International Master in Computer Vision

Fundamentals of machine learning for computer vision

Eva Cernadas

Contents

- **Machine learning theory (Dr. Jaime Cardoso)**
- **Linear regression and optimization (Dr. Jaime Cardoso)**
- **Clustering**
- **Model selection and evaluation**
- **Classical classification models**
- **Artificial neural networks**
- **Support vector machines (SVM)**
- **Ensembles: bagging, boosting and random forest**

Machine learning (I)

- **Machine learning**: synthesize a function from a set of selected values (training examples).
- **Simple case**: linear regression (it concerns two-dimensional sample points with one independent variable and one dependent variable, $(x_1,y_1),\ldots,(x_N,y_N)$, and finds a linear function that, as accurately as possible (minimizing the mean squared error), predicts the dependent variable values as a function of the independent variable.

Machine learning (II)

- The coefficients of the line or polynomial can be calculated minimizing the mean squared error. They can be used to predict a function for $x \neq x_i$.
- The polynomial coefficients are the learned or trained parameters. For higher polynomial degrees, more trained parameters.
- Other possibilities would be to do interpolation (lineal, spline, etc.) instead of adjustment.

Machine learning (III)

- So, we always need to learn a set of parameters.
- In some cases, the learning of parameters leads to analytical expressions (mathematical formulas like in linear regression).
- If the problem complexity or number of trained parameters increase, more sophisticated and efficient methods may be needed instead of analytical methods.

Classification or regression

- How are **the values of the function** to predict or approximate?
  - **Continuous**: then **regression**
  - **Discrete** (categorical): (integer values with or without order) then **classification**

Supervised and unsupervised learning

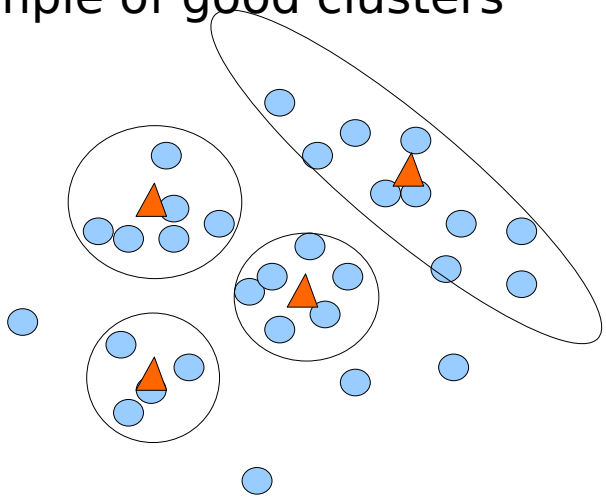
- **Supervised learning**: the value of the function to be predicted is already unknown, like in the linear regression, interpolation or classification.
- **Unsupervised learning**: the objective is to extract information from the data using criteria like similarity or distances, as in the case of **clustering** or **dimensionality reduction**.

Clustering

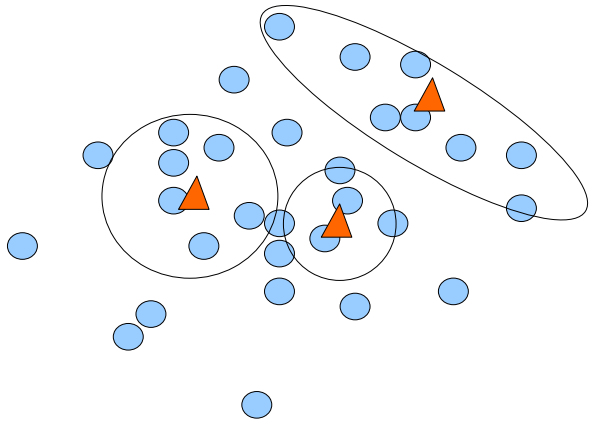
- Objective: to group a set of patterns into clusters (groups of patterns) using a similarity or distance measurement.
- To learn the data structure (the pattern groups).
- Similar patterns (low distances among them) are assigned to the same cluster and used to create a cluster prototype.
- Different patterns (high distances) are assigned to different clusters.
- The threshold to split the distances between similar and different patterns depends of the number of clusters.
- Many clusters: each cluster will have similar patterns.
- Few clusters: each cluster can have very different patterns.

Clustering

Example of good clusters



Example of difficult clustering



- The pattern distribution determines the clustering difficulty.
- The quality of the learned clustering: to what extent do cluster prototypes represent patterns?
- Are there pattern-free (empty) regions between clusters?
- In high dimensions, distances among patterns are very similar.

Clustering

- The results of the clustering are the cluster prototypes. The whole prototype set constitutes a compressed data representation.
- The prototypes should be placed on the regions in the input space with higher pattern density.
- Non-supervised learning: there is no pattern label (discrete nor continuous).
- There is only a distance measurement, which indicates how far two patterns are from each other or from a prototype.

