Overview of Topics

MAT1841 – Fall 2021

Yun William Yu

List of topics

- Random hashing
- Probabilistic streaming and sketching algorithms
- Matrix decompositions (including NMF and probabilistic)
- Random graph theory / percolation theory
- Wavelet bases
- Complexity and entropy
- Nonlinear dimensionality reduction
- Computational topology

Possibly out-of-scope topics (because ML)

These are all super fun topics, but also covered in many other classes. Take e.g. Prof. Papyan's MAT1510 instead: https://sites.google.com/view/mat1510

- Clustering
- Linear classifiers
- Kernel methods
- Deep learning
- Graphical models

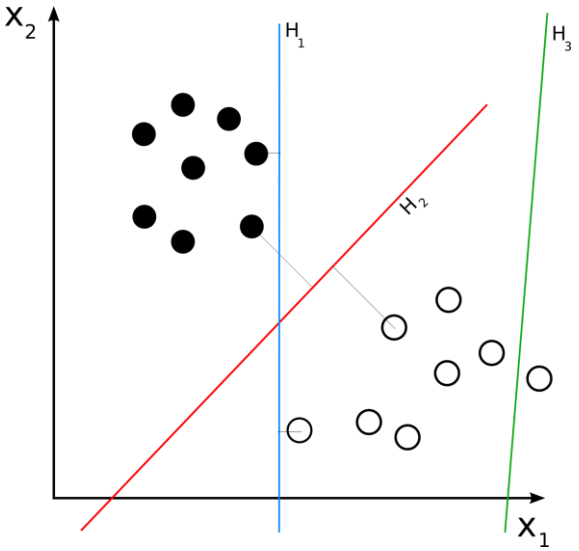
Machine Learning

- Intuition: try to learn the underlying probability distribution generating the data we care about.

- An algorithm builds a mathematical model based on training data, which it uses to make predictions or decisions on new data.
- We say that model parameters are “learned” from the data.
- We focus here on the supervised classification task, though many of the other topics in data science are sometimes “considered” ML.

ML: linear classifier

- Given an input vector $\boldsymbol{x}$, the output $y = f(\boldsymbol{w} \cdot \boldsymbol{x})$, where the weights $\boldsymbol{w}$ are learned from the data should match label l.
- Simple example: $f(a) = 1$ if $a > t$, for some threshold t, and 0 otherwise.
  - Dividing hyperplane, separating classes 0 and 1.



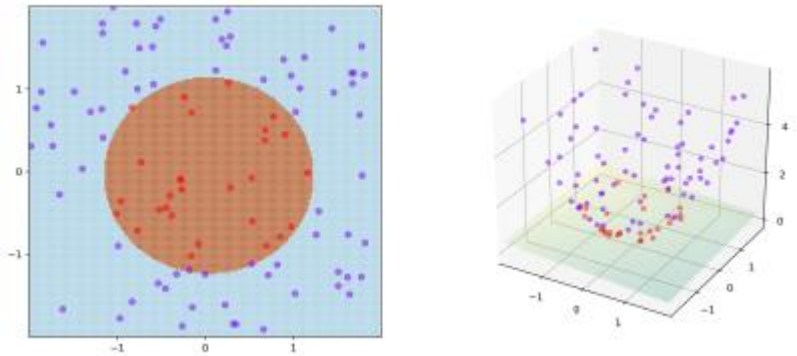
https://en.wikipedia.org/wiki/Linear_classifier

ML: linear classifiers

- Perceptron algorithm
  - Technical modification, $\hat{\boldsymbol{x}} = (\boldsymbol{x}, 1)$, $\hat{\boldsymbol{w}} = (\boldsymbol{w}, -t)$, making separating hyperplanes go through the origin.
  - Initialize with $\boldsymbol{w} \leftarrow 0$.
  - While there exists $\boldsymbol{x_i}$ with $\boldsymbol{x_i} l_i \cdot \boldsymbol{w} \leq 0$, update $\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{x_i} l_i$, where $l_i = \{-1, 1\}$ is class label.
- SVM (Support Vector Machine)
  - Tries to find the maximum-margin hyperplane, not just any hyperplane (like perceptron).

