Data Science - Module 5

Machine Learning
Unsupervised Learning

Andrea Cossu – andrea.cossu@sns.it
Scuola Normale Superiore
University of Pisa

Lecture outline

- The importance of unsupervised learning in ML
- Unsupervised models for clustering
  - K-Means
  - DBSCAN
  - Hierarchical clustering
- Dimensionality reduction
  - Principal Component Analysis
- Neural networks
  - Self Organising Maps
  - Autoencoders
  - Bonus track: generative models

Lab Outline

- K-Means
- Principal Component Analysis
- Autoencoder

Lecture outline

- The importance of unsupervised learning in ML
- Unsupervised models for clustering
  - K-Means
  - DBSCAN
  - Hierarchical clustering
- Dimensionality reduction
  - Principal Component Analysis
- Neural networks
  - Self Organising Maps
  - Autoencoders
  - Bonus track: generative models

Problems with supervised approach

- Require large datasets
- Data collection is expensive
- Sustainability issues
- Humans do not (always) require supervisory signal

Towards a more sustainable AI: unsupervised learning

- Recall: supervised learning → (features, target)
- With Deep Learning features are built hierarchically from raw data
  - The expensive part is the target!
- **Unsupervised Learning →** (features)
  - Features can be raw data
- A “blind” data collection process enable unsupervised learning on big data

A compromise: semi-supervised learning

- **Semi-supervised learning** (growing topic in these days)
  - Large amount of unlabeled data
  - Small amount of labeled data
- Combine supervised and unsupervised learning
- Unsupervised alone is not able to go much beyond clustering
- Supervised alone is expensive and strictly oriented to a specific objective
- Expert labeling a small subset of patterns can speed up unsupervised learning

Lecture outline

- The importance of unsupervised learning in ML
- Unsupervised models for clustering
  - K-Means
  - DBSCAN
  - Hierarchical clustering
- Dimensionality reduction
  - Principal Component Analysis
- Neural networks
  - Self Organising Maps
  - Autoencoders
  - Bonus track: generative models

Unsupervised learning for clustering

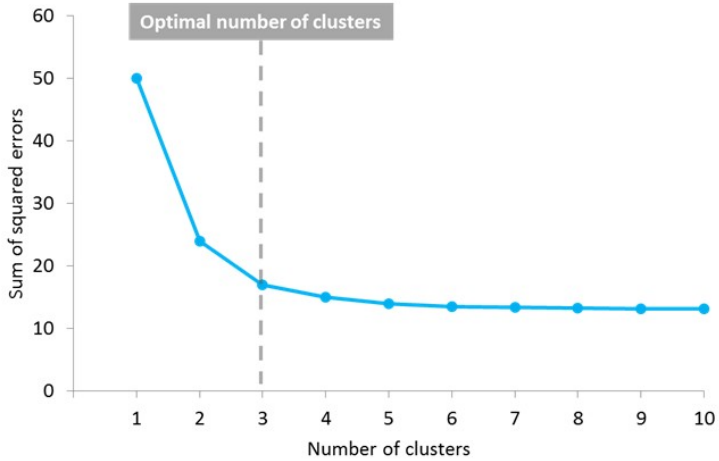
- The main objective of unsupervised learning is clustering
- Before delving deeper into neural unsupervised approaches let’s revise some of the main unsupervised techniques in Data Science

K-means

- Objective: partition data into K clusters (Voronoi diagram again)
  - Data live in a N-dimensional feature space
  - K has to be set by the user
  - Distance-based clustering
- Randomly initialize K centroids (points in N-dimensional feature space)
- Assign each pattern to the closest centroids (e.g. euclidean distance)
- Recompute centroids for each cluster (mean of patterns in that cluster)
- Iterate until convergence (centroids don’t move much)