Clustering: k-means

- The number K of clusters that you want to create must be known.
- Set randomly K prototypes $\{\mathbf{p}_k\}_{k=1}^{K}$ in K patterns.
- For each pattern $\{\mathbf{x}_i\}_{i=1}^{N}$:
  - Calculate the distance between $\mathbf{x}_i$ and $\{\mathbf{p}_k\}_{k=1}^{K}$
  - Select the closest prototype.
- For each prototype $\{\mathbf{p}_k\}_{k=1}^{K}$, update the prototype using only the closest patterns and return the calculation of the prototype.
- Repeat the process until the prototypes do not change significantly or a specified number of times.

K-means algorithm

Data: $K>1$, $\{\mathbf{x}_i\}_{i=1}^{N}$, $\varepsilon>0$

Output: $\{\mathbf{p}_j, E_j\}_{j=1}^{K}$: prototypes and pattern list for each cluster

$\{\mathbf{p}_k=\mathbf{x}_{s(k)}\}_{k=1}^{K}$ #$s(k)$: random number in $\{1..N\}$

repeat

$\mathbf{P}=\{\mathbf{p}_k\}_{k=1}^{K}$; $\{N_k\}_{k=1}^{K}=0$; $\{E_k=\emptyset\}_{k=1}^{K}$

for $i=1:N$

$$m=\underset{k=1..K}{argmin}\left\{\left|\boldsymbol{x}_i-\boldsymbol{p}_k\right|\right\}$$

$N_k=N_k+1$; $E_k=E_k\cup\{i\}$

endfor

for $k=1:K$

$$\boldsymbol{p}_k'=\frac{1}{N_k}\sum_{i\in E_k}\boldsymbol{x}_i$$

endfor

$\mathbf{P}'=\{\mathbf{p}_k'\}_{k=1}^{K}$; $\{\mathbf{p}_k=\mathbf{p}_k'\}_{k=1}^{K}$

until $|\mathbf{P}'-\mathbf{P}|<\varepsilon$

Distance measurements between patterns and prototypes

- Euclidean distance: $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\sum_{k=1}^{N} (x_{ik} - x_{jk})^2}$

- Mahalanobis distance: $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \Sigma^{-1} (\boldsymbol{x}_i - \boldsymbol{x}_j)}$

  $\boldsymbol{\Sigma}$ is the data covariance matrix

- Similarity: $s(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\boldsymbol{x}_i^T \boldsymbol{x}_j}{|\boldsymbol{x}_i||\boldsymbol{x}_j|}$

- Tanimoto similarity: $s(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\boldsymbol{x}_i^T \boldsymbol{x}_j}{|\boldsymbol{x}_i^2| + |\boldsymbol{x}_j^2| - \boldsymbol{x}_i^T \boldsymbol{x}_j}$

Measures of error and clustering quality (I)

- To minimize the quadratic error between patterns and prototypes:

$$J=\sum_{k=1}^{K}\sum_{x\in E_k}|\boldsymbol{x}-\boldsymbol{p}_k|^2$$

- To maximize the trace of the between-cluster covariance matrix $\mathbf{S}_b$:

$$tr(\boldsymbol{S}_B)=\sum_{k=1}^{K}N_k|\boldsymbol{p}_k-\boldsymbol{p}|^2 \qquad \boldsymbol{p}=\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i$$

- To minimize the determinant of within-cluster cov. matrix $\mathbf{S}_W$:

$$det(\boldsymbol{S}_W)=det\left(\sum_{k=1}^{K}\boldsymbol{S}_k\right) \qquad \boldsymbol{S}_k=\sum_{\boldsymbol{x}\in E_k}(\boldsymbol{x}-\boldsymbol{m}_k)(\boldsymbol{x}-\boldsymbol{m}_k)^T$$

$\mathbf{S}_W$: within-cluster covariance
$\mathbf{S}_B$: between-cluster covariance
$\mathbf{S}_T=\mathbf{S}_W+\mathbf{S}_B$: total covariance

Measures of error and cluster quality (II)

- To maximize the trace of $\mathbf{S}_W^{-1}\mathbf{S}_B$:

$$tr\left(\boldsymbol{S}_W^{-1}\boldsymbol{S}_B\right)=\sum_{i=1}^{n}\lambda_i$$

λ_i: i-th eigenvalue of $\mathbf{S}_W^{-1}\mathbf{S}_B$

- To maximize the trace of $\mathbf{S}_T^{-1}\mathbf{S}_B$:

$$tr\left(\boldsymbol{S}_T^{-1}\boldsymbol{S}_B\right)=\sum_{i=1}^{n}\frac{1}{1+\lambda_i}$$

- To maximize the determinant ratio between $\mathbf{S}_W$ and $\mathbf{S}_T$:

$$\frac{|\boldsymbol{S}_W|}{|\boldsymbol{S}_T|}=\prod_{i=1}^{n}\frac{1}{1+\lambda_i}$$

- Post-processing: the clusters of lower quality (for example, high quadratic error) can be splitted to reduce the error.

