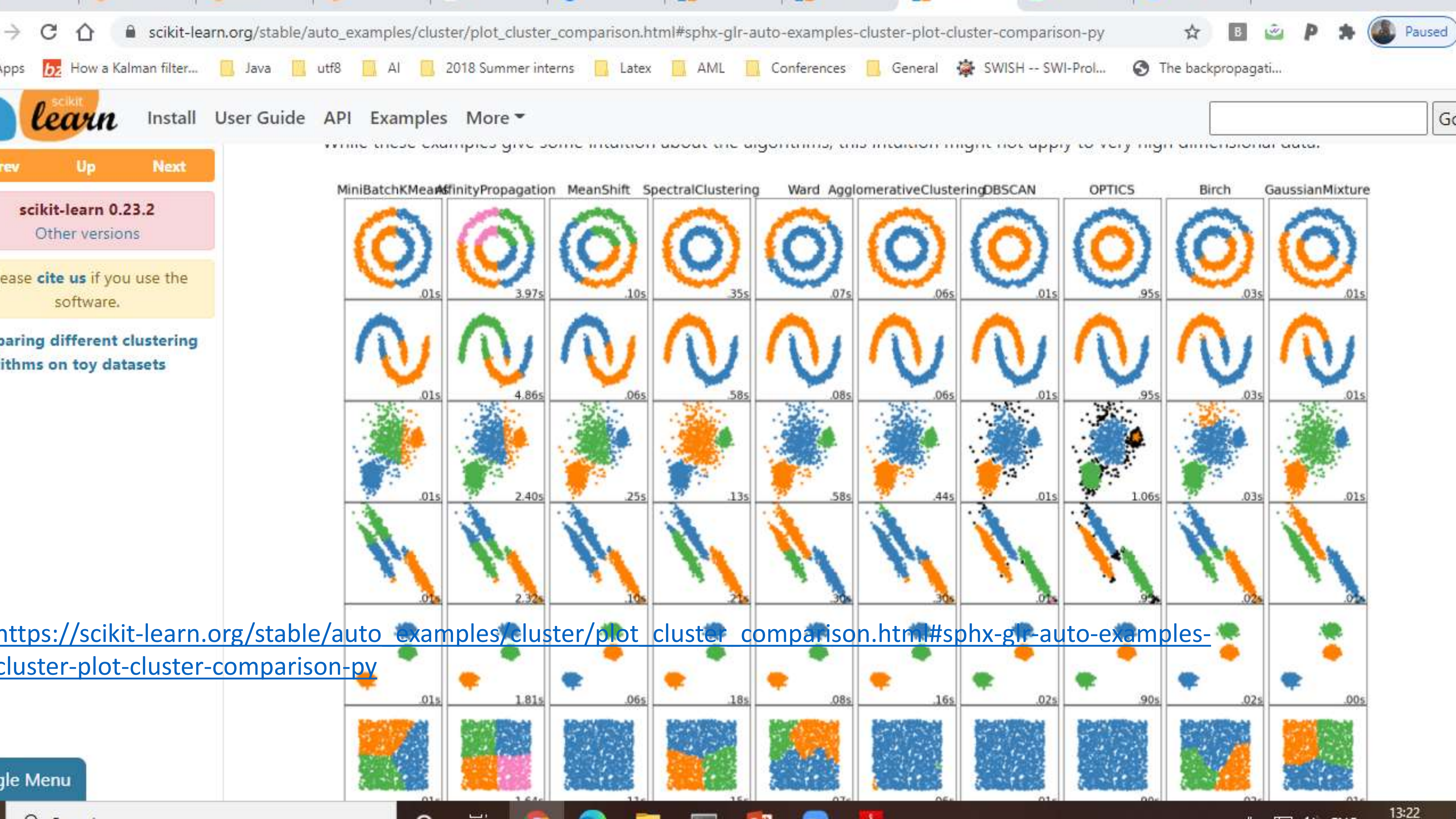
K Means Clustering

Dr. Kalidas Y., IIT Tirupati

By the end of this lecture you would have understood K Means clustering algorithm



scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html#sphx-glr-auto-examples-cluster-plot-cluster-comparison-py

While these examples give some intuition about the algorithms, this intuition might not apply to very high dimensional data.

scikit-learn 0.23.2

Other versions

ease cite us if you use the software.

paring different clustering ithms on toy datasets

https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html#sphx-glr-auto-examples-cluster-plot-cluster-comparison-py

111) key phrase… “Clustering”

- Data has ***Only xi***
- There is ***No yi***

- Birth of clustering
  - You have tons of data…
  - You have limited manpower… to mark yi
  - You have limited time…
  - What you do?

- More of a cost reduction step
  - pre-processing
  - shortlist what data to inspect later
  - ask humans to get yi on those

We want to understand this data

- What is the ***shape of the data spread***?
- Its composed of how many blobs?
- Its Mean
- Its Variance
- Plot it and see
- How do you visualize multi dimensional data
- What are the principal components
- What is the probability of this data set? If we assume some probability distribution!

112) key phrase… “K Means Clustering”

- Input: A set of *m-dimensional data points* $D = \{ x_i \} \ (\forall i \in 1 \ldots N)$
- Process: K-Means clustering algorithm
- Output: K groups of points, $D_1, D_2, \ldots, D_K \ (\forall D_i \subseteq D); D_i \cap D_j = \phi \ (\forall i \neq j)$

- Define: Cluster object
  - Centre (m dimensional point)
  - Member points – A list [x,x…] of data points

- Algorithm sketch :
  - Randomly start with K clusters,
  - assign each point to its closest cluster,
  - refine the cluster centre and
  - repeat until no change

```
KMeans(k, D) //k – number of clusters, D – data set of points (only xi m-dimensional)

STEP 1: Create k cluster objects: Clus[0],…,Clus[k-1]
    STEP 1A: Assign random center: Clus[i].cntr = random(m) // random m-dimensional vector
    STEP 1B: Initialize cluster members to empty, Clus[i].members = []
STEP 2: Assign each point to its closest cluster
        FOREACH i=1..N,
                j* = argmin_j dist(xi,Clus[j].cntr) //find cluster j* to which xi is closest
                Clus[j*].members.append(xi)
STEP 3: Update centroid
        FOREACH j=1..k,
                Clus[j].cntr = centroid(Clus[j].members)
STEP 4:
        If Clus[j].members has not changed since previous iteration, STOP!
        Otherwise, Repeat from STEP 2.
```

K Means as Data Transformer

- Given a new data point, xnew
- Determine euclidean to each of the clusters
- Put all these distance elements and construct a feature vector
- If there are K clusters,
- Then the feature vector is K dimensional

Scope and limitations of K Means

- K Means works best - When data is globular in shape (e.g. like blobs)
- K Means does not work well if data is convoluted in shape (e.g. like concentric circles, moons, maze)
- K Means is very fast
- K Means converges to optimal solution (ref here - https://www.cse.iitb.ac.in/~shivaram/teaching/old/cs344+386-s2017/resources/classnote-2.pdf)