K-means analysis

- Complexity (big O notation)
  - O(n* K * i * N) → O(#points * #centroids * #iterations * #features)
  - Many improvements have been made in the literature (e.g. automatically select K) bisecting K-means, X-means
- Cluster evaluation
  - Sum of Squared Error (SSE)
  - $dist^2(C, x_c)$ summed over C and x
- How to choose K empirically
  - Elbow rule



https://www.kdnuggets.com/2019/05/guide-k-means-clustering-algorithm.html

K-means analysis

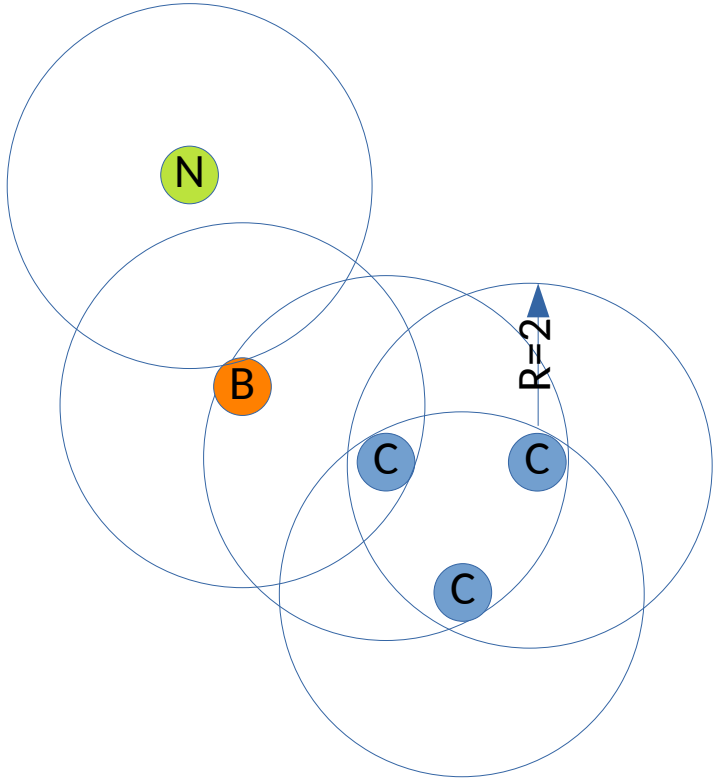
- Increasing K → smaller SSE (and viceversa)
- Overfitting still holds in unsupervised setting
  - What does that mean? What do you have to do with your dataset?
- K-means suffers if optimal cluster shape
  - is not “globular”
  - has varying density
- K-means suffers from outliers…
  - … but you know how to detect and remove them!

DBSCAN

- Density-based clustering
- Parameters:
  - Threshold R on neighborhood radius
  - Minimum number of points in the neighborhood (minpts)
- Three class of points
  - Core points: their R-neighborhood contains at least minpts points (including the point itself)
  - Border points: not core points, but reachable from a core point
  - Noise points (yes, it identifies noise!): the others

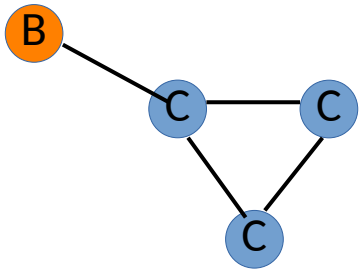
DBSCAN points

Minpts = 3
R = 2



DBSCAN algorithm

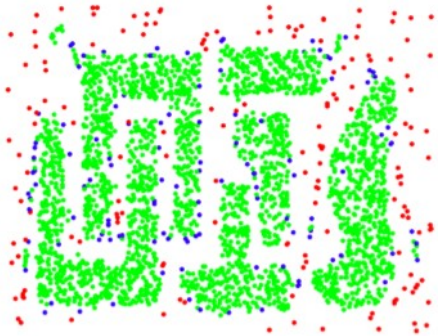
- Classify each point as core, border or noise
- Connect (assign to the same cluster) core points that are neighbors
- Connect border points to one of their core points
- Remove noise points



DBSCAN Analysis

- Complexity: $O(n^2)$ worst case
- Robust to noise
- Address non globular shapes
- Address cluster of different shape and size
- Does not address well differences in cluster density
- Curse of dimensionality!
  - Density with N=100?

