CLUSTERING

Not so used like multiclass classification

We have many alg for it

We see the 3 most famous, to be able to face all the others:

SUM

Supervised learning finished

Unsupervised learning with clustering; but others are:

- Dimensionality reduction

Unsupervised: don't have true label in dataset (no the col of the label), no target var, but only samples and features

So is more difficult than supervised learning (no true label: truth)

Is much more subjective than supervised learning

We don't know how good are the prediction => unsupervised is used for exploratory data analysis

- customer segmentation

But this is a starting point: custom not from same distribution: from diff universe doesn't work - use same distribution and then final goal

If clusterization is not good, also other goals is not good

- so imp but diff to get a good result

- So it’s hard to asses the result

We get the result without a measure enough robust to say how good is the clustering in doing classification (divide in class)

Is impossible to be sure of the results

From theoretical p.o.v. is difficult

- fit analytical model with data based on true result => fit for best prediction - no way to check goodness of the results

Which is the goal of clustering?

Find a natural grouping in data=> items in same cluster more similar than other of diff clusters

Universal def doesn't exists: group data such that data in same cluster are similar > natural grouping: we want to get out from data => if we know that in application domain there is no natural grouping

Before clustering understand if it makes sense to have clusters/groups How good the cluster are is a question of the domain

Clustering: group data in group of cluster

Assign each sample of the dataset to a group: from formalization pov is similar to classification (each sample to a class, here to a group) => assign data

Is different the reliability:

- method of true label

- no true label: method diff

So same goal but different methods

Clustering example:

- dataset of 2 features: x1, x2 => just have data: is there a natural grouping in the dataset?

We have 3 groups (a priori say that we classify in 3 groups = domain give the intuition of natural grouping

But in reality this not happen: we have globes of data

Similarity:

similar points in same cluster

But what is the concept of similarity in data/problem?

Is intuitive in some task, but difficult to decide the similarity measure in other tasks - Euclidean distance as similarity measure

very often other similarity measures (more useful)

- Evaluate similarity on how people use credit cards=food/travel/ecc Euclidean here is not good => use another similarity measure

Which is the similarity concept of data?

1) think what is the similarity concept

2) what is the best similarity measure for it

This is how we design clustering project

We know both, now we develop the alg for the clustering:

some classes of alg summarize all approaches

There is a class of alg, where I define the center of similarity

- K-means alg: (center oriented alg)

I have data: find center of the cluster and then group the data around it Works well in small data: small computational coast, but limitations

- Bottom-up approach: build tree=hierarchy of the clusters

Work well with nested clusters

- To identify arbitrary shape of clusters with density based approach Feature space: a cluster is a region where the density of points is higher than other parts

Learn the denser region => id cluster in them

**K MEANS (1st class of clustering methods)**

Called prototype-based clustering (center=prototype: I have 1 point that rep the class) Use different concept of prototype: centroid (average of similar points in a continuous space)

Partition the dataset in k(hyperpar) distinct non-overlapping clusters (diff separated circles)

If observation i is in cluster k, then it belong to class CK

K-means: define how many cluster to get a priori (hyperparameter)

Fix before the alg the n of clusters to get => limit of k-means (usually no idea of n of clusters to make for a good clustering, especially with many data)

K-means: specify the desider n of clusters k and assign each observation to exactly 1 of the k clusters: no overlapping and assign to a cluster

Limitation: all points assigned to a cluster => get best cluster considering all points

We have n observations, m features, k clusters (denoted as Ci)

Ci={all the points that bel to it}

Cluster=set of indices of observations that bel to that class

2 properties:

1) each observation bel to al least 1 class => union is the set of observation 2) each observation bel to at most 1 class => intersection is none

K-means: we want diff n of clusters

K=1: 2 clusters

K=3: 3 clusters

K=4: 4 clusters

With clusters we see color: color according to the diff clusters

The order of the cluster is the same, even if we have diff colors (no order in classes of clusters Ci)

THE ALG

Idea: a good clustering is the one for which the within cluster variation is as small as possible The variation is the smallest as possible

We denote it as W(Ci)

We measure it as the amount by which the observations within a cluster differ from each other

Divide in i clusters such that the total variation, summed over all k clusters, is small as possible

We have to balance: other clusters not so nice

Minimize over all possible partitio of data into k clusters:

- sum over all k clusters variation of k cluster

So k-means minimize this total variation

This is the optimization function:

e.g. use concept of similarity (square euclidean distance) write the intra class variation as: We want to have it as small as possible