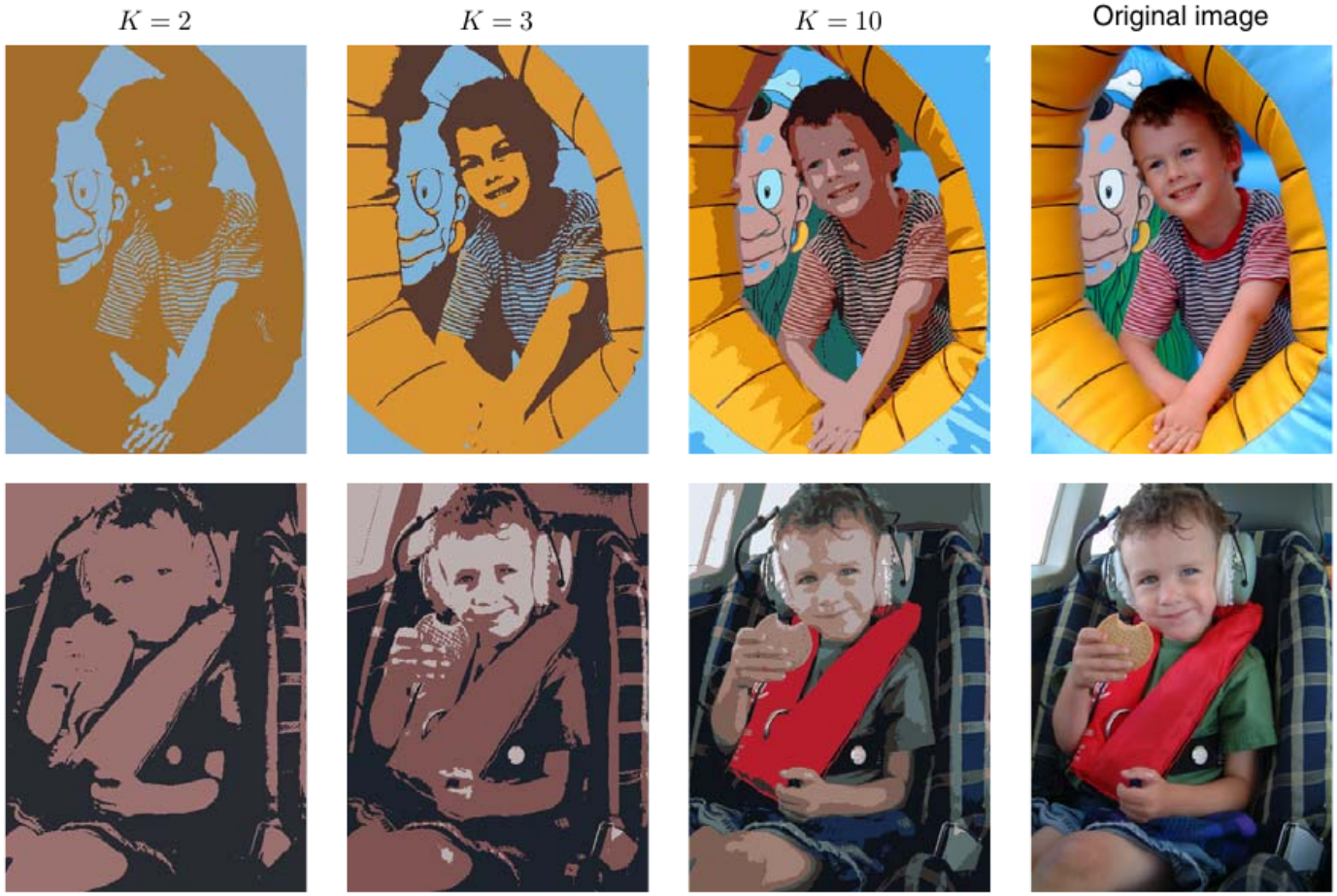
The number *K* of clusters

- It is a hyper-parameter which must be set before clustering.
- It can be tuned if there is some quality measure of the clustering.
- Sometimes, it is pre-defined by the problem.
- PCA (Principal Component Analysis) can be used to visualize the data in 2D and to select an appropiated number of clusters.
- Instead of setting *K*, we can set a maximum distance *D*:

1) If $d<D$, the pattern is asigned to the cluster and update its prototype.
2) If $d>D$ for all clusters, a new cluster is created. The final number *K* of clusters depends on data.

Example: image segmentation

- To divide a RGB image in regions with similar appearance.
- The K-means method shows the images with *K* colors.