Original Points



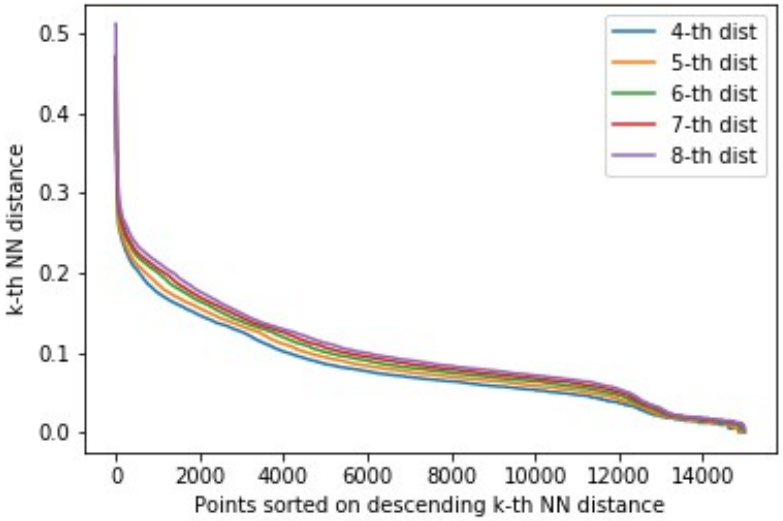
Point types: core, border and outliers

ε = 10, MinPts = 4

https://medium.com/@agarwalvibhor84/lets-cluster-data-points-using-dbscan-278c5459bee5

Choosing R and minpts

- The k^{th} nearest neighbor lies at the same distance for each point
- Noise point → larger distance from the k^{th} point
- Plot ordered distance from the k^{th} point and find the elbow
- Elbow → R
- Minpts ≈ 2*N
  - Larger for large N

