internal parameters are updated.
 - The number of **epochs** is a hyperparameter that controls the number of complete passes through the training dataset.



Handwritten Digit Recognition using Convolutional Neural Networks in Python with Keras

<https://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/>

Keras: The Python Deep Learning library

- Keras is a high-level neural networks API, written in Python and capable of running on top of [TensorFlow](#), [CNTK](#), or [Theano](#).
- Google's Tensorflow: is a low-level framework that can be used with Python and C++.
 - Install packages: tensorflow, keras

MINST – handwritten digits

- Each image is a 28 by 28 pixel square (784 pixels total).
- Normalized in size and centered
- A standard split of the dataset is used to evaluate and compare models, where 60,000 images are used to train a model and a separate set of 10,000 images are used to test it.

From the MINST Database of Hand-written Digits



Exercise

- Load the MNIST dataset in Keras.
- Train and evaluate a **baseline neural network** model for the MNIST problem.
- Train and evaluate a simple **Convolutional Neural Network** for MNIST.
- Implement a **close to state-of-the-art deep learning** model for MNIST.

Load the data: 9_neural_nets-0-load_data.py

```
from keras.datasets import mnist
import matplotlib.pyplot as plt

# Plot ad hoc mnist instances

(X_train, y_train), (X_test, y_test) = mnist.load_data() # Dataset of 60,000 28x28
grayscale images of the 10 digits, along with a test set of 10,000 images.
# plot 4 images as gray scale
plt.subplot(221)
plt.imshow(X_train[0], cmap=plt.get_cmap('gray'))
plt.subplot(222)
plt.imshow(X_train[1], cmap=plt.get_cmap('gray'))
plt.subplot(223)
plt.imshow(X_train[2], cmap=plt.get_cmap('gray'))
plt.subplot(224)
plt.imshow(X_train[3], cmap=plt.get_cmap('gray'))
# show the plot
plt.show()
```

Prepare data: 9_neural_nets-1-perceptron.py

```
# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)

# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# flatten 28*28 images to a 784 vector for each image
num_pixels = X_train.shape[1] * X_train.shape[2]
X_train = X_train.reshape(X_train.shape[0], num_pixels).astype('float32')
X_test = X_test.reshape(X_test.shape[0], num_pixels).astype('float32')

# train-validation split
X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train, test_size=0.1, random_state=42)

# normalize inputs from 0-255 to 0-1
X_train = X_train / 255
X_validation = X_validation / 255
X_test = X_test / 255

# one hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_validation = np_utils.to_categorical(y_validation)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]
```


One-hot Encoding for Multi-label and multi-target prediction

```
# one-hot encoding class labels
```

```
from keras.utils import np_utils
```

```
y_train[:10]
```

```
array([5, 0, 4, 1, 9, 2, 1, 3, 1, 4], dtype=uint8)
```

```
y_train_OneHotEncoding = np_utils.to_categorical(y_train)  
y_train_OneHotEncoding[:10]
```

```
array([[ 0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.],  
       [ 1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],  
       [ 0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.],  
       [ 0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],  
       [ 0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],  
       [ 0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],  
       [ 0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.],  
       [ 0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],  
       [ 0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.]])
```

0 1 2 3 4 5 6 7 8 9

Define + compile, fit, predict: 9_neural_nets-1-perceptron.py

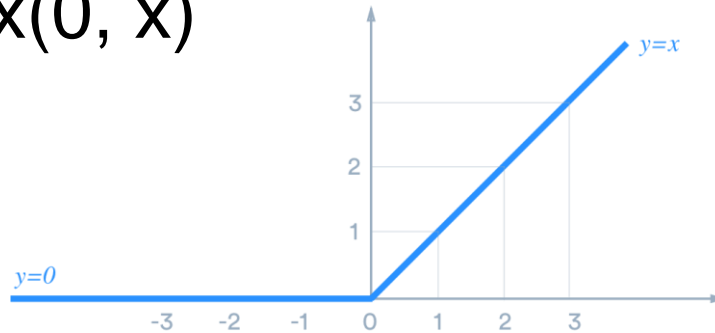
```
# define baseline model
def baseline_model():
    # create model
    model = Sequential()
    model.add(Dense(num_pixels, input_dim=num_pixels, kernel_initializer='normal', activation='relu'))
    model.add(Dense(num_classes, kernel_initializer='normal', activation='softmax'))
    # Compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

# build the model
model = baseline_model()
# Fit the model
model.fit(X_train, y_train, validation_data=(X_validation, y_validation), epochs=10, batch_size=200)

# Final evaluation of the model
print("Final evaluation of the model")
scores = model.evaluate(X_test, y_test, verbose=1)
print("Baseline Error: %.2f%%" % (100 - scores[1] * 100))
```

Activation functions

- $\text{relu}(x) = \max(0, x)$



- Softmax
 - After applying softmax, each component will be in the interval $[0,1]$, and the components will add up to 1
 - The softmax function is frequently used as the final activation function in neural networks for classification problems.
 - Maps the non-normalized output of a network to a probability distribution over predicted output classes.