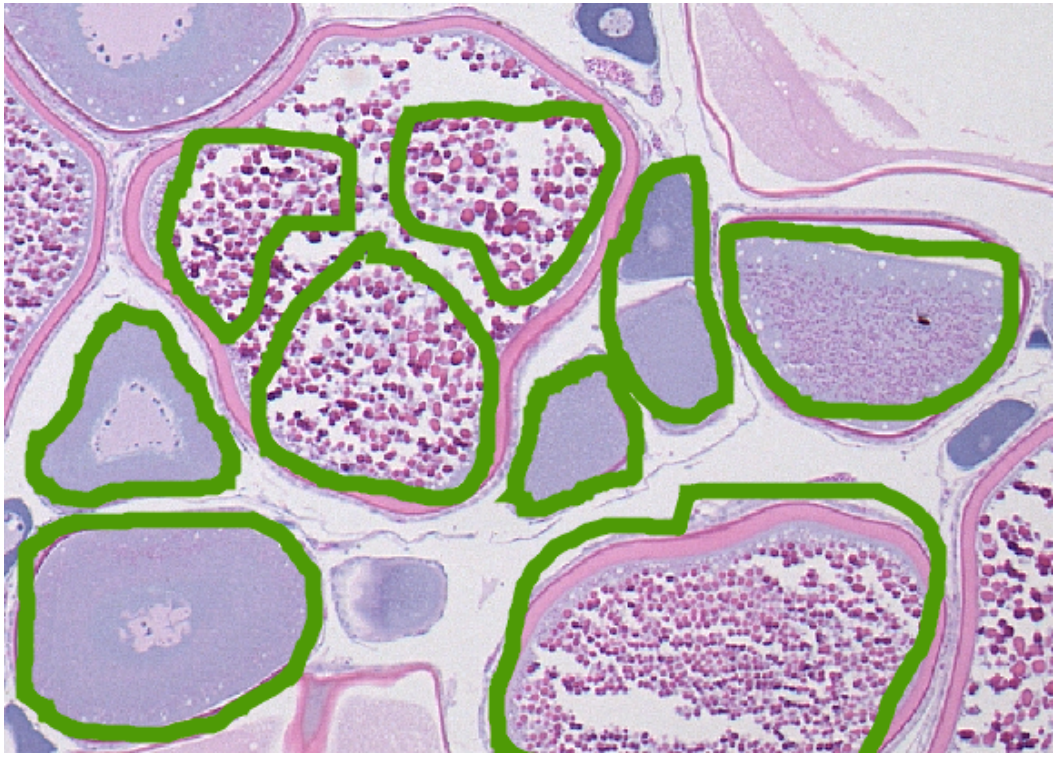
C. Bishop. Pattern recognition and machine learning

Image segmentation: performance

- **Pixel level evaluation:**
  - Assume that each pixel is a pattern
  - The image segmentation process is a two-class classification problem
  - Class 0 and 1 for background and foreground pixels
  - Use the quality measures for classification:
    - Kappa and accuracy
    - Precision, recall and F1
- The K-means method shows the images with *K* colors.

Image segmentation: performance

- **Region level evaluation:**
  - Both examples are rather bad



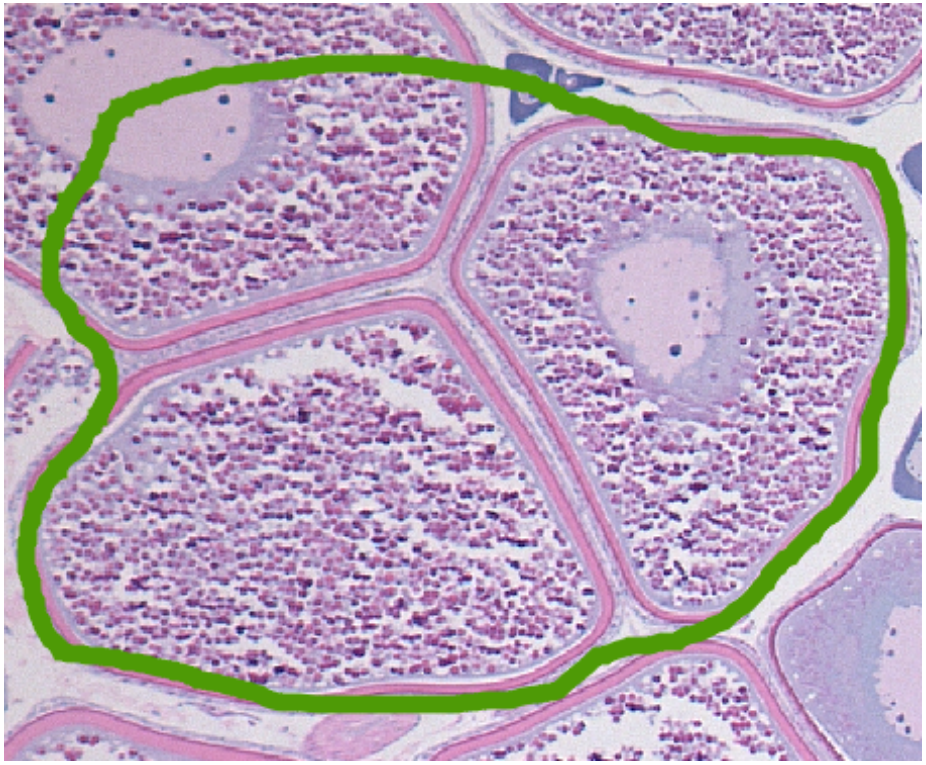


Image segmentation: performance

- **Region level evaluation:**
  - Quantify the agreement of expert and algorithm at region level.
  - Measure the overlap between the expert and computer to segment objects.
  - D^i and A^i the number of segmented and true objects in image I^i.
  - P_d^i, number of pixels of recognized object R_d^i, $d=1..D^i$
  - P_a^i, number of pixels of true object R_a^i, $a=1..A^i$

Image segmentation: performance

- **Region level evaluation:**
  - $O_{da}^{i} = P_d^{i} \cap P_a^{i}$ number of overlapped pixels between R_d^{i} and R_a^{i}
  - $O_{da}^{i} = \varnothing$ both regions are not overlapped.
  - $O_{da}^{i} = R_d^{i} = R_a^{i}$, the overlapping is complete.