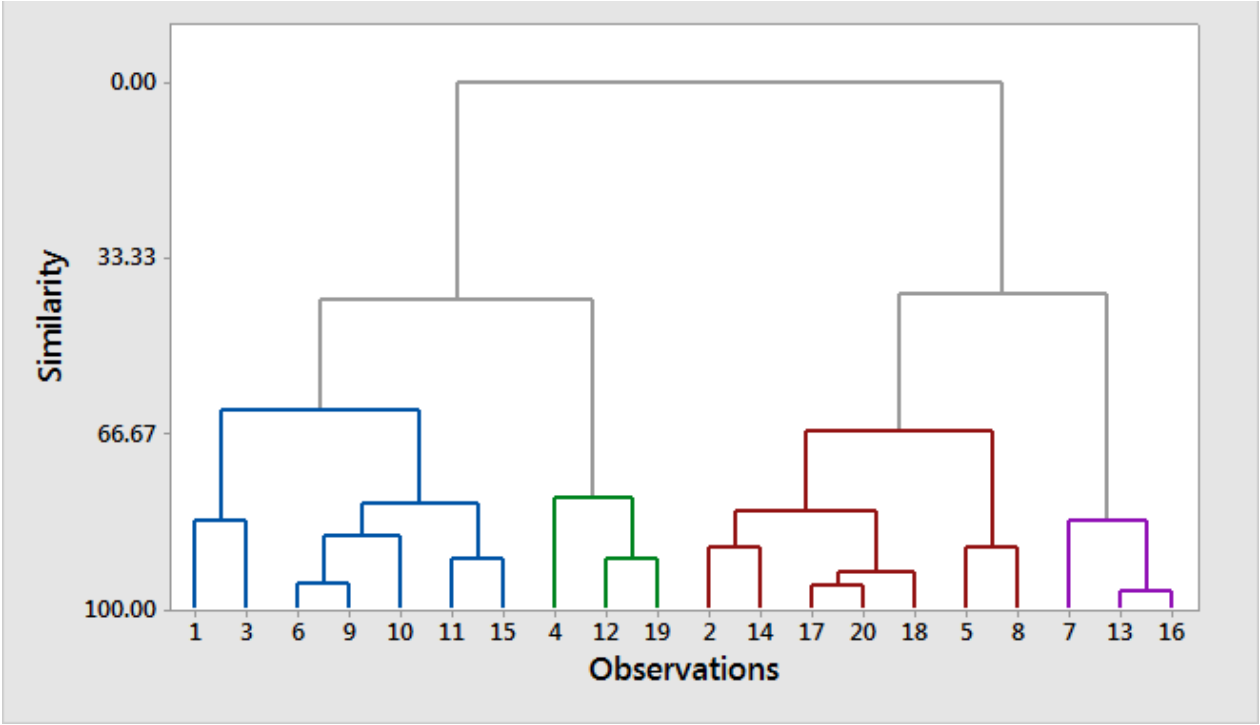


https://i.stack.imgur.com/cTvn5.png

Hierarchical clustering

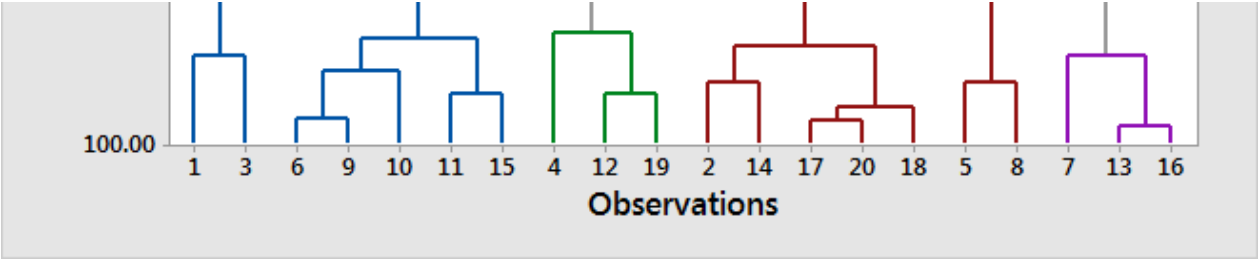
- Let’s understand right away what hierarchical means
- The dendrogram

READ UPWARD



Where are the clusters?

- Cut the dendrogram at custom height!



- The algorithm does not require to know how many clusters you want
  - You have to decide the cut height, though :)
  - You can still decide to set the number of clusters

Build the dendrogram

- Bottom-up (agglomerative)
  - Start with all points as separate points
  - Gather together group of points progressively
  - Stop when obtaining 1 or K clusters
- Top-Down (divisive)
  - Start with all points in one cluster
  - Split iteratively into multiple clusters
  - Stop when each cluster has 1 point or when you have K clusters
- $O(n^3)$ due to similarity/distance matrix computation for each step

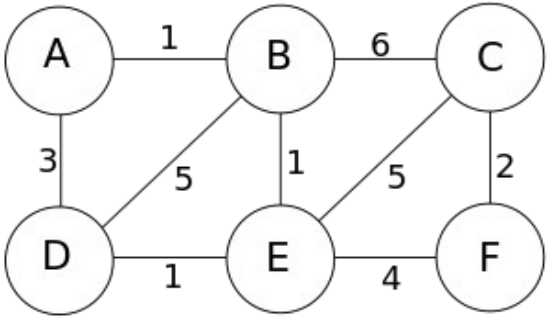
Agglomerative clustering

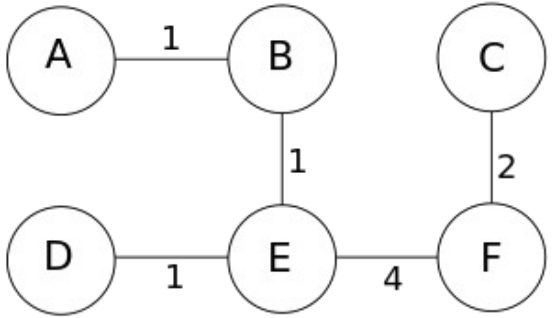
- Compute proximity matrix between all clusters (at first, cluster=point)
- Merge two closest clusters
- Repeat until 1 (or K) clusters

