Boring but important disclaimers:

- If you are not getting this from the GitHub repository or the associated Canvas page (e.g. CourseHero, Chegg etc.), you are probably getting the substandard version of these slides Don't pay money for those, because you can get the most updated version for free at

  `https://github.com/julianmak/OCES4303_ML_ocean`

  The repository principally contains the compiled products rather than the source for size reasons.
- Associated Python code (as Jupyter notebooks mostly) will be held on the same repository. The source data however might be big, so I am going to be naughty and possibly just refer you to where you might get the data if that is the case (e.g. JRA-55 data). I know I should make properly reproducible binders etc., but I didn't...
- I do not claim the compiled products and/or code are completely mistake free (e.g. I know I don't write Pythonic code). Use the material however you like, but use it at your own risk.
- As said on the repository, I have tried to honestly use content that is self made, open source or explicitly open for fair use, and citations should be there. If however you are the copyright holder and you want the material taken down, please flag up the issue accordingly and I will happily try and swap out the relevant material.

OCES 4303 :

an introduction to data-driven and ML methods in ocean sciences

Session 4: clustering

Outline

- motivation for clustering
  $\rightarrow$ some oceanographic examples
- clustering
  $\rightarrow$ K-means to demonstrate algorithm
- manifold learning revisited
  $\rightarrow$ funny data where K-means as is will not work
  $\rightarrow$ including LLE and t-SNE in the pipeline
  $\rightarrow$ beyond K-means (e.g. DBSCAN)
- demonstration of pipeline on penguins data

Recap: penguins

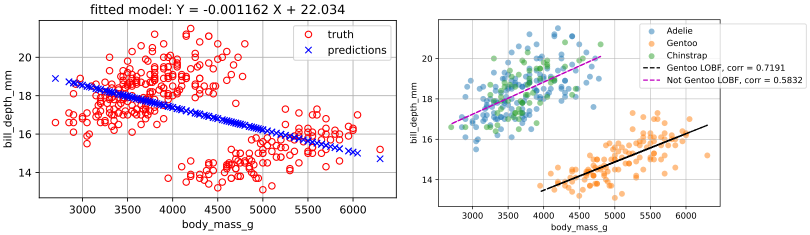


Figure: Contrived example of completely different models depending on data selection.

- if we had no labels in penguins data the linear regression is pretty bad (although it was never going to be good...)

  $\rightarrow$ use clustering to create labels?

  $\rightarrow$ e.g. use that to inform training of model, or to train different models

Oceanographic examples

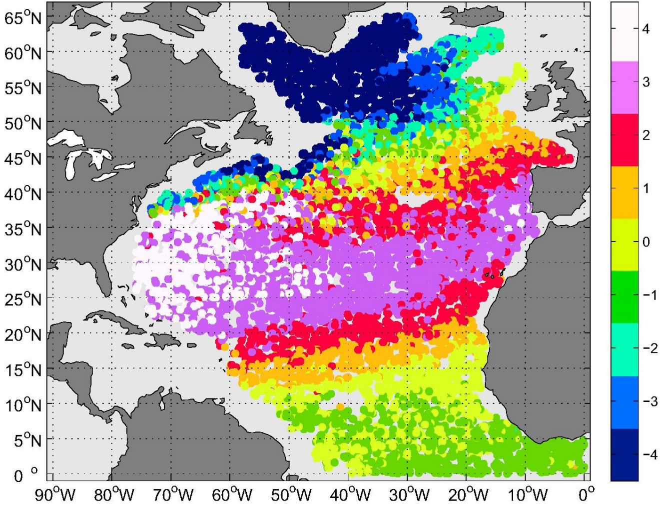


Figure: From Maze *et al.* (2017), Fig. 4. Gaussian Mixture Model to identify watermass clusters from Argo data in Atlantic.