ML: kernel trick

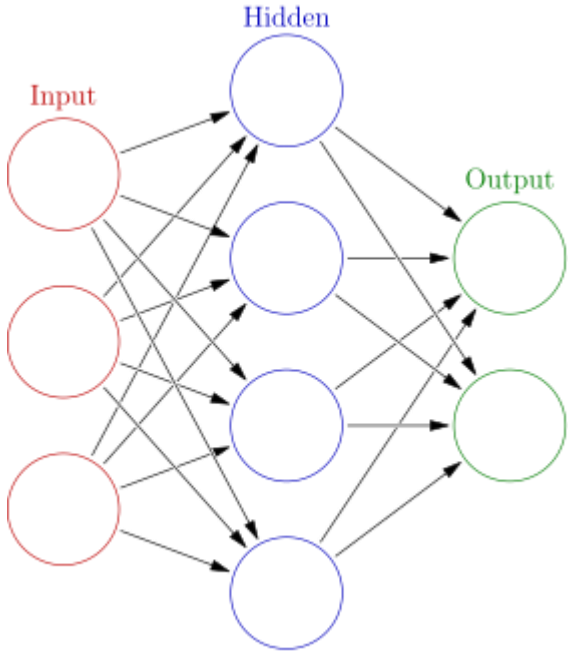
- Data may not be linearly separable
- But we can often map the data to another space where it is linearly separable.
- E.g. $\varphi\big((x_1, x_2)\big) = (x_1, x_2, x_1^2 + x_2^2)$
- Kernel: $K(\boldsymbol{x}, \boldsymbol{y}) = \varphi(\boldsymbol{x}) \cdot \varphi(\boldsymbol{y}) = \boldsymbol{x} \cdot \boldsymbol{y} + |\boldsymbol{x}|^2 |\boldsymbol{y}|^2$
- Careful choice of map allows using kernel function instead of explicit mapping.



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

ML: deep learning

- Chaining together a bunch of simple nonlinear classifiers empirically improves classification.
- Each node represents a linear combination of parent node values, modified by a nonlinearity (often a ReLU).
- Empirically, using a deep network allows us to use a much simpler nonlinearity than more complicated kernel functions.



ML: back-propagation

- The network can be thought of as a function

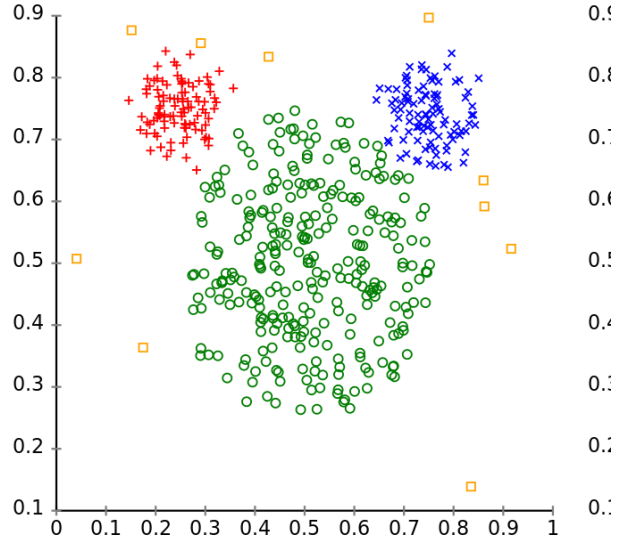
$$g(x) = f^L\left(W^L f^{L-1}(W^{L-1} \cdots f^1(W^1 x) \cdots)\right)$$

where, f^l is the nonlinearity, and W^l is a weights matrix at layer at layer l.

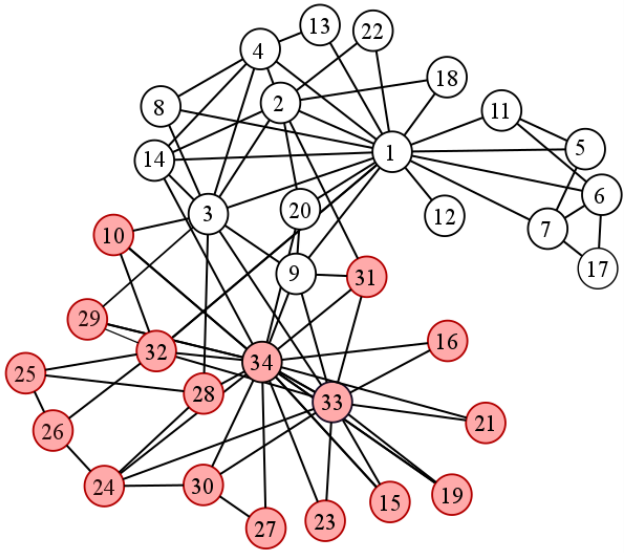
- We also have a loss/cost function $C\left(y_i, g(x_i)\right)$, where y_i is the true label of a data point x_i.
- We want to use gradient descent to optimize the weights based on the training data.
- Each individual component of the gradient $\partial C / \partial w_{jk}^l$ can be computed via the chain rule.
- The back-propagation algorithm avoids duplicate calculations by computing the gradient of each layer from back to front. (i.e. starting from the output layer)