- What is proximity between clusters?
  - MIN (single linkage) → distance between two closest points
    Handle non globular, not robust to noise
  - MAX (complete linkage) → distance between two farthest points
    Robust to noise, does not handle well non globular clusters, break large clusters
  - Group Average → average distance between all pairs of points (scaled by product of cluster sizes)
    Robust to noise, does not handle well non globular clusters

Divisive clustering

- Build a minimum spanning tree on the points
- Break edge corresponding to largest distance
- Repeat until each cluster has 1 point or there are K clusters
- Takes into consideration global information instead of local (as in agglomerative)





https://en.wikipedia.org/wiki/Minimum_spanning_tree

Evaluating clustering

- Sum of Squared Errors
  - Good for evaluating a single clustering choice but also to compare different clustering
  - Keep in mind that more clusters will bring SSE down
- Cohesion + Separation
  - The more a cluster is cohesive and the more it is separated from the others, the better
  - Cohesion → SSE
  - Separation → sum of squared distance between centroids weighted by cluster size

$$\sum_i |C_i|(m - m_i)^2$$

Evaluating clustering

- Silhouette
- Silhouette for each point
  - MINEXT=Minimum distance from points in another cluster
  - AVGINN=average distance from neighbors
  - (MINEXT - AVGINN) / max(MINEXT, AVGINN)
  - Better if closer to 1
- Average for a cluster or for a single clustering result

Lecture outline

- The importance of unsupervised learning in ML
- Unsupervised models for clustering
  - K-Means
  - DBSCAN
  - Hierarchical clustering
- Dimensionality reduction
  - Principal Component Analysis
- Neural networks
  - Self Organising Maps
  - Autoencoders
  - Bonus track: generative models

Why dimensionality reduction

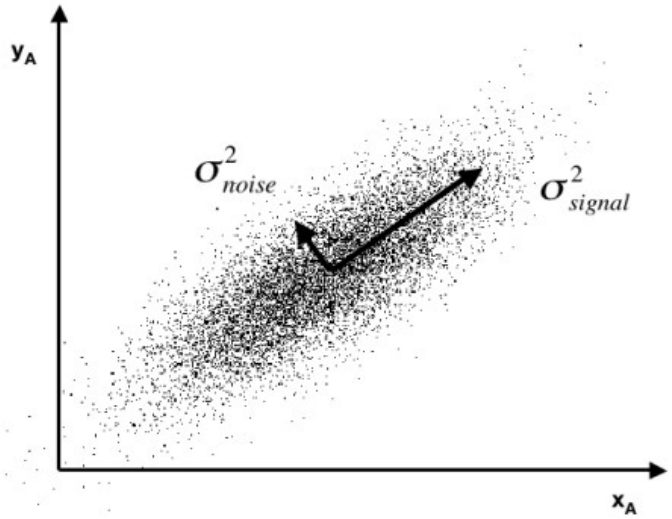
- Manage curse of dimensionality
- Visualization
- Feature selection
- Feature construction
  - It could enhance explainability
- Compression of representation
- Outlier detection

Principal Component Analysis

- It is a linear technique → cfr. Kernel PCA for nonlinear extensions
- It discovers important directions in the feature space
  - Important → capture large portion of variance in data
- Project data in the new, low dimensional, feature space
- Do stuff
- Project back to the original space, if you want
  - With errors proportional to variance not captured by principal components

Math derivation: recap?

- Feature matrix X with patterns on columns, features on rows
- Normalize X by subtracting the mean of each feature (row)
- Covariance matrix $C = XX^T / \#patterns$
  - covariance(a,b)=0 → a,b not correlated
  - covariance(a,b)=variance(a) → a=b
- Compute eigenvectors (and eigenvalues) of C
- Select K eigenvectors with largest eigenvalues
- Matrix P of principal components, one component per row



https://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition_jp.pdf

Principal Components

- Orthonormal basis of the feature space
- Covariance matrix is diagonalized
  - covariance symmetric → spectral theorem → orthonormal basis
- Project on the principal components
  PROJECTED = P X
- Project back on the original space (inverse of orthogonal matrix is its transpose)
  RECONSTRUCTION = P^T PROJECTED
- If P does not contain all components, reconstruction will not be lossless (unless some component is redundant)

