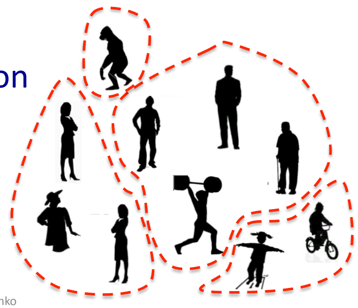
IAML: K-means Clustering

Victor Lavrenko and Nigel Goddard
School of Informatics

Semester 1

Clustering

- Discover the underlying structure of the data
  - unsupervised task, not predicting anything specific
- What sub-populations exist in the data?
  - how many are there?
  - what are their sizes?
  - do elements in a sub-population have any common properties?
  - are sub-populations cohesive? can they be further split up?
  - are there outliers?

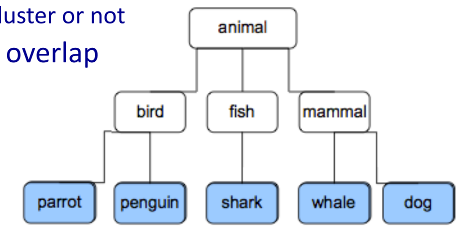


Overview

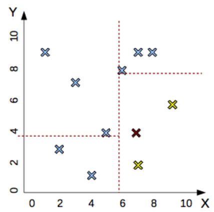
- Clustering
- K-means algorithm
- Practical issues: local optimum, selecting K
- Evaluating clustering algorithms
- Application: image representation
- Reading:
  - Witten & Frank sections 4.8 and 6.6

Types of clustering methods

- Goal:
  - monothetic: cluster members have some common property
    - e.g. all are males aged 20-35, or all have X% response to test B
  - polythetic: cluster members are similar to each other
    - distance between elements defines membership
- Overlap:
  - hard clustering: clusters do not overlap
    - element either belongs to a cluster or not
  - soft clustering: clusters may overlap
    - “strength of association” between element and cluster
- Flat or hierarchical
  - set of groups vs. taxonomy



Methods we will cover



- K-D Trees (see k-NN lecture)
  - monothetic, hard boundaries, hierarchical
- K-means clustering
  - splits data into a specified number of populations
  - polythetic, hard boundaries, flat
- Gaussian mixtures (EM algorithm)
  - fits a mixture of K Gaussians to the data
  - polythetic, soft boundaries, flat
- Agglomerative clustering
  - creates an "ontology" of nested sub-populations
  - polythetic, hard boundaries, hierarchical

K-means clustering algorithm

- Input: K, set of points $x_1 \dots x_n$
- Place centroids $c_1 \dots c_K$ at random locations
- Repeat until convergence:
  - for each point x_i:
    - find nearest centroid c_j $\quad \arg\min_j D(x_i, c_j)$
    - assign the point x_i to cluster j

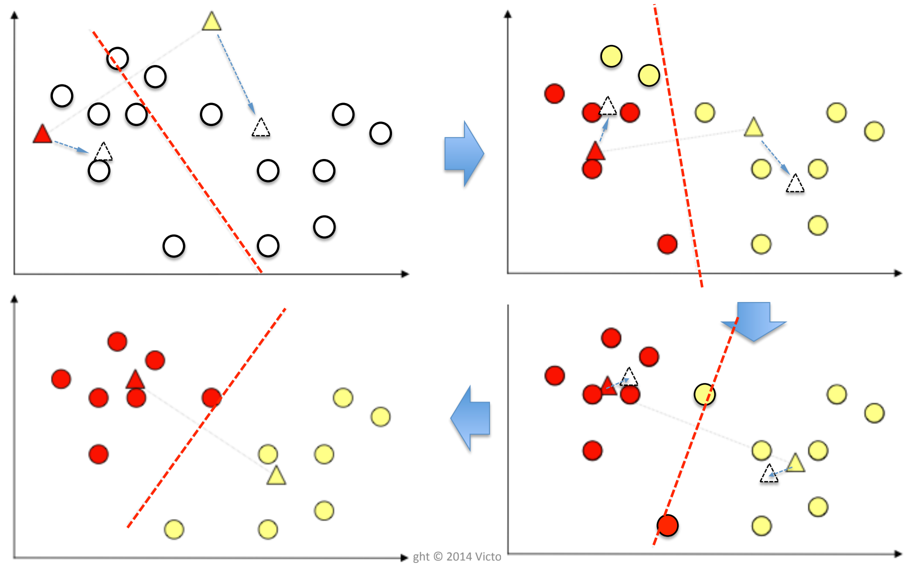
    distance (e.g. Euclidian) between instance x_i and cluster center c_j
  - for each cluster j = 1 … K: $\quad c_j(a) = \frac{1}{n_j} \sum_{x_i \to c_j} x_i(a) \quad \text{for } a = 1..d$
    - new centroid c_j = mean of all points x_i assigned to cluster j in previous step
- Stop when none of the cluster assignments change

O (#iterations * #clusters * #instances * #dimensions)