Oceanographic examples

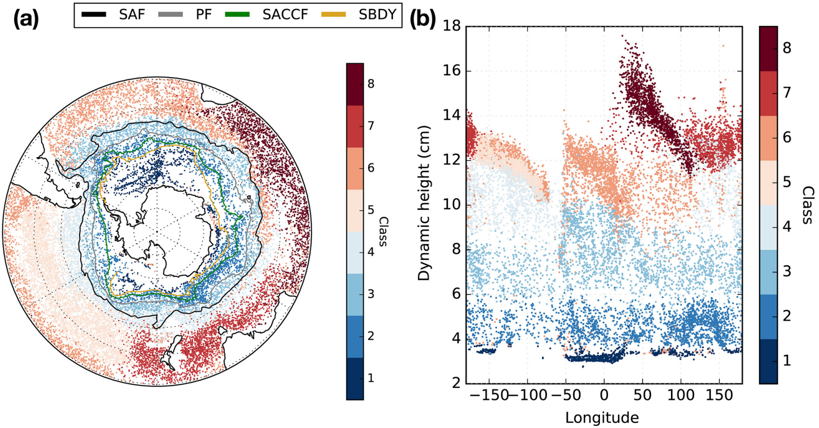


Figure: From Jones *et al*. (2019), Fig. 5. Gaussian Mixture Model to identify watermass clusters from Argo data in Southern Ocean.

Oceanographic examples

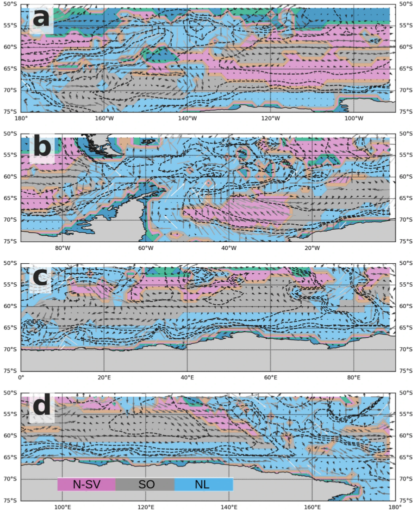


Figure: From Sonnewald *et al.* (2023), Fig. 4. *k*-means to identify clusters based on dynamic (from barotropic vorticity budget).

Oceanographic examples

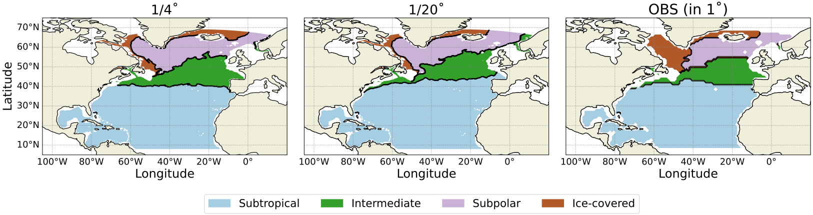
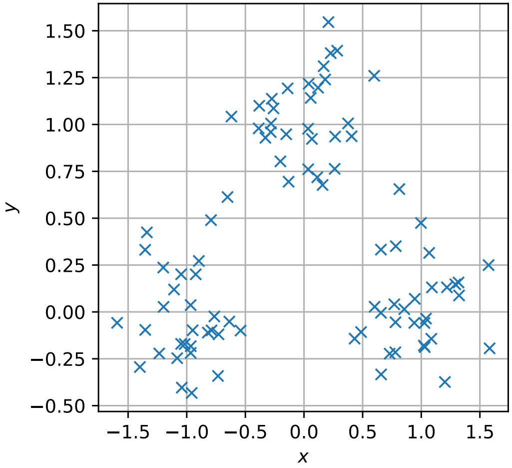


Figure: From Ruan *et al.* (in prep.), which identifies regions depending on biogeochemical activity.

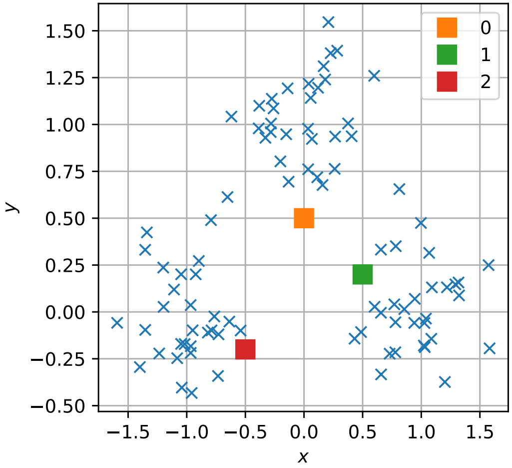
e.g. K-means

- demonstrate the algorithm with K-means
  $\rightarrow$ artificially create some data ($K = 3$ is sensible)