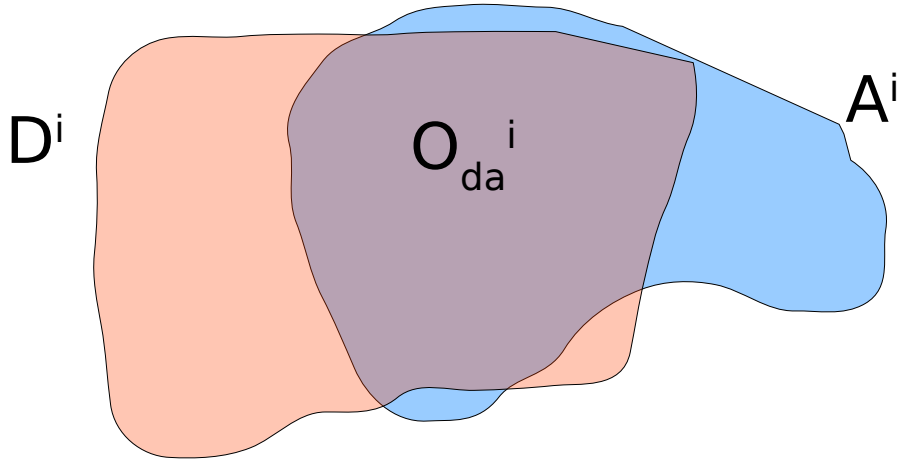
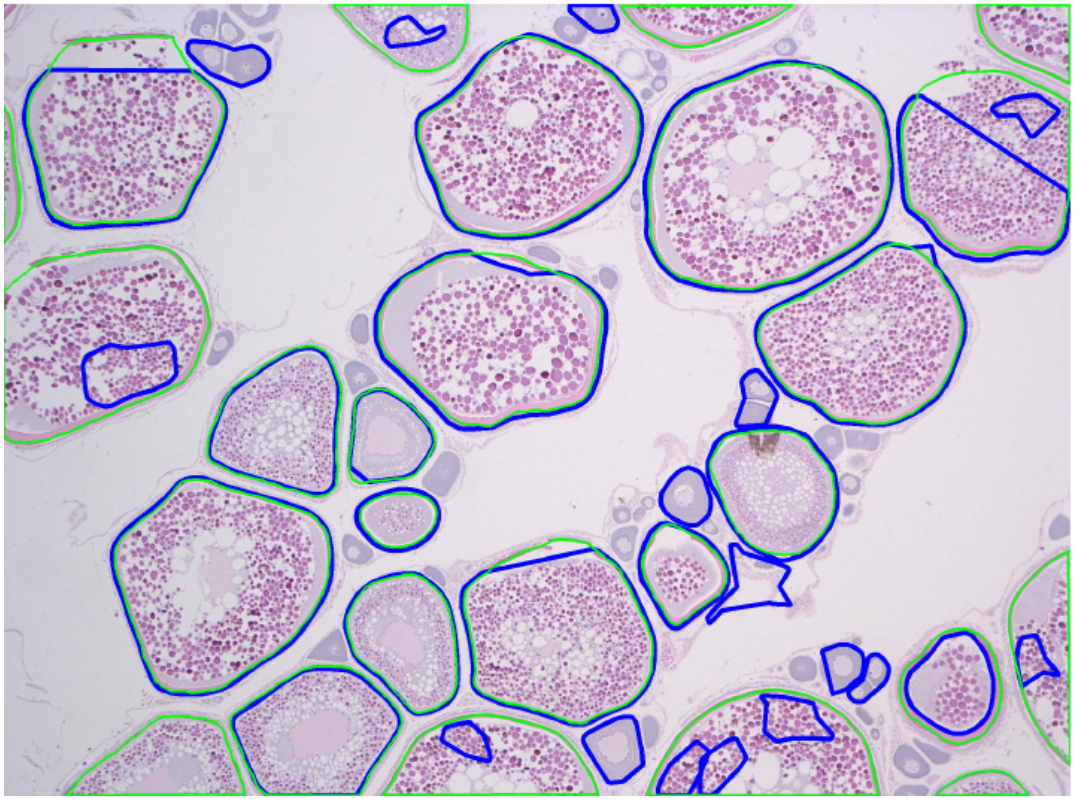