Clustering

- Grouping together data points into “meaningful” groups.
- Also known variously as partitioning, community detection, finding spin glass states, etc.
- Two major versions
  - –High-dimensional space (not just vector spaces)
  - –On a graph
- Hard and soft (depending on group assignment)



https://en.wikipedia.org/wiki/K-means_clustering



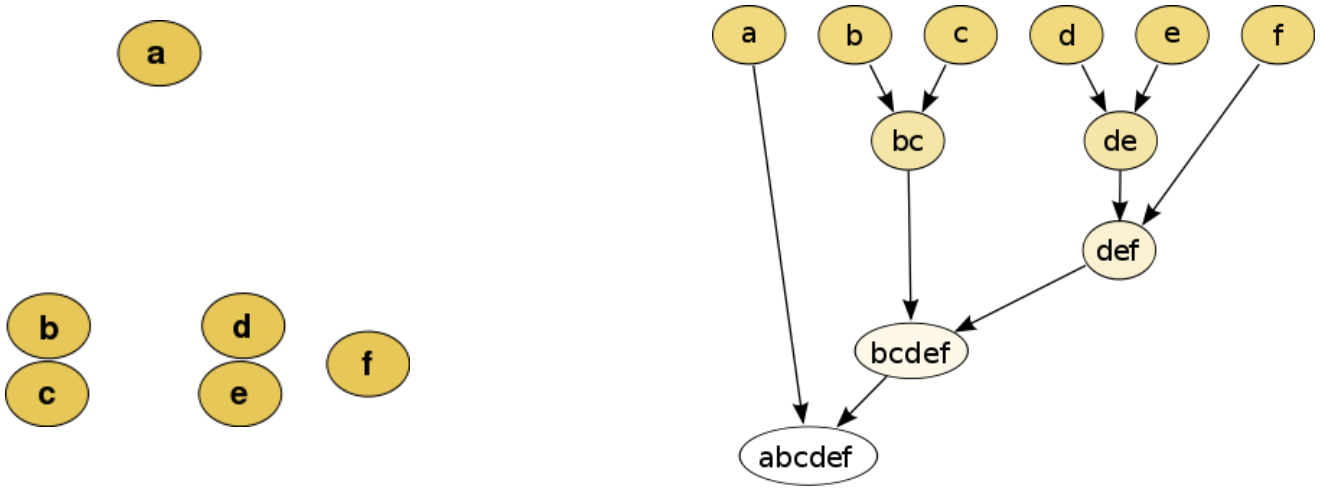
https://commons.wikimedia.org/wiki/File:Zachary%27s_karate_club.png

K-means

- Given a set of observations $(x_1, \ldots, x_n)$, $x_i \in \mathbb{R}^d$, find a partition $\boldsymbol{S} = \{S_1, \ldots, S_k\}$ that minimizes squared distances to cluster centers.
- Naïve k-means algorithm
  - Initialize means (e.g. with random choice)
  - Iterate until convergence:
    - Assign each observation to nearest cluster center
    - Calculate new cluster means based on assignment.
- Converges if using Euclidean distance

Hierarchical clustering

- E.g. on a graph, repeatedly cut the graph in half to minimize the cut weight.
- Alternately, iteratively link together pairs of points that are closest together.



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

Scoring functions

- E.g. Girvan-Newman modularity.
  - The fraction of edges within clusters minus the expected fraction if edges were distributed at random (under several different random graph models).
- Related to Hamiltonian of spin glass in physics. (i.e. energy of a system where adjacent nodes want the same spin).
- Cluster scoring function independent of number of clusters. Often paired with a hierarchical clustering algorithm to allow choosing the correct level.

