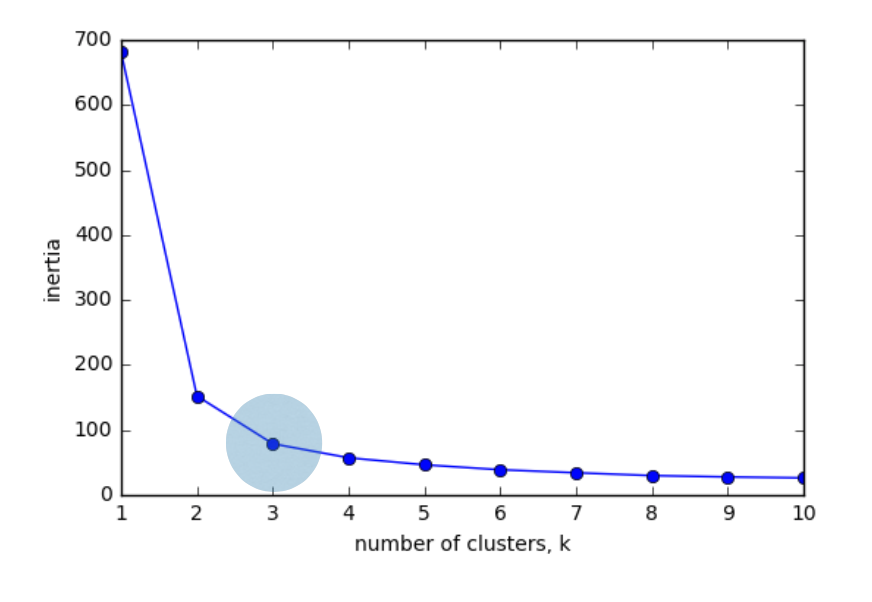
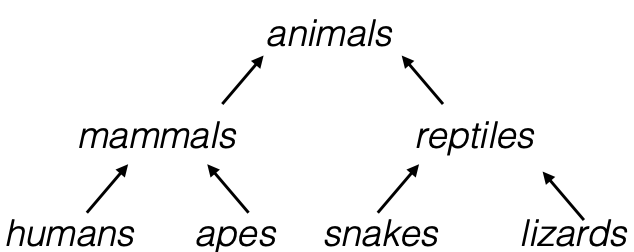
Chapter 1 - Unsupervised Learning

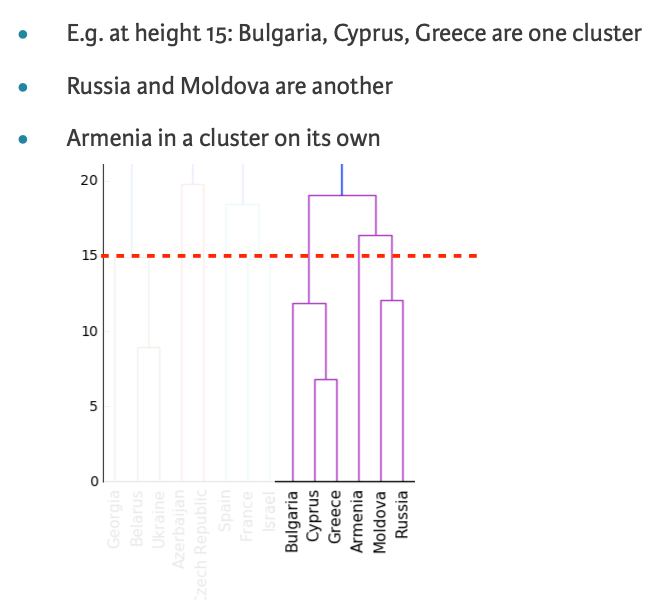
* 1. Clustering for dataset exploration
* Unsupervised learning
  + Finds patterns in data
  + E.g. clustering customers by their practices
  + Compressing the data using purchase patterns (dimension reduction)
* Supervised vs. unsupervised learning
  + Supervised learning finds patterns for a prediction task
  + Unsupervised learning finds patterns in data, without a specific task in mind
* K-means clustering
  + Finds clusters of samples
  + Number of clusters must be specified; implemented in sklearn
* Cluster labels for new samples
  + New samples can be assigned to existing clusters
  + K-means remembers the mean of each cluster (the ‘centroids’)
  + Finds the nearest centroid to each new sample
* Evaluating a clustering
  + Measure quality of a clustering
  + Informs choice of how many clusters to look for
* Cross tabulation with pandas
  + Clusters vs. species is “cross-tabulation” (they refer to the iris data set and the result of “dumb” clustering)
* Measuring clustering quality
  + Using only samples and their clusters
  + Good clustering has tight clusters
  + And samples in each cluster bunched together
* Inertia measures clustering quality
  + Measures how spread out the clusters are (lower is better)
  + Distance from each sample to centroid of its cluster
  + After fit(), available as attribute inertia\_
  + K-means attempts to minimize the inertia when choosing clusters
* The number of clusters – how many to choose?
