

Machine learning with R

Uwe Reichel,
ELTE NYTK
uwe.reichel@nytud.hu

October 8th 2025

What is machine learning?

- Learning the relations between properties of an object and its target value
- **example: vowel classification**
 $F1=800\text{Hz}$, $F2=1200\text{Hz}$, $\text{duration}=0.1\text{s} \longrightarrow \text{'/a:/'}$
- **object:** vowel segment
- **target:** its category

What is machine learning?

Definitions

- **Features = independent variables:** properties of an object, e.g. *F1, F2, duration*
- **Feature vector:** object representation [*800, 1200, 0.1*]
- **Target = dependent variable:** value, feature vector is to be mapped on; e.g. *phoneme '/a:/'*

Machine Learning = Learning to map objects represented as feature vectors to their target values

Which method to choose?

Are targets known?

- yes: **supervised** learning
- no: **unsupervised** learning

Of what variable type is the target?

- categorical: **classification** task
- continuous: **regression** task

... instead of End-to-end Deep Neural Networks

- which learn their own object representations (such as *bottleneck features*, *embeddings*, etc.)
- if data is sparse (e.g. clinical data)
- if one can/wants to rely on expert features
- if explainability matters

- **Supervised learning**
 - The CARET R package
 - Instance-based learning
 - Classification and regression trees
 - Ensemble models
 - Support vector machines
 - Bias and Variance
- **Unsupervised learning**
 - Clustering

Supervised learning – the Caret package

Basics

- **Classification and regression training**
- wrapper around lots of machine learning methods provided by other packages
- allowing for a common workflow
- CRAN: <https://cran.r-project.org/web/packages/caret/index.html>
- Function reference: <https://cran.r-project.org/web/packages/caret/caret.pdf>
- Tutorial: <https://topepo.github.io/caret/index.html>

Workflow

- 1 split data into train and test partition
- 2 feature normalization
- 3 model hyperparameter optimization and training
- 4 evaluation

Data split

- to allow for testing whether or not the model can generalize from the examples it was trained on
- if not: **overadaption** to training data
- **stratified split**: same relative class proportions in training and test partition
- Function: *createDataPartition()*

Data preprocessing

- **normalization** for mean and standard deviation
 - features usually have different mean and range values (e.g. fundamental frequency vs. formants)
 - thus contribute to different extent to classification, e.g. distance between objects more determined by features with high range
 - solution: z-transform $x \leftarrow \frac{x - \text{mean}(x)}{\text{std}(x)}$, so that all features have mean 0 and std 1
 - `train(... preProc = c("center", "scale") ...)`
- **decorrelation**
 - features might be highly correlated and thus are redundant in training
 - solution: orthogonalization by Principal component analysis
 - `train(... preProc = c("pca") ...)`

Training

- **Tuning: hyperparameter optimization**
 - **grid-search:** loop over all hyperparameter value combinations
 - for each value combination, do an **n-fold cross-validation**, i.e.
 - divide training set n times into fitting and development partition
 - fit the model to the data in the fitting partition and evaluate it on the development partition.
 - Keep the hyperparameter combination with highest mean evaluation score
- **Fitting:** fit model with optimized hyperparameters on training partition
- specify training procedure: $tc \leftarrow trainControl()$
- training: $train(\dots trControl = tc \dots)$

Evaluation

- Apply the model to unseen test set: *predict()*
- measure model performance by comparing its output with the reference targets from test set
 - **Classification:** *confusionMatrix()*
 - **Regression:** *RMSE()*

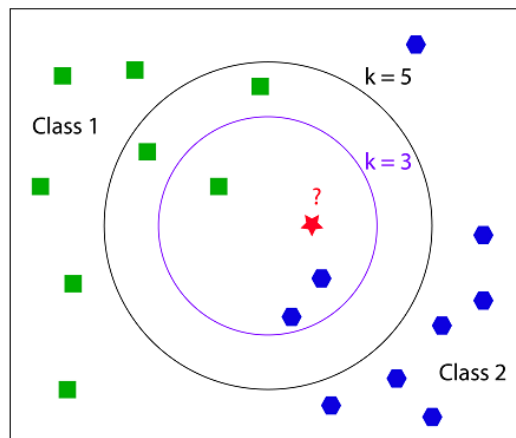
Metrics

- **Classification**
 - classes balanced: **Accuracy**, i.e. percentage correct, **Mean F-Score**
 - classes not balanced: **Unweighted Average Recall or F-Score**
- **Regression**
 - distance matters: **Mean Absolute or Squared Error**
 - correlation matters, but not distance: **Pearson Correlation Coefficient**
 - both matters: **Concordance Correlation Coefficient**