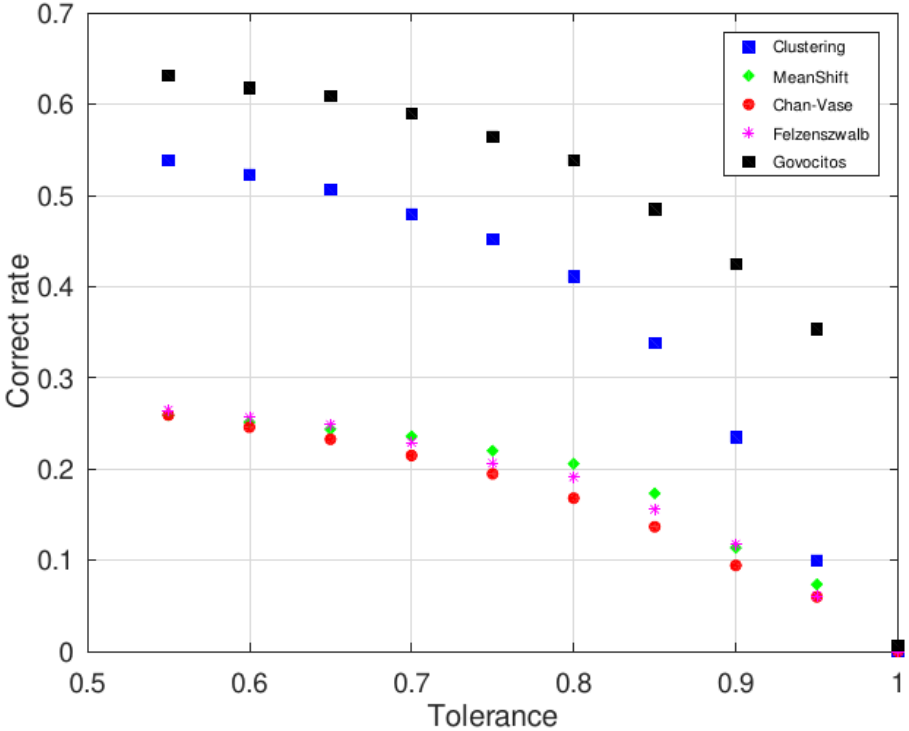
Image segmentation: performance

- **A region is classified in the following types:**
  - Let $0 \leq T \leq 1$ is a strictness criterion
  - **Correct:** a region R_a^i is correctly detected if $O_{da}^i \geq P_d^i T$
  - **Missed:** a region R_a^i that it is not correctly detected is missed.
  - **Noise:** a region R_d^i that does not participate in any instance of correct detection is classified as noise.

Image segmentation: performance

- **Example:** “MSCF: Multi-Scale Canny Filter to Recognize Cells in Microscopic Images”, 2023.
- https://doi.org/10.3390/su151813693





Example: image segmentation

- Matlab: function **imsegkmeans()**

K-means clustering based image segmentation

collapse all in page

Syntax

```
L = imsegkmeans(I,k)
[L,centers] = imsegkmeans(I,k)
L = imsegkmeans(I,k,Name,Value)
```

Description

`L = imsegkmeans(I,k)` segments image `I` into `k` clusters by performing k-means clustering and returns the segmented labeled output in `L`. example

`[L,centers] = imsegkmeans(I,k)` also returns the cluster centroid locations, `centers`. example

`L = imsegkmeans(I,k,Name,Value)` uses name-value arguments to control aspects of the k-means clustering algorithm.

Example: K-means clustering

- Matlab/Octave: function **kmeans()**

kmeans

K-means clustering

Syntax

```
IDX = kmeans(X,k)
[IDX,C] = kmeans(X,k)
[IDX,C,sumd] = kmeans(X,k)
[IDX,C,sumd,D] = kmeans(X,k)
[...] = kmeans(...,param1,val1,param2,val2,...)
```

Example: image segmentation

- OpenCV: function kmeans() (see OpenCV help)

Functions

double **cv::kmeans** (**InputArray** data, int K, **InputOutputArray** bestLabels, **TermCriteria** criteria, int attempts, int flags, **OutputArray** centers=**noArray**())

Finds centers of clusters and groups input samples around the clusters. More...

template<typename _Tp , class _EqPredicate >

int **partition** (const std::vector< _Tp > &_vec, std::vector< int > &labels, _EqPredicate predicate=_EqPredicate())

Splits an element set into equivalency classes. More...

Detailed Description

Function Documentation

◆ kmeans()

```
double cv::kmeans ( InputArray        data,
                    int               K,
                    InputOutputArray  bestLabels,
                    TermCriteria      criteria,
                    int               attempts,
                    int               flags,
                    OutputArray       centers = noArray()
                  )
```

Python:

retval, bestLabels, centers = cv.kmeans(data, K, bestLabels, criteria, attempts, flags[, centers])

Example: image segmentation

- Sklearn: module cluster (python)

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

scikit learn | Install | User Guide | API | Examples | Community | More | Go

Prev | Up | Next

scikit-learn 1.2.0
Other versions

Please cite us if you use the software.

sklearn.cluster.KMeans
KMeans
Examples using
sklearn.cluster.KMeans

Toggle Menu

sklearn.cluster.KMeans

class sklearn.cluster.**KMeans**(*n_clusters=8, *, init='k-means++', n_init='warn', max_iter=300, tol=0.0001, verbose=0, random_state=None, copy_x=True, algorithm='lloyd'*) [source]

K-Means clustering.