  + A good clustering has tight clusters (so low inertia) but not too many
  + Choose an “elbow” in the inertia plot
  + When inertia begins to decrease more slowly



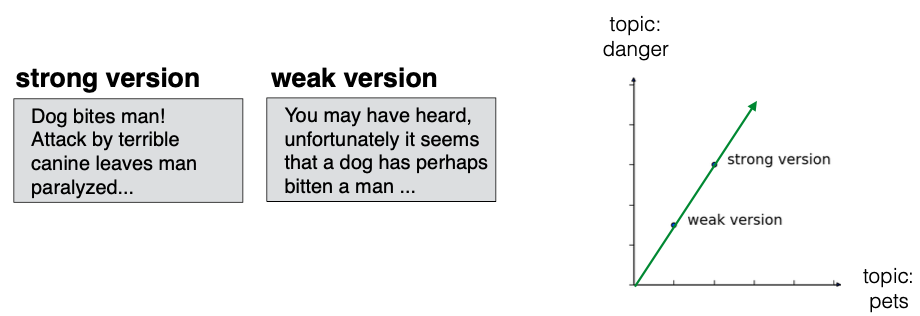
* StandardScalar
  + In kmeans: feature variance = feature influence
  + StandardScalar transforms each feature to have mean 0 and variance 1
  + Features are said to be “standardized”
  + StandardScalar and KMeans have similar methods
    - Use fit()/transform() with StandardScalar
    - Use fit()/predict() with KMeans
* Combining multiple steps with pipelines
  + Use sklearn pipeline to combine multiple steps
  + Data flows from one step into the next
  1. Hierarchical clustering and t-SNE
* Visualizing hierarchies
  + Visualizations communicate insight
    - t-SNE creates a 2D map of a dataset
    - “Hierachical clustering”
  + A hierarchy of groups
    - Groups of living things can form a hierarchy
    - Clusters are contained in one another



* + Hierarchical clustering
    - Every country begins in a separate cluster
    - At each step, the two closest clusters are merged
    - Continue until all countries in a single cluster
    - This is “agglomerative” hierarchical clustering
  + The dendrogram of hierarchical clustering
    - Read from the bottom up
    - Vertical lines represent clusters
* Cluster labels in hierarchical clustering
  + Not only a visualization tool
  + Cluster labels at any intermediate stage can be recovered
  + For use e.g. in cross-tabulations
* Intermediate clustering & height on dendrogram



* Dendrogram show cluster distances
  + Height on dendrogram = distance between merging clusters
  + E.g. clusters with only Cyprus and Greece had distance of approx.. 6 (see depiction)
* Intermediate clustering & height on dendrogram
  + Height on dendrogram specifies max. distance between merging clusters
* Distance between clusters
  + Defined by a “linkage” method
  + Specified via *linkage(samples, method=”complete”)*
  + In “complete” linkage: distance between clusters is max. distance between their samples
  + Different linkage method, different hierarchical clustering
  + Using “fcluster” returns a NumPy array of cluster labels
* t-SNE for 2-dimensional maps
  + t-SNE = “t-distributed stochastic neighbor embedding”
  + Maps samples to 2D space (or 3D)
  + Map approximately preserves nearness of samples
  + Great for inspecting datasets
  1. Decorrelation and dimensionality reduction
* Visualizing the PCA transformation
  + Dimension reduction
    - More efficient storage and computation
    - Remove less-informative “noise” features
    - …which cause problems for prediction tasks, e.g. classification, regression
  + Principal Component Analysis (PCA)
    - Fundamental dimension reduction technique
    - First step: decorrelation
    - Second step: reduces dimension
  + PCA features
    - Rows of transformed correspond to samples
    - Row gives PCA feature values of corresponding sample
  + PCA features are not correlated
    - PCA aligns data with axes
    - Resulting PCA features are not linearly correlated (‘decorrelation’)
  + Pearson correlation
    - Measures linear correlation of features
    - Value between -1 and 1
    - Value of 0 means no linear correlation
  + Principal components
    - “Principal components” = directions of variance
    - PCA aligns principal components with the axes
    - Available as components\_ attribute of PCA object
    - Each row defines displacement from mean
* Intrinsic dimension
  + Is number of features needed to approximate dataset
  + Essential idea behind dimension reduction
  + What is the most compact representation of the samples?