Overview

- **supervised** learning method
- **task:** classification
- **features:** categorical, continuous
- **targets:** categorical
- **scenarios:** exemplar theory modeling
- **training:** store feature vectors with their corresponding class (*lazy learning*)
- **application:** for an object to be classified select the k nearest feature vectors. Assign to the object the **class with the highest support (weight sum)** among the retrieved feature vectors.
- **CARET method:** *kknn()*

IBL example for different k



Ref.: <https://machine-learning-tutorial-abi.readthedocs.io/>

Classification: $\hat{f}(y) = \arg \max_{c \in C} \sum_{i=1}^k w_i \delta(c, f(x_i))$

- y : object to be classified
- C : set of possible classes
- $\sum_{i=1}^k w_i \delta(c, f(x_i))$: support for class c
- k : number of nearest neighbors to be considered
- $f(x_i)$: class of neighbor x_i
- $\delta(a, b)$: 1 if $a=b$, else 0
- w_i : weight of neighbor x_i derived from its distance d to y by a Kernel function, e.g. the **inversion kernel** $\frac{1}{|d|}$
- **Parameters:**
 - $kmax$ – maximum number k of neighbors
 - $distance$ – parameter of **Minkowski distance** (1=Manhattan, 2=Euclidean)
 - $kernel$ – to map distance to weight; 'rectangular' – unweighted, 'optimal' – design depends on $kmax$ (i.e. for high k weight for high distance gets reduced more strongly), others – differ wrt extent the weight decreases with increasing distance

Classification and regression trees (CARTs)

Overview

- **supervised** learning method
- **tasks:** classification and regression
- **features:** categorical, continuous
- **targets:** categorical, continuous
- **training:** represent feature vectors as paths through tree, and targets as tree leaf labels
- **application:** follow path according to the object's feature vector. Assign to the object the terminal leaf label.
- **CARET method:** *rpart2()*
- **Parameter:** maximum tree depth *maxdepth*
 - deep trees can take more features and their interactions into account
 - flat trees are less prone to overadaption, and thus often generalize better

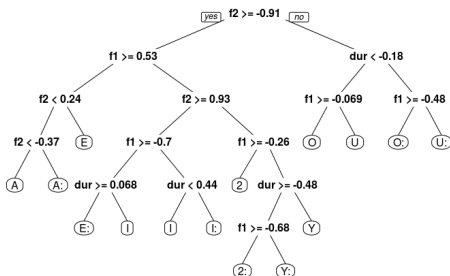
Training procedure

- divide and conquer: recursive partitioning of an object set into subsets
- resulting tree with one leaf for each final subset
- **Regression tree:** for each subset a simple regression model is fitted (which simply corresponds to the subset mean value)
- **Classification Tree:** to each subset a class label is assigned (which is the most frequent class in this subset)

Classification and regression trees

CART examples

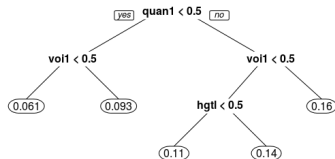
Classification



Target: Vowel class

Features: 1st and 2nd formant, duration

Regression



Target: Vowel duration

Features: Vowel height and position,
voicing of neighbor-Cs

Split criteria

- according to which variable the objects are to be divided?
- **regression tree:** search over variables (and split points) to minimize the variation within a subset
 - **example:** target T = phone duration, features X_1 = prominence, X_2 = sentence mode
 - by X_1 the phones can be divided into 2 subsets one for long, the other for short vowels
 - by X_2 the division yields overlapping subsets wrt to phone duration
 - $\rightarrow$ use X_1 to subdivide the objects