Read more in the User Guide.

Parameters:

n_clusters : ***int, default=8***
The number of clusters to form as well as the number of centroids to generate.

init : ***{'k-means++', 'random'}, callable or array-like of shape (n_clusters, n_features), default='k-means++'***
Method for initialization:

'k-means++' : selects initial cluster centroids using sampling based on an empirical probability distribution of the points' contribution to the overall inertia. This technique speeds up convergence. The algorithm implemented is "greedy k-means++". It differs from the vanilla k-means++ by making several trials at each sampling step and choosing the best centroid among them.

'random': choose `n_clusters` observations (rows) at random from data for the initial centroids.

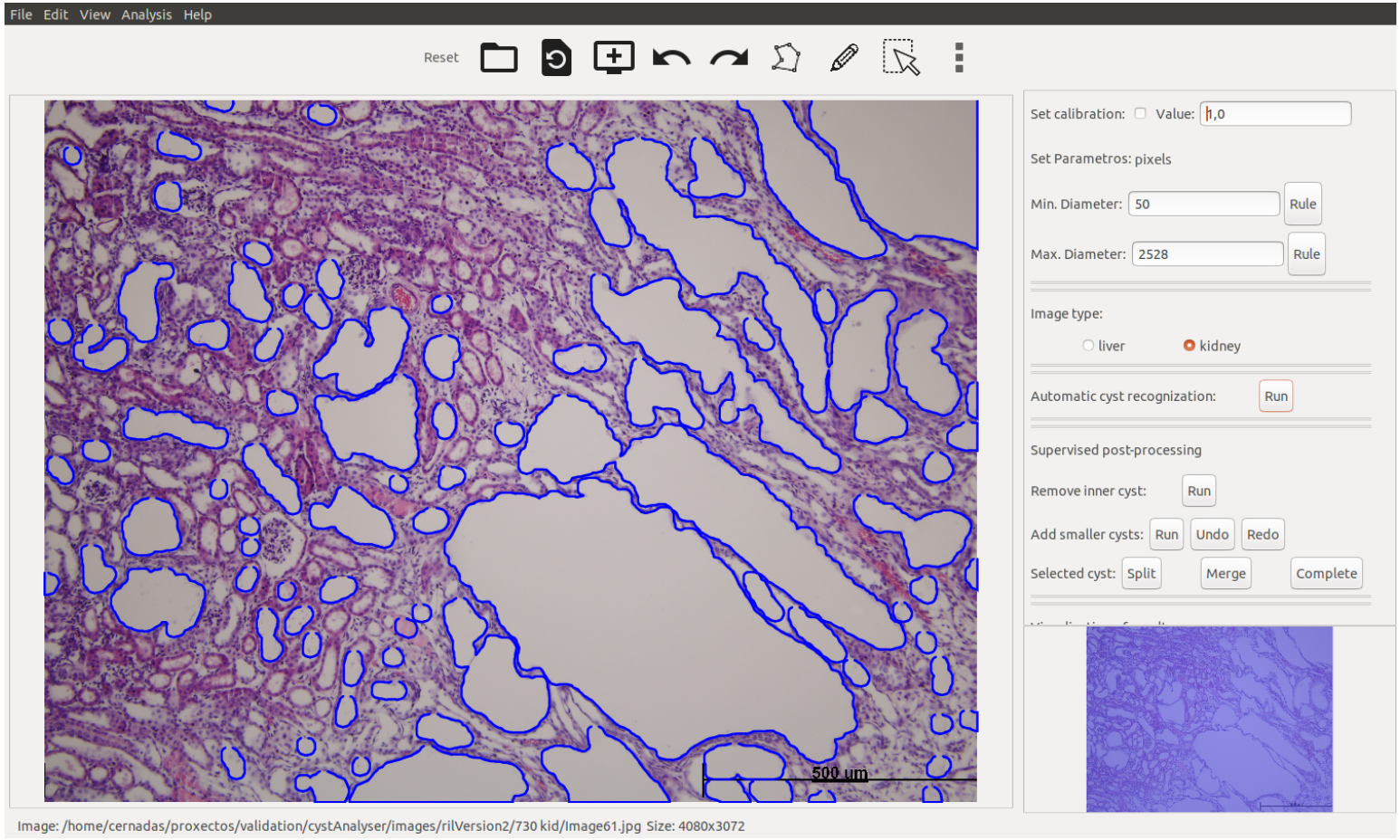
If an array is passed, it should be of shape (n_clusters, n_features) and gives the initial centers.