Which is the similarity we use? Depending on its value we get diff results Sum over all possible point pairs and divide by n of points in the cluster

So from general optimization function, we open the variation and we write it according to the similarity measure we use

If we optimize we get this objective function

- minimize over all possible partitions, the sum over all k clusters of the intra cluster variation given by euclidean distance

1/Ci to get the average

We have k^n ways to get the clusters

SO:

k-means want to try an optimal solution for this optimization problem

1)

Initialization: decide how to pick the starting centroid=many diff ways to initialize it (diff initialization = diff results, but is just a matter of converge time: no very dissimilar results; with good starting point we get a better convergence)

- randomly pick the centroids, deciding which of the n observation will be the k centroid So now we have k points: prototypes of the k clusters

2)

Assign each sample to the nearest centroid

Each sample assigned to the centroid = most similar to point (nearest, according to the sim measure we use)

Each observation is assigned to a cluster rep by 1 of this centroid

3)

move the centroid to the samples that were assigned to it

4)

Repeat 2-3 until a stop rule happen (many different)

- can't improve assignments: don't move centroids => stop

So, even if 1) and stopping rules are very different we dont get diff results We get the local optimum solution

Example:

- Same data of before

K=3 => fix that we want 3 classes

We use a random assignment as initial step

Without reasoning dont fix center of centroid, but assign randomly the points Compute new centroid by the average of the points

We get this 3 centroids=orange-green-pink

Assign again all the points wrt to centroid

Now we have new clusters: repeat=find new centroids (average of the points of the cluster)

Then assign data to new centroid (orange before, now green)

Repeat until no new modification (here this is reached after 10 iterations) So just move centroids and re assign points

How we formalize what the alg does:

Sum of the square err of the points => minimize sum of square err

At each step, the centroid of cluster j, and compute the square norm of distance with centroid and all points

Consider with the sum all points that bel to clsuter j (only them) and sum over all observations

K\_means: optimization problem with an approximation of it at the beginning => find a local minimum

We can obtain diff results, but not so diff from each other=consider the one with min square error

Example:

- k=3

with diff random assignment in step 1

According to the value of the square err we get best k-mean models

K-means ++=alg used to have a fast convergence of the alg

Tricky point: if we start with good centroid=faster to get res

So get centroids very distant=class more separated since the initiation

NEW DATA

How to predict the good cluster for a new points

- custom segmentation: have history perform segmentation

If new custom, assign it to best cluster=assign to closest cluster center

With new data we dont make only set of points, but we partition feature space Assign a new point to the closest center

HARD VS SOFT CLUSTERING

Hard: each sample is assign to exactly 1 cluster

Soft/fuzzy: give a prob to the point, to bel to the cluster

- seems better, but without label is more difficult

Use fuzzy k means for it, to check how it works

K-meas is the most used alg=understand how it works

Alg is so simple=2 steps

Is the most common used alg

DRAWBACKS:

1) fix a priori the n of clusters, but we don't have to fix other hyperparameters 2) based only on the center of the cluster=build points around the center No cluster with complex shapes (like a spiral)

Is only goos with really good data=no complex shape in this case

With a domain where cluster are diff according to the feature we have: diff scale=complex scale => k means doesn't work (all features have same role for it, but this dep on the domain)

It can only find balls

ELBOW METHOD

Hyperpar: n of clusters

To decide the best value of it use elbow method

- run over diff values of this hyperpar

- If high n of clusters, then distorsion increase fast

best n of clusters is the elbow point

No theoretical reason about this (don't know a priori the best n of the hyperpar) try many times with diff k

**HIERARCHICAL CLUSTERING**

Nice because dont fix n of clusters before alg= great advantage

Find cluster with hierarchy

Dendrogram: tree based rep

2 kinds of clustering alg:

1) Divisive: alg where we start 1 cluster (embed all examples) then split in small clusters 2) Agglomerative: each sample in a diff cluster and then merge

SIMILARITY MEASURE

Find group of points similar in 1 class

Fix it: e.g. uclidean distance

Then diff ways to know if 2 clusters are similar or not

With similarity between cluster=obtained in diff ways => single leakage (compute distances with most similar members for a pair of clusters)

- blue-orange

Define as cluster

Complete leakage: define distance between cluster as the most dissimilar members for each pairs of clusters

Average linkage: merge the 2 clusters with smallest average d between all points Ward linkage: pick 2 clusters to mere such that the sum of the squared err increased at least