(Gaussian) mixture models

- Recall we covered a simple Gaussian mixture model where we assumed our dataset was generated by a combination of different radially symmetric Gaussians.
- In general, let $p(\theta) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\mu_i, \Sigma_i)$, where ϕ_i is a weight associated with each multivariate Gaussian distribution $\mathcal{N}(\mu_i, \Sigma_i)$.
- How can we estimate $p(\theta)$ from a bunch of samples drawn from it?

Expectation-maximization iterative algorithm

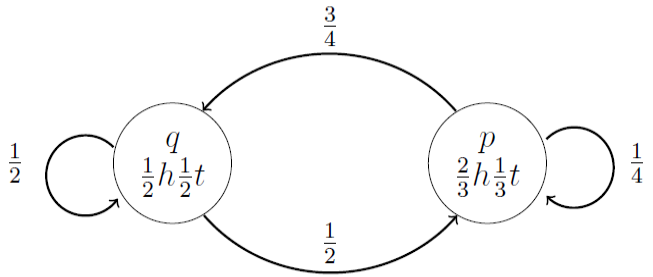
- One commonly used iterative technique to fit parameters θ and missing latent variables Z is the EM-algorithm.
- Algorithm:
  - Initialize parameters θ to random values
  - Compute the probability of each possible value of Z, given θ (E-step).
  - Then, use the just-computed values of Z to compute a better estimate for the parameters θ (M-step)
  - Iterate the last two steps until convergence.

Hidden Markov models (HMM)

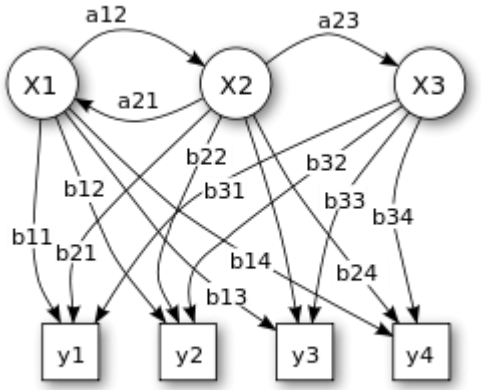
- Let X_n and Y_n be discrete-time stochastic processes and $n \geq 1$. The pair (X_n, Y_n) is a Hidden Markov Model if X_n is a Markov process and not directed observable and $P(Y_n \in A | X_1 = x_1, \ldots, X_n = x_n) = P(Y_n \in A | X_n = x_n)$.
- Generalization of a mixture model where the hidden (latent) variables controlling the mixture component are related through a Markov chain instead of independent.
- System being modelled is assumed to be a Markov process with unobservable (hidden) states.
- Can be learned using a variation of the EM algorithm.

Hidden Markov Models

- Inference tasks:
  - given parameters of a model, compute probability of a particular output sequence.
  - Figure out the distribution over hidden states of the last latent variable at the end of the sequence.



Blum, Hopcroft, Kannan, 2020



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

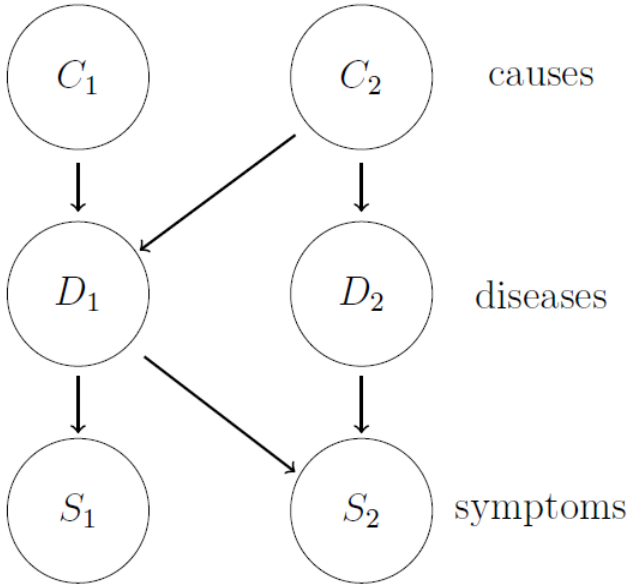
Graphical models

- “Graphical” in the sense of “graph theory”
- A graphical model is a compact representation of a probability distribution over n variables $x_1, \ldots, x_n$.
- When using a directed acyclic graph, is known as a Bayesian or belief network.
- When using an undirected graph, is known as a Markov random field

