# Clustering

## Supervised vs. Unsupervised

* In unsupervised, we have no labels in our training data.

## Unsupervised Learning

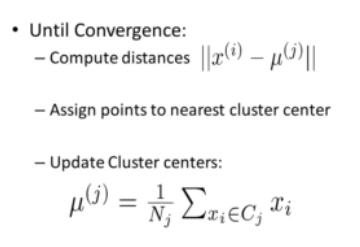
* Find patterns in unlabeled data
* Sometimes used for a supervised setting in which labels are hard to get
* Can identify new patterns that you were not aware of.

### Methods

* K-means
* Mean-shift
* Hierarchical clustering
* Rand index, stability

#### k-means

##### algorithm

* Initialization
  + Choose k random positions
  + Assign cluster centers to these positions
* 

##### Initialization methods

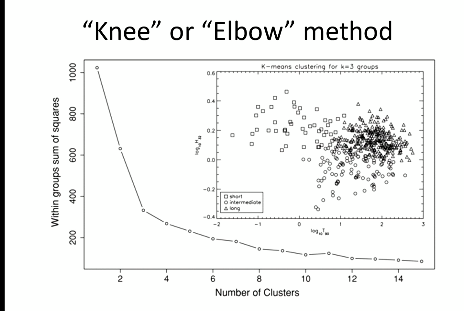
* Random positions
* Random data points as centers
* Random cluster assignment to data point
* Start several times

##### How to find K

* Extreme cases:
  + K=1
  + K=N
* Choose K such that increasing it does not model the data much better.

###### When We Don’t Have Labels?

‘Knee’ or “Elbow” Method

* 

Silhouette Method

Gap Statistic

###### When We Do Have Labels?

Adjusted Rand Index

Mutual Information

V-Measure

Fowlkes-Mallows Index

##### Cross Validation

* Use this if you want to apply your clustering solution to new unseen data
  + Partition data into n folds
  + Cluster on n-1 folds
  + Compute sum of squared distances to centroids for validation set

##### Summary

* Guaranteed to converge
* Result depends on initialization
* Number of clusters is important
* Sensitive to outliers
  + Use median instead of mean for updates
* Having to specify K is annoying

#### Mean Shift

* Put a window around each point
* Compute mean of points in the frame
* Shift the windows to the mean
* Repeat until convergence

##### Summary

* Does not need to know number of clusters
* Can handle arbitrary shaped clusters
* Robust to initialization
* Needs bandwidth parameter (window size)
* Computational expensive

#### Hierarchical Clustering

* It goes from one extreme from each point is it’s own cluster, to where all of them are one cluster
* You must choose the threshold.
* Produces complete structure
* No predefined number of clusters

##### Similarity between clusters

* + Single-linkage
  + Complete-linkage
  + Average linkage

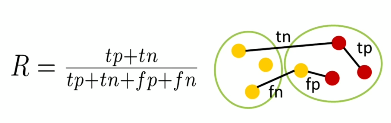
##### Linkage Matters

* Single linkage: tendency to form long chains
* Complete linkage: sensitive to outliers
* Average-link: trying to compromise between the two

## Evaluation Criteria

* Based on expert knowledge
* Debatable for real data
* Hidden unknown structures could be present
* Do we even want to just reproduce known structure?

### Rand Index

* Percentage of correct classifications
* Compare pairs of elements
* 

### Stability

* What is the right number of clusters?
* What makes a good clustering solution?
* Clustering should generalize!
* Use clustering to turn the unsupervised problem into a supervised one
  + Assign labels to the clusters
  + Now you can train the classifier of your choice
* There are different ways of assessing stability
  + Resampling
  + Prediction accuracy

# Unsupervised Learning

## Clustering for Dataset Exploration

* Finds patterns in data
  + Ex: clustering customers by their purchases
  + Compressing the data using purchase patterns (dimension reduction)

### ­Evaluating a clustering

#### Cross tabulation with pandas

#### Inertia measures clustering quality

### Transforming Features for Better Clusterings

#### StandardScaler

* In kmeans: feature variance = feature influence
* StandardScaler transforms each feature to have mean 0 and variance 1

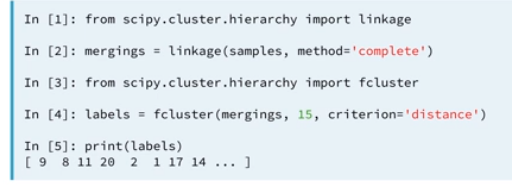
## Visualization with Hierarchical Clustering and T-SNE

### Visualizing Hierarchies

#### Using SciPy

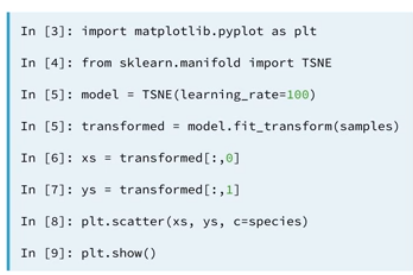
* 

#### Cluster lables in hierarchical clustering



### t-SNE for 2-dimensional Maps

* Maps samples to 2d space or 3d
* Map approximately preserves nearness of samples
* Great for inspecting datasets



#### T-SNE learning rate

* + Choose learning rate for the dataset
  + Wrong choice is when the points bunch together
  + Try values between 50 and 200

#### Different Every time

* Their features are different every time

## Decorrelating your data and dimension reduction

### Dimension Reduction

* More efficient storage and computation
* Remove less-informative ‘noise’ features