Loss function: categorical_crossentropy

- Multi-class classification tasks
- Must be combined with Softmax

$$L(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_{j=0}^M \sum_{i=0}^N (y_{ij} * \log(\hat{y}_{ij}))$$

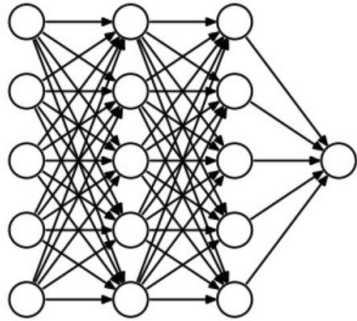
- \hat{y}_{ij} is the predicted value
- y_{ij} is the actual (correct) value

Architecture

- Layers: type, initialization, regularization
 - Dense
 - Convolutional
 - Pooling
 - Dropout – for regularization
 - Recurrent
 - Embedding
- Activation functions
 - relu
 - softmax (output layer)
- Loss function
 - Classification
 - **categorical_crossentropy**, categorical_hinge, sparse_categorical_crossentropy, binary_crossentropy, ...
 - Numeric prediction
 - **mean_squared_error**, mean_absolute_error, mean_absolute_percentage_error, mean_squared_logarithmic_error, cosine_proximity, ...
- **Model.compile**

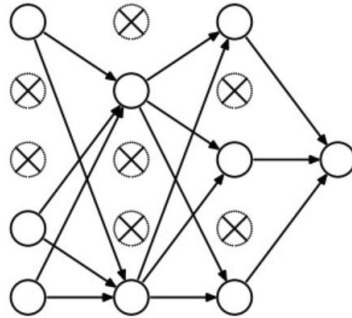
Types of layers (1)

Dense



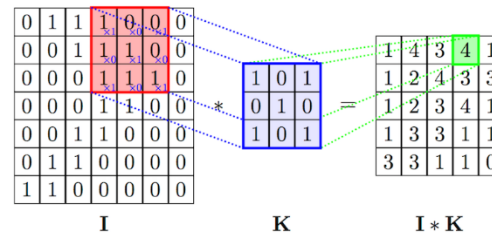
Fully connected.

Dropout



During training, some neurons on a particular layer will be deactivated. This improves generalization because it forces the layer to learn with different neurons the same "concept".

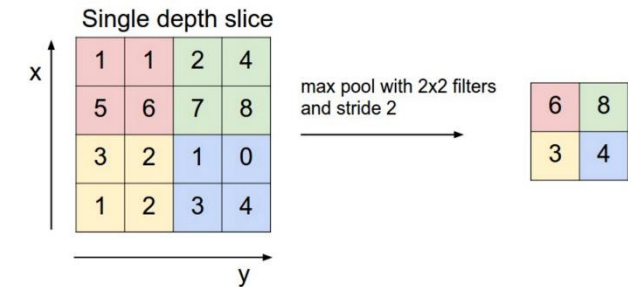
Convolutional



The convolution layer comprises of a set of independent filters. Each filter is independently convolved with the image.

Example: [link](#)

Pooling

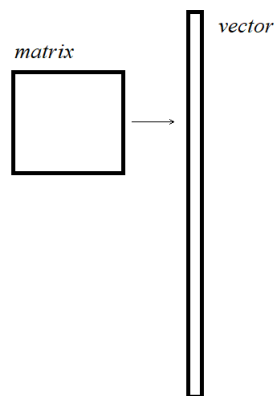


A max-pooling layer takes the maximum of features over small blocks of a previous layer.

Edge detection example
<https://youtu.be/puxHUGpuOVw>

Types of layers (2)

Flatten



Fully connected.

LAYERS

About Keras layers

Core Layers

Convolutional Layers

Pooling Layers

Locally-connected Layers

Recurrent Layers

Embedding Layers

Merge Layers

Advanced Activations Layers

Normalization Layers

Noise layers

Layer wrappers

Writing your own Keras layers

Convolutional model

```
def baseline_model():
    # create model
    model = Sequential()
    model.add(Conv2D(32, (5, 5), input_shape=(1, 28, 28), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))
    # Compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```