PCA and ML

- Used mainly in the preprocessing pipeline
  - To build meaningful features
  - To reduce feature space
- Used also for getting acquainted with the dataset
  - Visualization

Lecture outline

- The importance of unsupervised learning in ML
- Unsupervised models for clustering
  - K-Means
  - DBSCAN
  - Hierarchical clustering
- Dimensionality reduction
  - Principal Component Analysis
- Neural networks
  - Self Organising Maps
  - Autoencoders
  - Bonus track: generative models

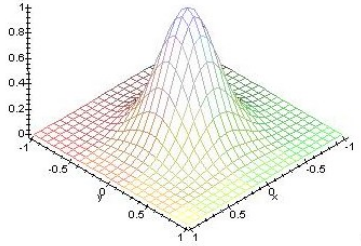
Vector quantization

- Remember Voronoi? Yes, again!
- Vector quantization approach → encode manifold with set of vectors
  - Encode patterns with a fixed set of P vectors of dimensionality N
  - Each pattern is associated to a winning vector (minimum distance or distortion error)
- Vector quantization can be used for clustering
- We can revisit K-means in a vector quantization fashion
  - we won’t, but you can try it out :)

Self Organising Map - SOM (Kohonen Map)

- Adopt vector quantization approach
- N units, each of which is associated to a reference vector *w*
  - Fixed topological structure, most of the times a regular grid of neurons if in 2D
  - Winning unit is the one whose reference vector is closer to input x
- Which units to update?
  - When x is assigned to a unit, a learning process begins which may involve only the winning unit or many units
  - Usually, gaussian distance prior centered on the winning unit weigh the update

http://davis.wpi.edu/~matt/courses/soms/



SOM learning

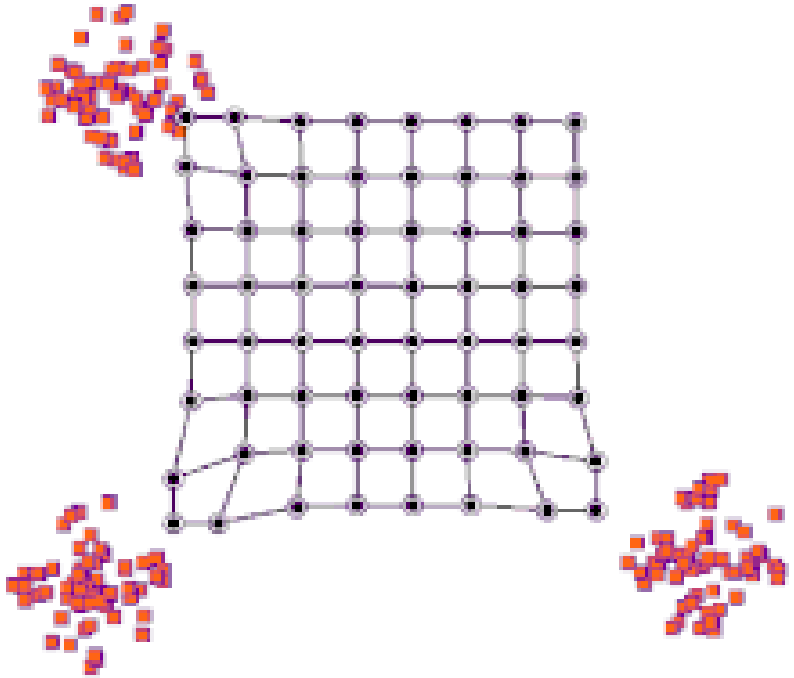
- Nearby units respond to similar inputs and viceversa
- Randomly initialize reference vectors w for each unit
- For each pattern x (Repeat until all w do not change)
  - **Competition:** identify winning unit x (according to some distance function, e.g. Euclidean)

    $\hat{i} = argmin_i \|x - w_i\|$
  - **Cooperation:** soft-max, update all units which are topologically related to the winning one (and the winning one itself) – r is positional index

$$w_i = w_i + \exp\left(\frac{\|r_i - r_{\hat{i}}\|^2}{2\sigma^2}\right)\eta(x - w_i)$$