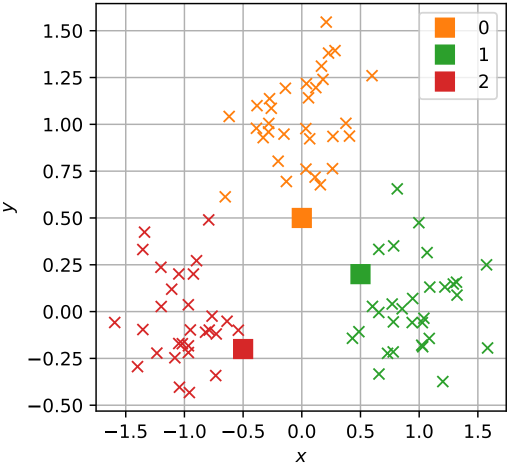
e.g. K-means

- provide initial centres of clusters
  $\rightarrow$ K-means is quite robust, can just guess



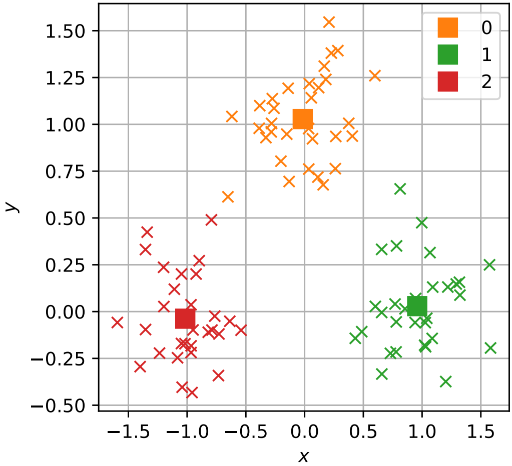
e.g. K-means

- label points closest to centres accordingly
  $\rightarrow$ distance dependent, normally L^2 (Euclidean distance)



e.g. *K*-means

- from new cluster, find the new location of the centre
  $\rightarrow$ update and iterate accordingly



K-means in `sklearn`

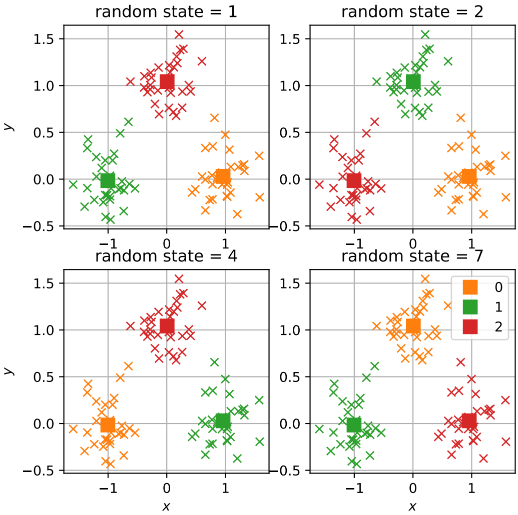


Figure: K-means through `sklearn`. Random state changes initial guess.

A case where K-means as is will never(?) work

- all of the above depends on the choice of ‘distance’ again

A case where K-means as is will never(?) work

- all of the above depends on the choice of ‘distance’ again
- there are well-known examples where the L^2 distance is simply not the relevant one
  $\rightarrow$ e.g. the moon data (2d example) and Swiss roll data (can do 2d or 3d)

Figure: Moon Moon and two rolled up towels resembling a Swiss roll.

A case where K-means as is will never(?) work

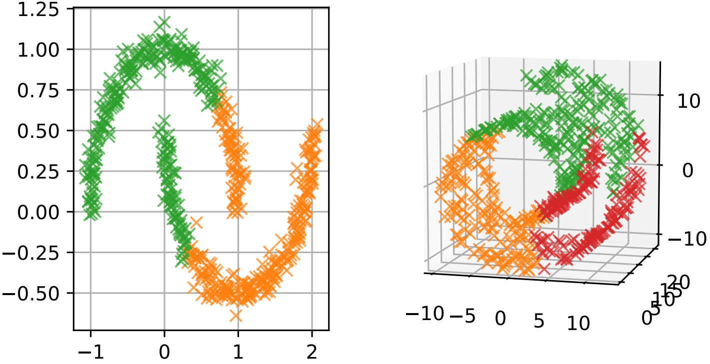


Figure: K-mean as applied to the crescent moons and swiss roll data.