Bayesian or Belief networks

$$p(x_1, \ldots, x_n) = \Pi_{i=1}^{n} p(x_i | \text{parents of } x_i)$$

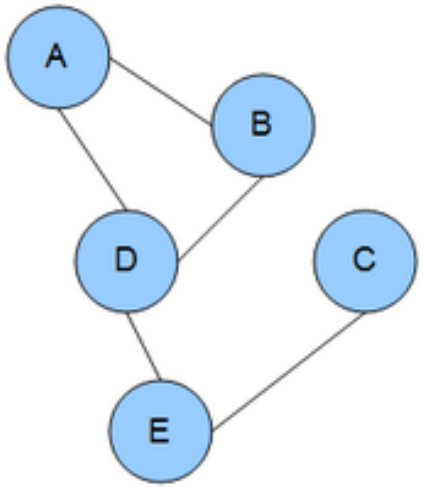
- Each directed edge from y to x represents a conditional probability $p(y|x)$.
- A variable without any in-edges has an unconditional probability distribution.
- We observe only certain variables, known as "evidence".
- E.g. A doctor observes an ill patient's symptoms
  - What disease does the patient have?
  - What is the probability of a specific disease?



Blum, Hopcroft, Kannan, 2020

Markov random field

- Given an undirected graph $G = (V, E)$, a set of random variables $X = (X_v)_{v \in V}$ indexed by V form a Markov random field with respect to G if every variable is conditionally independent of all other variables given its neighbors.



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

Markov random field examples

- Application: Ising model of spin glasses / community detection.
  - Each particle $x_1, \ldots, x_n$ can have a spin ± 1, and the energy of the system is $\exp\left(c \sum_{i \sim j} \left| x_i - x_j \right|\right)$.
  - Minimizing the energy, subject to specified constraints, is a Markov random field.
- Application: Image reconstruction
  - Each pixel is a graph vertex, and we may wish nearby pixels to be similar, with some penalty.

List of topics

- Random hashing
- Probabilistic streaming and sketching algorithms
- Matrix decompositions (including NMF and probabilistic)
- Random graph theory / percolation theory
- Wavelet bases
- Complexity and entropy
- Nonlinear dimensionality reduction
- Computational topology

Random hashing

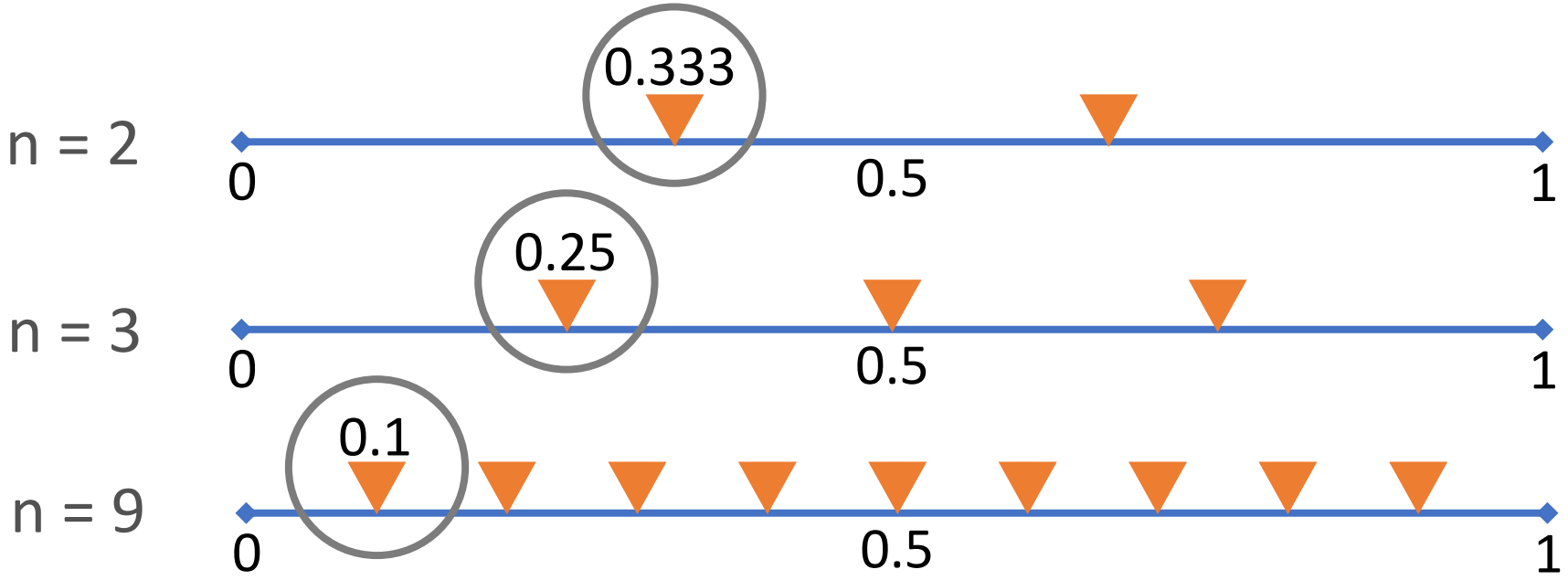
- Real data has all kinds of often unknown probability distributions
- Two approaches to handling large data:
  - Try to understand the distributions it comes from (ML)
  - Force it into a particular distribution that we can reason about (hashing and projections)
- An idealized oracle hash function $f: U \rightarrow [0,1)$ maps each item in U to a uniform random number between 0 and 1.
- Such an oracle would however be expensive to store.
- Can we approximate an idealized hash function with less randomness?