Decrease sigma and eta as more patterns are observed (small neighborhood and slow update)

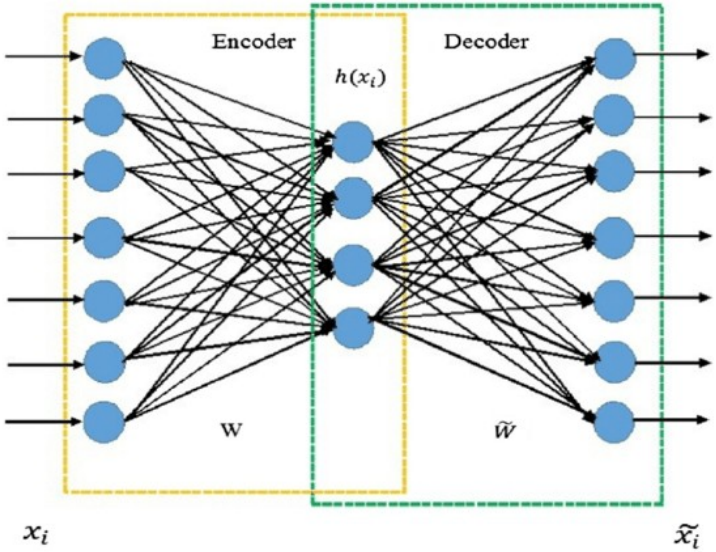
SOM clustering and visualization



https://upload.wikimedia.org/wikipedia/commons/thumb/3/35/TrainSOM.gif/220px-TrainSOM.gif

Autoencoders

- Take input, compute “latent” representation, reconstruct input
- We will focus on the most basic form of autoencoder
  - MLP! What is the latent representation? (hint: latent = hidden)
- Autoencoder
  - Encoder → MLP
    Input x, output h
  - Decoder → MLP
    input h, output o≈x
- Compression
  dimensionality reduction



https://www.researchgate.net/publication/318204554_Intelligent_condition_monitoring_method_for_bearing_faults_from_highly_compressed_measurements_using_sparse_over-complete_features

Autoencoders flavors (there are many!)

- Denoising autoencoders
  - Corrupt input x with some noise z, reconstruct the noise-free input x
- Sparse autoencoders
  - Use a code (hidden layer) larger than input space, but regularize by penalizing large activations (in absolute value)
- Deep autoencoders
  - Stack more layers! → connection to deep learning, first deep model, layerwise pretraining.

Autoencoder learning

- Loss function is reconstruction loss
  - Basic one: MSE between input x and reconstruction (decoder output)
  - Additional penalties are often used (cfr. sparse AE)
- End-to-end: trained with backpropagation and gradient descent
- Hidden layer is the compressed representation of the input
- Used for clustering, visualization (by projecting hidden layer in 2/3D)
- Can be used to *generate* new patterns similar to the ones in training set (Variational Autoencoders)

Bonus track: Generative Adversarial Networks (GANs)

- Learn to generate patterns similar to the ones in the training set
- Generator → generate patterns
  - Input: random noise
  - Output: pattern
- Discriminator → tell apart real patterns from fake ones
  - Input: a pattern
  - Output: prediction of real/fake
- Both are neural networks
- Training set: (patterns) → unsupervised
- Where is the target?
  - Well, we know that real images are only the ones belonging to the dataset
- So, actually, training set: (patterns, real)

ORIGINAL GAN PAPER → http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

GAN learning, an adversarial game

- Discriminator tries to maximize its accuracy
- Generator tries to fool the discriminator
- MinMax game: z noise, x real data

$$min_G max_D L(D,G) = \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log(1 - D(G(z)))]$$

- Bad gradient, better to change to

$$max_G max_D L(D,G) = \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log D(G(z))]$$

- At the end, throw away discriminator and keep generator
- Mode collapse problem → same answer for all z

Stopping GAN Violence: Generative Unadversarial Networks

Samuel Albanie, Sébastien Ehrhardt, João F. Henriques