The issue of 'distance'

- the data may have structure and lives on a curve/surface/volume etc. (e.g. manifold up to noise)

  → manifold may have a lower dimension structure (1d and 2d here)

  → but embedded in an ambient space ($\mathbb{R}^2$ and $\mathbb{R}^3$ here)

The issue of 'distance'

- the data may have structure and lives on a curve/surface/volume etc. (e.g. manifold up to noise)

  $\to$ manifold may have a lower dimension structure (1d and 2d here)

  $\to$ but embedded in an ambient space ($\mathbb{R}^2$ and $\mathbb{R}^3$ here)
- it is the distance intrinsic to the manifold that is presumably of interest

  $\to$ L^2 is the distance extrinsic to manifold and inherited from the ambient/embedding

- find a suitable embedding, as in dimension reduction previously?

  $\to$ reduce dimension first, then cluster

Moon data demonstration

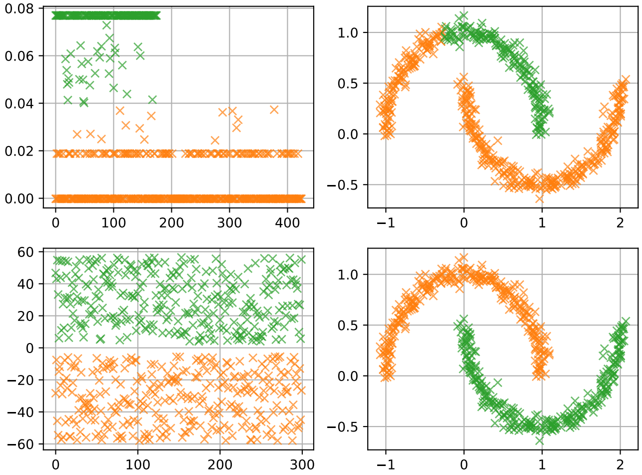


Figure: 1d projection (of 2d data) via (top) LLE and (bottom) t-SNE before K-means. LLE can work on occasions, but t-SNE seems more robust.

Swiss roll data demonstration

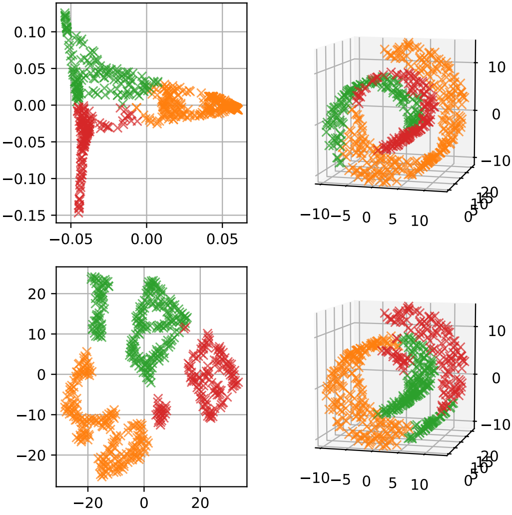


Figure: 2d projection (of 3d data) via (top) LLE and (bottom) t-SNE before K-means. LLE arguably segments better.

Other algorithms (e.g. DBSCAN)

- other algorithms do things differently
- DBSCAN considers clusters as high density regions and gaps are low density regions

  $\rightarrow$ two parameters: radius ϵ of some ball and number of points within said ball to quantify 'density'

  $\rightarrow$ number of clusters identified is a result of the above two choices (unlike in K-means)

  $\rightarrow$ can deal with non-Euclidean distances (you need to provide it though)

- hyper-parameter tuning + cross-validation needed!

Other algorithms (e.g. DBSCAN)

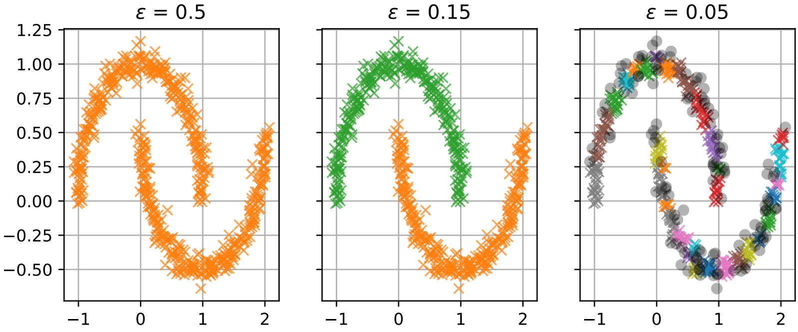


Figure: DBSCAN on moon data at varying ϵ.