If n observations=start with n clusters

Then compute d of all pairs of clusters

- you have a matrix

Each el rep the distance between 2 clusters in that el

When er have all the distance merge into a unique class the 2 clusters more similar Update d matrix = decreasing n of clusters for merging

Repeat until we get 1 single cluster

Example:

- 2 features (we can only visualize it)

Start with n clusters = compute distance matrix (9x9 matrix)=compute only a part Cluster 5-7 smallest d=so merge them => cluster with 2 data points, diff from others with only 1 data point

Repeat: merge 2 clusters with smallest distance

Go on and repeat

Similarity measure between points is euclidean distance, while between clusters is complete linkage

**How to construct the dendrogram**

We have the vertical axis=give the d between clusters

- cluster 5-7=first part of the tree

- cluster 1-6(greater d than before, but is the 2 least d)

- Then cluster 5-7 with 8

- Follow the tree to see all the clusters

- At the end merge the 2 big clusters to make the single 1 = end

Each branch is a cluster at diff level of the tree

**How to read the dendrogram**

9-2 near in dendrogram=similar => not true

We have to read in vertical axis not in horizontal

Similarity is checked only in the vertical line: give the d

We can't say that all these observations are similar to observation 3: because similar to vertical axis

Example:

- 45 observations

clusters are evident=write agglomerative alg

Euclidean distance and complete linkage as similarity functions

Don't fix a priori the n of clusters:

1) build dendrogram

2) decide n of cluster you want =decide where to cut the tree (n of clusters chosen by yourself=pay attention to over/underfitting)

Example (for best perf of linkage)

- single: bad perf

- complete: best perf (shape of cluster not balanced=give idea of most dissimilar)=more balanced clusters

- average: also best

**Hierarchical clustering**

Works well only if we have a hierarchy in the clusters

- observation of a group of people 50-50 males-females among

american-japanese-french

Is better to use k-means here=no hierarchy

**DENSITY BASED CLUSTERING**

Overcome problem of k-means, and works also with hierarchical problem The most famous alg is DBSCAN: all based on idea that cluster is a dense region of the feature space

Empty spaces divide the clusters

How it works?

DBSCAN

Define density as n of point within a specific radius

Label each point in dataset according to the following notation

- Core point: another hyperpar of neighboring points fall within the specified radius So there are a mi n of points within the radius

Hyperpar: 3 points to have a core point

- Order/boundary point: point that has a fewer neighbors than MinPts within the radius, but lies in the core point

- Noise point: all the other points (no enough neighbors/no border of core point)

So:

Fix the 2 hyperpar

- n of neighbors

- radius

Select a random point to start (points with a d less than r)

Around corepoints build clusters

If not enough neighbors=noise point

Visit al neighbors: if no assigned => assign to the cluster created

Repeat

Start with core points: include points around it

If no neighbors: => start with another points

Repeat until no new points to catalog

In diff runs we can change the boundary points state(if another order we assigned to diff clusters)

But core and noise remain the same

Can deal very complex shapes: recognized a dense region in diff shape Get very complex shape if we have densest region

DBSCAN: dont fix n of cluster we want, but we fix 2 hyperpar=min n of neighbors and radius Change these 2 hyperp to get diff clusters

So we fix hyperpar for which the cluster dep

We can decide that some points are noisy points=no assigned to a cluster (not true that all points assigned to a cluster)=don't change the shape of the cluster

||

Better cluster shapes=imp when we have data not nice (no noise inside)

DBSCAN:

- consider that data can be noisy

- get complex shapes

**QUALITY OF CLUSTERING**

How to assess the quality of clustering=very difficult

We dont know the best alg

Since we are in unsupervised no label to judge

We use the silhouette coefficient:

1) compute cohesion a(i) of a sample x(i): compute the distance between it and all the other points of the same cluster

How much cluster are separated: points in the same cluster close, but from other we want distant

- average distance between the sample x(i) and all samples in the nearest cluster

Silhouette is the diff between cluster cohesion and separation divided by the greater of the 2 - Now we have it for each data point

Plot the silhouette as the silhouette coefficient of each point divide by cluster

Nice clustering:

- 1st cluster balance

- points have similar silhouette witch is the higher

We can have high average silhouette but unbalanced classes

Examples

We have a cluster with more points and silhouette higher than other

- we find a good class, but the other not => so we don't have enough cluster: use more

TESI

Vai dai vari prof, piace una materia, chiedi i progetti in questione

- prossima settimana tesi su argomenti di lab

- presentano linea di ricerca