### Principal Component Analysis

* Fundamental dimension reduction technique
* First step ‘decorrelation’
* Second step reduces dimension

#### PCA aligns data with axes

* Rotates data samples to be aligned with axes
* Shifts data samples so they have mean 0
* No information is lost

#### PCA Features are not correlated

* After aligning the data with axes
* The resulting PCA features are not linearly correlated (decorrelated)

#### Principal components

* Principal components = directions of variance
* PCA aligns principal components with the axes
* Available as the components\_ attribute of PCA object
* Each row defines displacement from mean

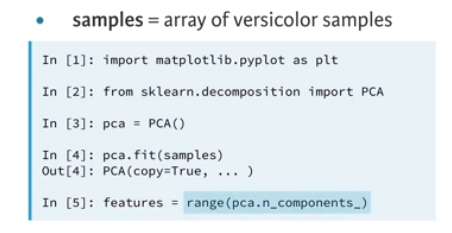
### Intrinsic Dimension

* Intrinsic dimension = number of features needed to approximate the dataset
* Essential idea behind dimension reduction
* What is the most compact representation of the samples?
* Can be detected with PCA

#### PCA identifies intrinsic dimension

* Scatter plots work only if samples have 2 or 3 features
* PCA identifies intrinsic dimension when samples have any number of features
* Intrinsic dimension = number of PCA features with significant variance

#### Plot the variances of PCA features



### Dimension Reduction with PCA

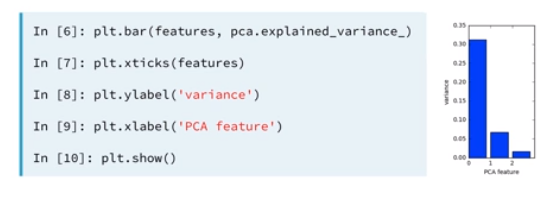
* PCA features are in decreasing order of variance
* Assumes the low variance features are ‘noise’ and high variance features are informative
* Specify how many features to keep
  + PCA(n\_components=2)
  + The intrinsic dimension is a good choice

#### Sparse arrays and csr\_matrix

* Array is sparse when most entires are zero
* Can use scipy.sparse.csr\_matrix instead of NumPy array
  + Csr\_matrix remembers only the non-zero entries

#### TruncatedSVD and csr\_matrix

* Scikit-learn PCA doesn’t support csr\_matrix
* Use scikit-learn TruncatedSVD instead
* Performs the same transformation



## Discovering interpretable features

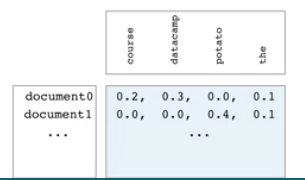
### Non-negative matrix factorization (NMF)

* Another dimension reduction technique
* NMF models are interpretable (unlike PCA)
* It requires that all sample features be non-negative (>=0)

#### Interpretable parts

* NMF expresses images as combinations of patterns
* Works with numpy arrays and csr\_matrix

#### Example word-frequency array

* Word frequency array, 4 words, many documents
* Measure presence of words in each document using ‘tf-idf’
* Tf = frequency of word in document
* Idf = reduces influence of frequent words
* 

#### NMF Components

* It has components just like PCA has principal components
* Dimension of components = dimension of samples
* Entries are nonnegative

#### NMF features

* NMF feature values are non-negative
* Can be used to reconstruct the samples and combine feature values with components

#### Sample Reconstruction

* Multiply components by feature values, and add up
* Can also be expressed as a product of matries

#### Example Uses of NMF

* Word frequencies in each document
* Images encoded as arrays
* Audio spectrograms
* Purchase histories on e-commerce sites

### NMF Learns Interpretable Parts

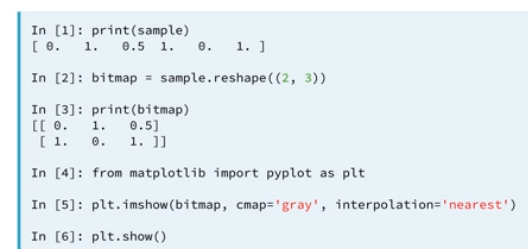
#### NMF Components

* For documents
  + NMF components represent topics
  + NMF features combine topics into documents
* For images, NMF components are parts of images
  + 

#### Encoding a collection of images

* Collection of images of the same size
* Encode as 2d array
* Each row corresponds to a image
* Each column corresponds to a pixel

#### Visualizing samples

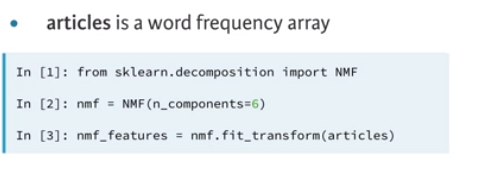


### Building recommender systems using NMF

#### Finding similar articles

* Engineer at a large online newspaper
* Task: recommend articles similar to article being read by customer
* Similar articles should have similar topics

#### Strategy

* Apply NMF to the word-frequency array
  + 
* NMF feature values describe the topics
  + So similar documents have similar NMF feature values
* Compare NMF feature values?
  + Versions of articles
    - Different versions of the same document have same topic proportions
      * Exact feature values may be different
      * One version may use many meaningless words
      * But all versions lie on the same line through the origin
  + Cosine similarity
    - Uses the angle between the lines
    - Higher values means more similar
    - Maximum value is 1, when angle is 0
    - 