If a callable is passed, it should take arguments X, n_clusters and a random state and return an initialization.

n_init : ***'auto' or int, default=10***
Number of times the k-means algorithm is run with different centroid seeds. The final results is the best output

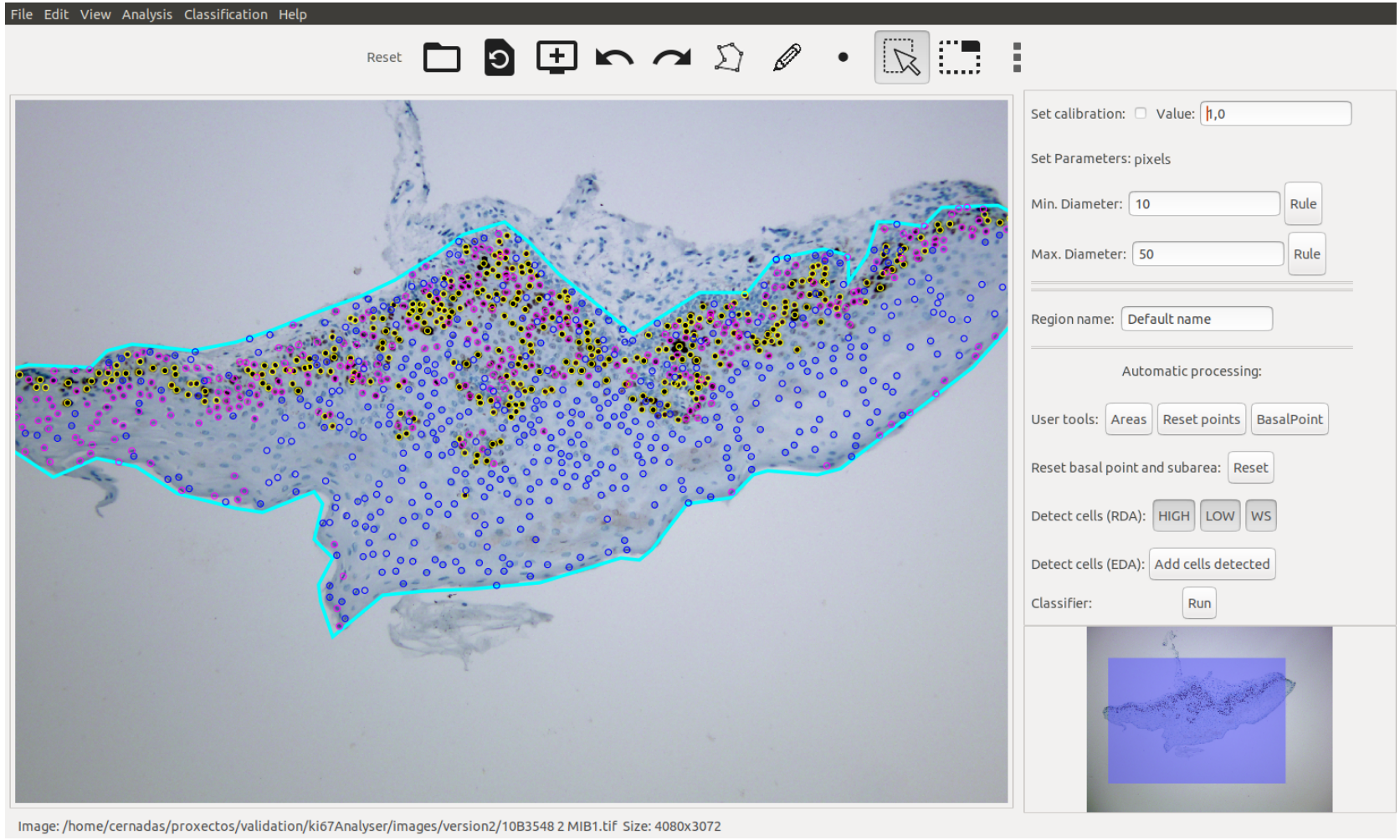
Real examples of clustering

- **CystAnalyser**: 10.1371/journal.pcbi.1008337 or https://citius.usc.es/transferencia/software/cystanalyser



Real examples of clustering

- **OralImmunoAnalyser**:



Example: image compression

- Image N pixels, 1 pixel=3 colors x 8 bits/color=24 bits: size=24N bits.
- Using K ($\ll N$) prototypes, an input pattern is a pixel $\mathbf{x}_i=(R_i,G_i,B_i)$.
- Clustering: K prototypes $\{\mathbf{p}_k\}_{j=1}^{K}$, with $\mathbf{p}_k=(R_k,G_k,B_k)$, stored only once.
- Compression: pixel $\mathbf{x}_i$ (8x3 bits) → prototype index $\mathbf{p}_k$ of size $\log_2 K$
- Size of compressed image: number N of pixels multiplied by the number of bits required to code the number K of prototypes ($N\cdot\log_2 K$ bits) and the K prototypes (24K bits)
- Compresion factor: $\alpha=\frac{T_0}{T_1}=\frac{24N}{24K+N\log_2 K}\approx\frac{24}{\log_2 K}$ for large N
- With K=10, α=7.2