Sketching algorithms

- Sub-linear space algorithms
- Family of algorithms for representing big data as small probabilistic data structures called "sketches"
- Fast accurate estimates of cardinality, quantiles, frequency distributions, set membership, majority element, etc.
- Widely used: routers, databases, search, etc.
- Also used in software for (meta)genome distance approximation, including Mash and Dashing.

Count Distinct Problem

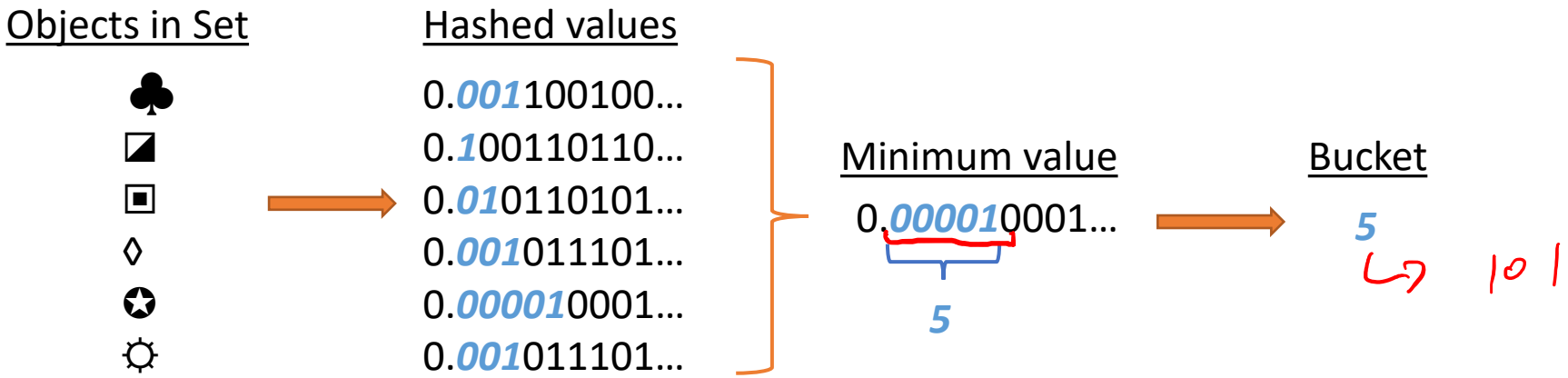
- How many distinct items exist in a list? [Flajolet, Martin, Ziv Bar-Yossef]



- Expected minimum is about $\frac{1}{n+1}$, so we need $O(\log(n))$ bits of storage.

(Hyper)LogLog counting [Flajolet, et al]

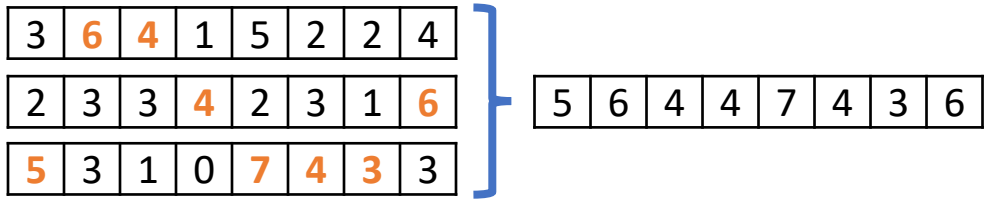
- Only need to store the order of magnitude to get a good estimate, so can compress hashed values.