  + Can be detected with PCA
* Intrinsic dimension of a flight path
  + 2 features: longitude and latitude at points along a flight path
  + Dataset appears to be 2-dimensional
  + But can approximate using one feature: displacement along flight path
  + Is intrinsically 1-dimensional
* 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
* Variance and intrinsic dimension
  + Intrinsic dimension is number of PCA features with significant variance
  + In our example: the first two PCA features
* Intrinsic dimension can be ambiguous
  + Intrinsic dimension is an idealization
  + …there is not always one correct answer
* Dimension reduction
  + Represents same data, using less features
  + Important part of machine-learning pipelines
  + Can be performed using PCA
* Dimension reduction with PCA
  + PCA features are in decreasing order of variance
  + Assume the low variance features are “noise”
  + … and high variance features are informative
  + Specify how many features to keep, e.g. PCA(n\_components=2)
  + Keeps the first 2 PCA features
  + Intrinsic dimension is a good choice
  + Discards low variance PCA features
  + Assumes the high variance features are informative
  + Assumption typically holds in practice
* Word frequency arrays
  + Rows represent documents, columns represent words
  + Entries measure presence of each word in each document
  + … measure using “tf-idf”
* Sparse arrays and csr\_matrix
  + Array is sparse: most entries are zero
  + Can use scipy.sparse.csr\_matrix instead of Numpy array
  + Csr\_matrix remembers only the non-zero entries (saves space!)
* TruncatedSVD and csr\_matrix
  + Scikit-learn PCA doesn’t support csr\_matrix
  + Use scikit-learn TruncatedSVD instead
  + Performs same transformation
  1. Discovering interpretable features
* Non-negative matrix factorization (NMF)
  + Dimension reduction technique
  + NMF models are interpretable (unlike PCA)
  + Easy to interpret means easy to explain
  + However, all sample features most be non-negative (>=0)
* Interpretable parts
  + NMF expresses documents as combinatinos of topics (or themes)
  + NMF expresses images as combinations of patterns
* Example word-frequency array
  + Measure presence of words in each document using “tf-idf”
  + ‘tf’ = term frequency, i.e. frequency of word in document
  + “idf” = reduces influence of frequent words
* NMF components
  + NMF has components
  + … just like PCA has principal components
  + Dimension of components = dimension of samples
  + Entries are non-negative
* NMF features
  + NMF feature values are non-negative
  + Can be used to reconstruct the samples
  + … combine feature values with components
* Sample reconstruction
  + Multiply components by feature values, and add up
  + Can also be expressed as a product of matrices
  + This is the “Matrix Factorization” in “NMF”
* NMF fits to non-negative data, only
  + Word frequencies in each document
  + Image encoded as arrays
  + Audio spectrograms
  + Purchase histories on e-commerce sites (tabular data)
* NMF learns interpretable parts
  + Example of applying NMF to articles > uncovers topics in articles
  + Example of applying NMF to grayscale images > uncovers patterns
* Building recommender systems using NMF
  + Finding similar articles example
    - 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
    - E.g. because one version uses 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°