- optimal ϵ for the two clusters
- if too small the too many clusters identified

  $\rightarrow$ black points are 'noise' points (no confidence in which cluster it should fall in)

Other algorithms (e.g. DBSCAN)

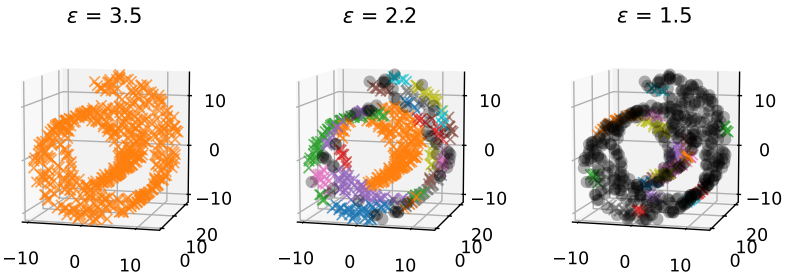


Figure: DBSCAN on swiss roll data at varying ϵ.

- optimal ϵ for the one giant cluster (bit contrived though...)
- smaller ϵ leads to segmenting along the surface, so ok

Demonstration: penguins data

- K-means ($K = 3$) on full 4d data (standardised per feature), then compare classification skill
  $\rightarrow$ some manual remapping needed
  $\rightarrow$ skill is around 91% accuracy for this realisation

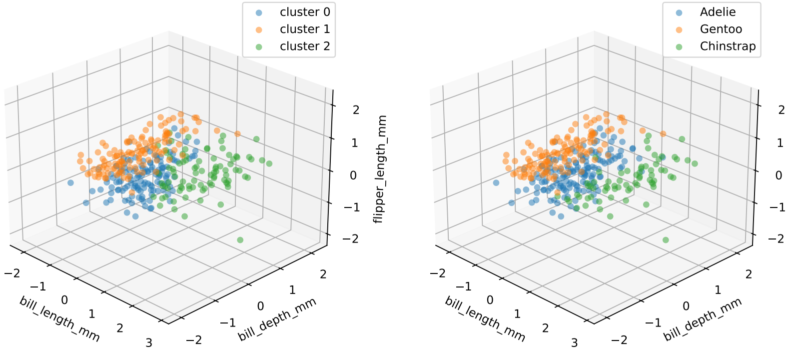


Figure: K-means on standardised penguins data.

Demonstration: penguins data

- as above but with a t-SNE to 2d before K-means
  $\rightarrow$ some manual remapping needed
  $\rightarrow$ skill is around 96% accuracy for this realisation

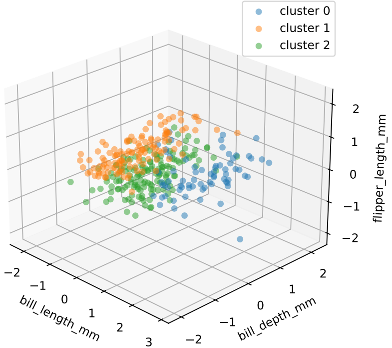


Figure: t-SNE to 2d before K-means on standardised penguins data.

Demonstration

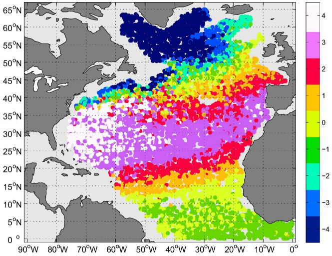


Figure: From Maze *et al*. (2017), Fig. 4. Gaussian Mixture Model to identify watermass clusters from Argo data in Atlantic.

- demonstrating clustering and combinations with dimension reduction techniques

  $\rightarrow$ similar ideas with classification next session
- need to cross-validate and tune hyper-parameters accordingly!

  Moving to a Jupyter notebook $\rightarrow$

assignment: linear models and clustering with Argo data