K-means clustering

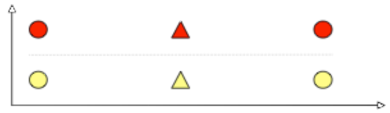
- Produces hard, flat, polythetic clusters
  - data partitioned into K sub-populations (need to know K)
  - points in each sub-population similar to a "centroid"
    - centroid = attribute-value "representation" of a cluster
    - "prototypical" individual in a sub-population
- Uses:
  - discover classes in an unsupervised manner
    - e.g. cluster images of handwritten digits (with K = 10)
  - smoothness over space
    - in the same cluster ➔ similar representations / class labels / …
  - dimensionality reduction: clusters = "latent factors"
    - replace representation of each data point with its cluster number
    - assumes all pertinent qualities reflected in cluster membership
    - related to basis / kernels in linear classifiers

K-means clustering example



K-means properties

- Minimizes aggregate intra-cluster distance $\sum_j \sum_{x_i \to c_j} D(c_j, x_i)^2$
  - total squared distance from point to centre of its cluster
  - same as variance if Euclidian distance is used
- Converges to a local minimum
  - different starting points ➔ very different results
  - run several times with random starting points
    - pick clustering that yields smallest aggregate distance
- Nearby points may not end up in the same cluster
  - the following clustering is a stable local minimum:



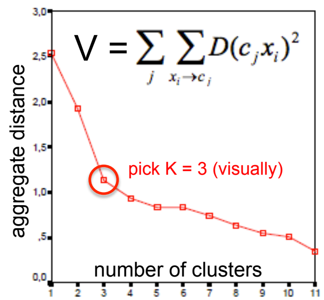
Evaluating Clustering Algorithms

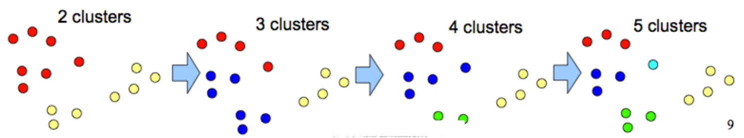
- Extrinsic (helps us solve another problem)
  - represent images with cluster features
  - train different classifier for each sub-population
  - identify and eliminate outliers / corrupted points

  } did your classifier improve?
- Intrinsic (useful in and of itself)
  - helps understand the makeup of our data (qualitative)
  - clusters correspond to classes (digits → 10 clusters)
    - align, evaluate as you would a normal classifier
  - compare to human judgments
    - can't ask humans to "cluster" a dataset manually
    - sample pairs x_i, x_j ask humans if they "match"

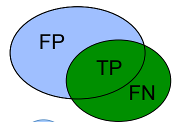
Optimal number of clusters

- How many clusters are there in your data?
  - class labels may suggest the value of K (e.g. digits 0..9)
  - optimize distance V: for K = 2,3,…
    - run K-means, record distance
    - problem: V minimized when K = n
      - what if we use a validation set?
    - W&F: Minimum Description Length
      - total bits to encode K centroids + V
    - visually from scree plot:
      - point where "mountain" ends, "rubble" begins
      - elbow method: maximize 2nd derivative of V: point where rate of decline changes the most

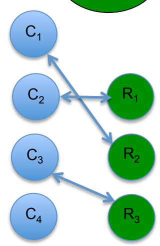




Intrinsic Evaluation 1



- System produces clusters C_1 C_2 … C_K
- Reference clusters (classes) R_1 R_2 … R_N
- Align up $R_i \Leftrightarrow C_j$, measure accuracy, F1, …
  - many different ways to align:
    - Weka: C_j → R_i with max overlap
  - if many C_j → same R_i :
    - re-assign in a greedy manner
    - non-greedy: K!/(N-K)! ways (very slow)
  - can we have multiple C_j → same R_i?
  - can we have multiple R_i → same C_j?
  - can we have overlapping clusters?



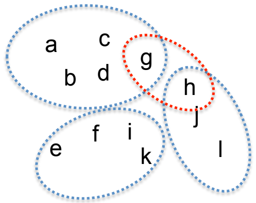
true class → / cluster →	R2	R1	R3	
C1	3	1	2	6
C2	0	0	1	1
C3	7	1	8	16
C4	2	0	1	3
	12	2	12	