- **classification tree:** use variable X containing the **highest information** about the target class T

$$\hat{X} = \arg \max_V [MI(X; T)]$$
$$MI(X; T) = H(T) - H(T|X)$$

- $H(T)$: entropy (incertitude) of target value
- $H(T|X)$: remaining incertitude if value of X is known
 - **example** T = vowel class, X = F2
 - $MI(X; T)$: mutual information between the F2 and the vowel class
 - $H(T)$: the incertitude of predicting the vowel, if no cues are given
 - $H(T|X)$: the remaining incertitude to predict the vowel class if the F2-value is known

Termination criteria

- all objects in a subset have (about) the **same target value**
 - *classification*: all vowel objects in a subset are /i/s
 - *regression*: all phones in a subset have about the same duration
- the objects are **not further dividable** by their feature vectors
 - *all phones within a subset are in prominent position of a declarative sentence*
- the **number** of objects in a subset is **below** a specified **threshold**

Leaf labels

- *classification*: class occurring most often in the subset at that leaf
- *regression*: mean value of all targets at that leaf

Application

- follow the tree from the node to a leaf according to the object's feature values
- assign the leaf label to the object

Overview

- **supervised** learning method
- **tasks:** classification and regression
- **features:** categorical, continuous
- **targets:** categorical, continuous
- **CARET method:** e.g. `xgbTree()`
- **Parameters:** number of trees (*nrounds*) maximum tree depth (*max_depth*), minimum number of items at a leaf (*min_child_weight*), learning rate (weight of each subsequent tree in correcting the preceding tree's error; *eta*), proportion of features a single tree is trained on (*colsample_bytree*), proportion of training items a single tree is trained on (*subsample*), minimum required loss reduction to further split a tree (*gamma*)

Ensemble models

- e.g. Random Forests, Gradient Boosting, Extreme Gradient Boosting
- **Boosting:** combining several weak classifiers to built a strong one (cf *wisdom of crowds*).
- **Bagging:** combining classifiers trained on parts of the training data
- consists of many **classification (or regression) trees**
- **training:**
 - see CARTs, each tree trained on feature and/or data subset
 - **parallel training:** trees are trained independently in parallel (e.g. Random Forest)
 - **sequential training:** trees are trained sequentially (Gradient Boosting); the later trees are trained to predict the residual, which is the deviation of the previous tree's predictions from the target value; i.e. later trees correct the errors of the previous trees
 - for classification these residuals refer to class probabilities

■ application:

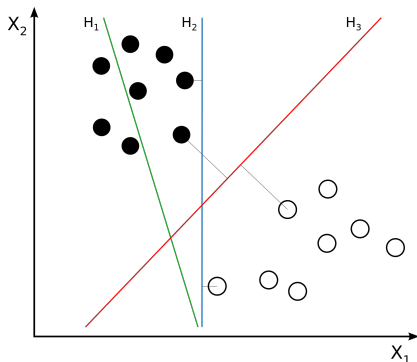
- the feature vector is accordingly subdivided in the forest
- each tree responds a class based on the feature values it sees
- **classification result: most frequent class**, if trees are trained in parallel (e.g. Random Forests); **class with the highest summed probability**, if trees are trained sequentially (Gradient Boosting)
- **regression result:** prediction **mean** (parallel training), or **sum** (sequential training)
- Fernández-Delgado et al. (2014): **Random forest as best performing classifier (next to SVMs)** for various data sets
- **XGBoost and SVMs are currently the overall best performing classical (i.e. non deep-learning) machine learning methods**

Support vector machines (SVM)

Overview

- for **supervised learning**
- **task:** binary classification, (regression)
- **features:** binary, continuous
- **targets:** categorical, (continuous)
- **Caret method:** *svmLinear()*
- **Parameters:** misclassification cost C
 - low values: **soft margin**, allowing for more errors
 - high values: **hard margin**, potentially overadapt to training data

Support vector machines (SVM)



Training:

- find the best plane (**H3**) in the feature space to separate two classes
- maximize the distance between the plane and the **support vectors**, i.e. the vectors closest to the plane

Training cntd.

■ Kernel:

- similarity function $K(x, y) = f(x) \cdot f(y)$
- inner product of feature vectors x and y that are mapped to a higher dimensional space by $f(\cdot)$

■ **Kernel trick:** if classes are not separable, then map feature vectors to a higher dimensional feature space and try again (increase of separability – but also overadaption!)

■ linear, polynomial, and RBF Kernel functions: different distance calculation in the feature space

■ robust against outliers, since these are ignored

■ **from two to n classes:**

- one SVM for each class pair c_i vs c_j , or
- one SVM for each class c_i vs $\neg c_i$

Bias and Variance

High bias, low variance

- **Simple model:** low number of neighbors (KNN), flat/few trees (CART, XGBoost), low misclassification cost (SVM), low learning rate (XGBoost)
- **Disadvantage:** does not make use of all information available in training data, e.g. feature interactions
- **Advantage:** robustness; does not over-adapt to training data and works equally well unseen data

Low bias, high variance

- **Complex model:** high number of neighbors (KNN), deep/many trees (CART, XGBoost), high misclassification cost (SVM), high learning rate (XGBoost)
- **Advantage:** powerful; can make use of more information in training data
- **Disadvantage:** might overfit to training data and thereafter not work well on unseen data

Hyperparameter tuning by cross validation to balance power (high variance) and robustness (high bias)

- What is machine learning?
- Which method to choose?
- Supervised learning
- **Unsupervised learning**
 - **Clustering**

Overview

- **unsupervised** learning method
- **task:** partition of the data into similar objects
- **features:** categorical, continuous
- **targets:** yet unknown
- **scenarios:** intonation contour classification
- **R packages:** *stats*

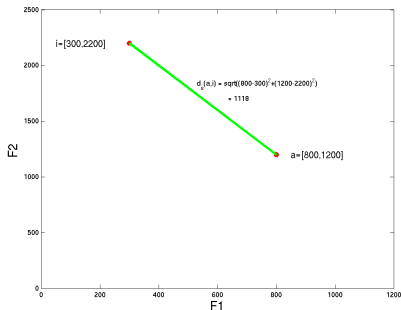
When to use?

- if no categories are available yet

Objects

- feature vectors
- points in a Cartesian coordinate system

Distance (between feature vectors a and b)



- **continuous variables:** e.g. *Euclidean distance*

$$d_e(a, b) = \sqrt{\sum_i (a_i - b_i)^2}$$

- **categorical variables:** e.g. *Hamming distance*

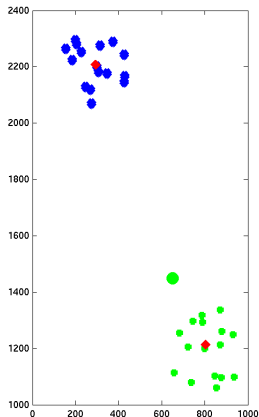
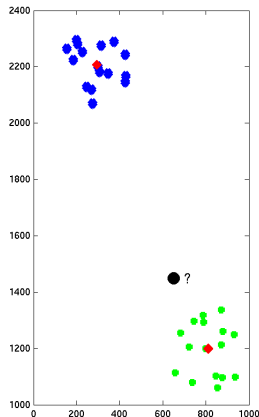
$$d_h = \frac{\sum_{i:a_i \neq b_i} 1}{\sum_i 1}$$

- **example: distinctive feature vectors for /a/ and /i/**
/a/ [low, back, spread, lax]
/i/ [high, front, spread, tense]
 $d_h(a, i) = \frac{3}{4} = 0.75$

kmeans Algorithm

```
 $X \leftarrow$  objects to be clustered  
 $k \leftarrow$  intended number of clusters  
init: determine  $k$  clusterCenters  
until all clusters stable  
  foreach  $x \in X$   
     $c \leftarrow$  closest cluster  
     $c \leftarrow [c, x]$   
    update clusterCenter( $c$ )  
  endforeach  
enduntil
```

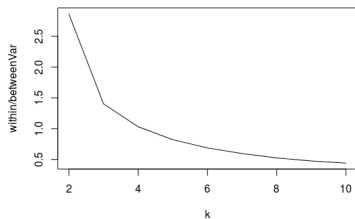
Clustering



- kmeans **cluster center: centroid** (mean vector)
- **closest** cluster: the cluster with the nearest centroid

How many clusters?