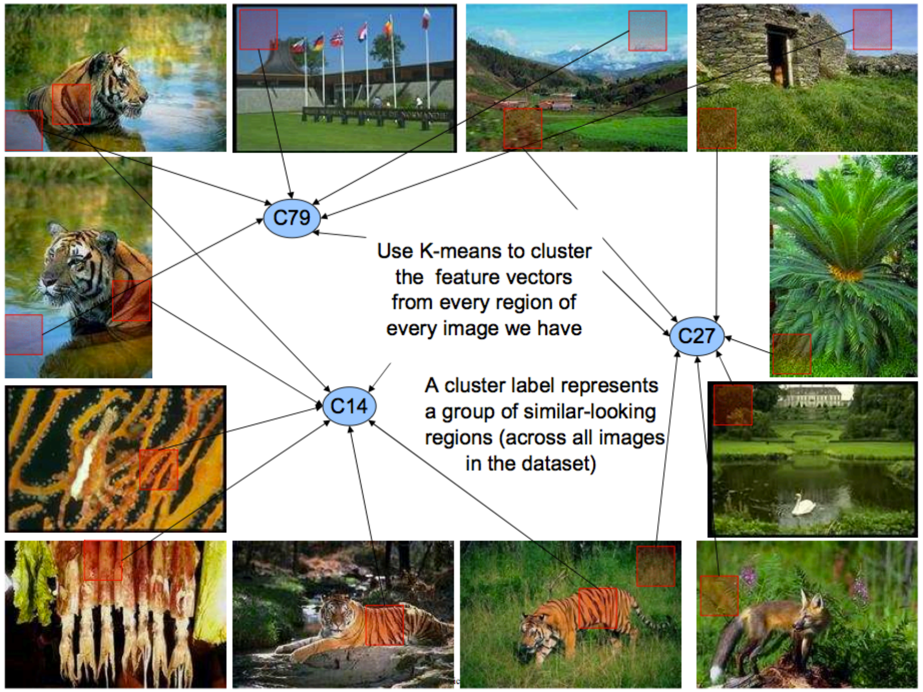
Accuracy = (3+0+8)/26

Intrinsic Evaluation 2

- Sample pairs x_i, x_j
  - ask human if x_i, x_j should be in the same group
  - easy task (cognitively)
  - can't ask them to "cluster" dataset manually
- System produces clusters
- Count errors, compute accuracy, F1, etc
  - FN: matching pairs x_i, x_j that are in different clusters (e,h)
  - FP: non-matching pairs x_i, x_j that are in same cluster (c,d)
- Doesn't require a pairing strategy
- Can handle overlapping clusters (a bit tricky)
  - same pair can count as both TN and FP (g,h = No)
- Can generate pairs from classes

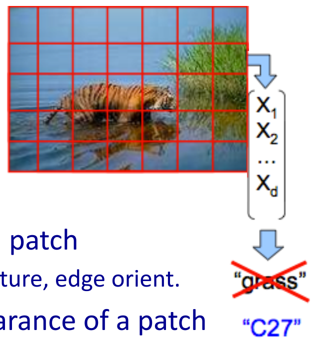
a,b = Yes
c,d = No
e,h = Yes
g,h = No





Application: image representation

- Goal: detect presence / absence of objects in image
- First step: represent images as attribute-value pairs
  - pixels as attributes: $10^3 \times 10^3 \times 10^3$ (conservative)
  - large and not very meaningful for learning
  - bag-of-words would be nice
    - {"water", "grass", "tiger", "cat", "ripples"}
    - requires human annotation
  - break image into a set of patches
    - patch = part of some object
  - compute appearance features for each patch
    - relative position, distribution of colors, texture, edge orient.
  - convert to a "word" that reflects appearance of a patch
    - similar-looking feature vectors ➔ same word to represent them

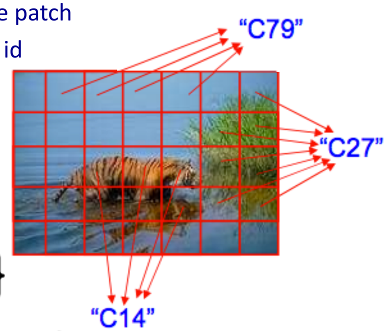


K-means for image representation

- What K-means does:
  - groups all feature vectors from all images into K clusters
  - provides a cluster id for every patch in every image
    - represents the salient properties of the patch
    - similar-looking patches have the same id
- Represent patch with cluster id
  - image = bag of cluster ids
    - one for each patch in the image
  - K-dimensional representation:

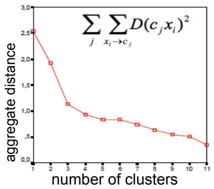
{4 x "C14", 7 x "C27", 24 x "C79", 0 x everything else}

  - similar to bag-of-words
  - cluster ids sometimes called vis-terms or "visual words"



Summary

- Clustering: discover underlying sub-populations
- K-means
  - fast, iterative method: O(i*K*n*d)
  - converges to a local minimum of $\sum_j \sum_{x_i \to c_j} D(c_j, x_i)^2$
    - run several times with different starting points
  - need to pick K: use scree plot
  - need to pick distance function (Euclidean)
  - nearby points may end up in diff. clusters



- Application: image representation
  - cluster image patches based on visual similarity
  - cluster numbers (vis-terms) becomes attributes

– Evaluation: intrinsic vs. extrinsic

Clustering: general structure

- Task: unsupervised / generative
  - group instances into K clusters
- Model structure
  - K cluster centroids (d-dimensional vectors)
- Score function
  - average distance from instance to cluster centre
- Optimization / search method
  - iteratively re-assign instances to clusters and update cluster centroids