- i.e. determine k
- **Maximize Cohesion:** low within-cluster variability
- **Maximize Separation:** high between-cluster variability
- **Maximize Generalisation:** keep number of clusters as low as possible

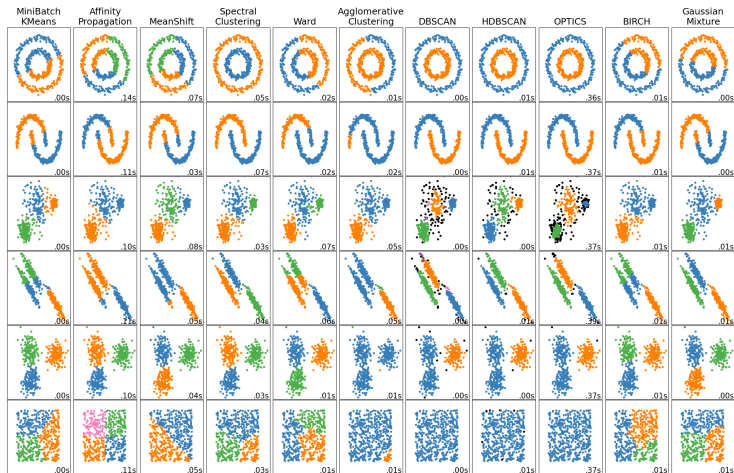


- best k at some **breakpoint** in $\frac{\text{withinClusterVariability}(k)}{\text{betweenClusterVariability}(k)}$ curve
- here: e.g. 3 or 6

Selected other clustering methods

- **KMedoid** as KMeans but using *medoids* (based on medians) instead of *centroids* (based on arithmetic means). More robust against outliers.
- **DBSCAN** based on neighborhood densities. Number of clusters does not need to be set but emerges from the data. More flexible cluster shapes, e.g. inner vs outer circle
- **Hierarchical clustering:** Creates a dendrogram which can be cut at any level resulting in more or less clusters.

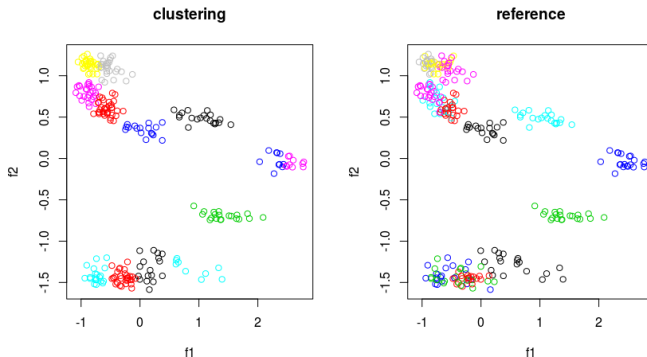
Selected other clustering methods



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

Comparing clustering output and reference

- Hungarian vowel classes by formants 1 and 2 (centered and scaled)

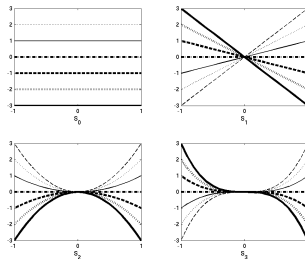
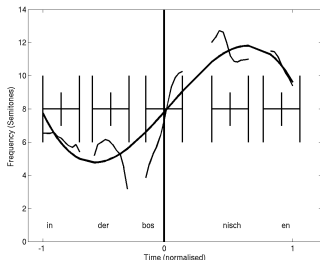


Validation of clustering results

- **without reference:** cluster separability in terms of e.g. *Mean Silhouette score*, *Dunn index*
- **with reference:**
 - *Mutual information:* How much information about the reference classification is contained in the clustering output
 - $I(X, Y) = H(X) - H(X|Y)$, where X stands for the underlying reference classes and Y stands for the cluster IDs. $H(X)$ is the entropy of X (the mean number of bits needed to encode values of X). $H(X|Y)$ is the conditional entropy of X given that values of Y are known.
 - *Adjusted Mutual information:* corrects for chance-level agreement and limits the I range to $(0, 1)$, higher is better.

Intonation contour classes

- derived from 3rd order polynomial stylization of f0 contours
- $f0 = \sum_{i=0}^3 s_i t^i$ (t : time, s_i : coefficients to be fitted)



Resulting contour classes (k=6)