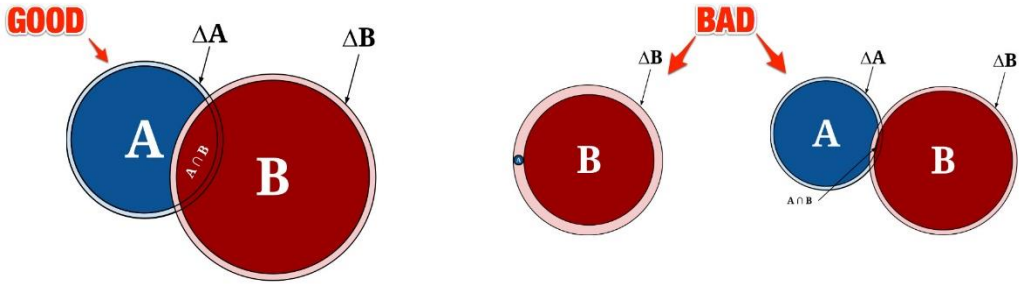
- With some correction terms, get errors that are $O\left(\frac{1}{\sqrt{k}}\right)$, where k is number of buckets / iterations.
- But need only $O(\log\log(n))$ space.

HyperLogLog set operations

- Union cardinality
  - Cardinality of the union of sets is lossless with HLL
  - Determine the largest value for each bucket (iteration)
  - Estimate cardinality using the new sketch



- Intersection cardinality
  - Use inclusion-exclusion principle: $|A \cap B| = |A| + |B| - |A \cup B|$
  - Only accurate if the union and intersection cardinalities are comparable.
  - Complexity grows exponentially with number of sets

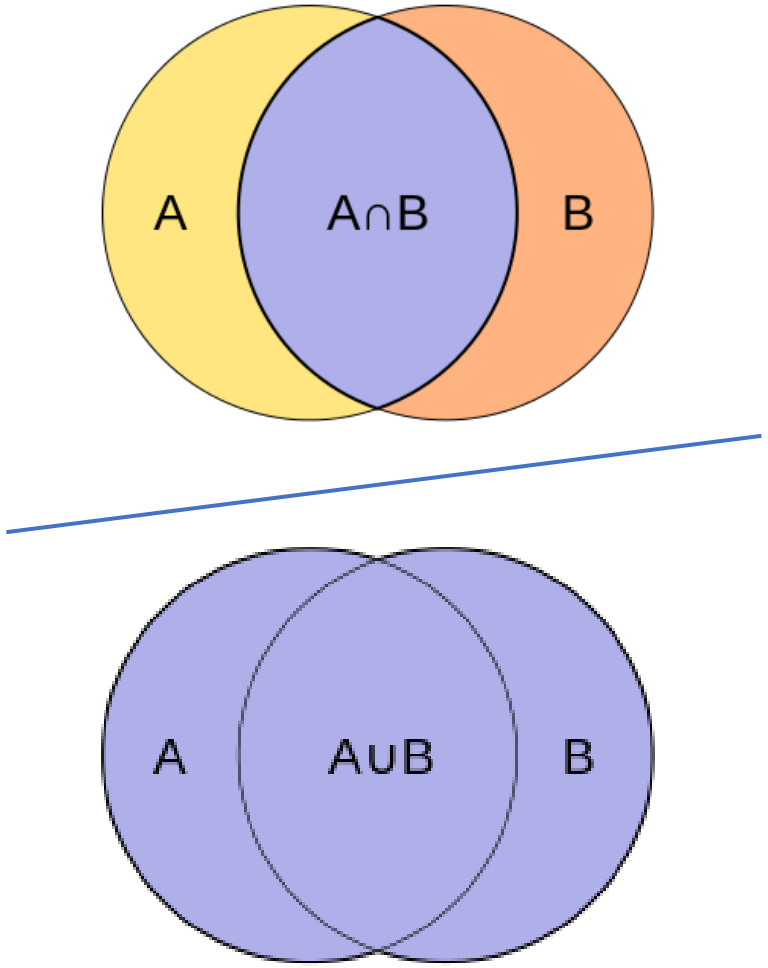


https://research.neustar.biz/2012/12/17/hll-intersections-2/

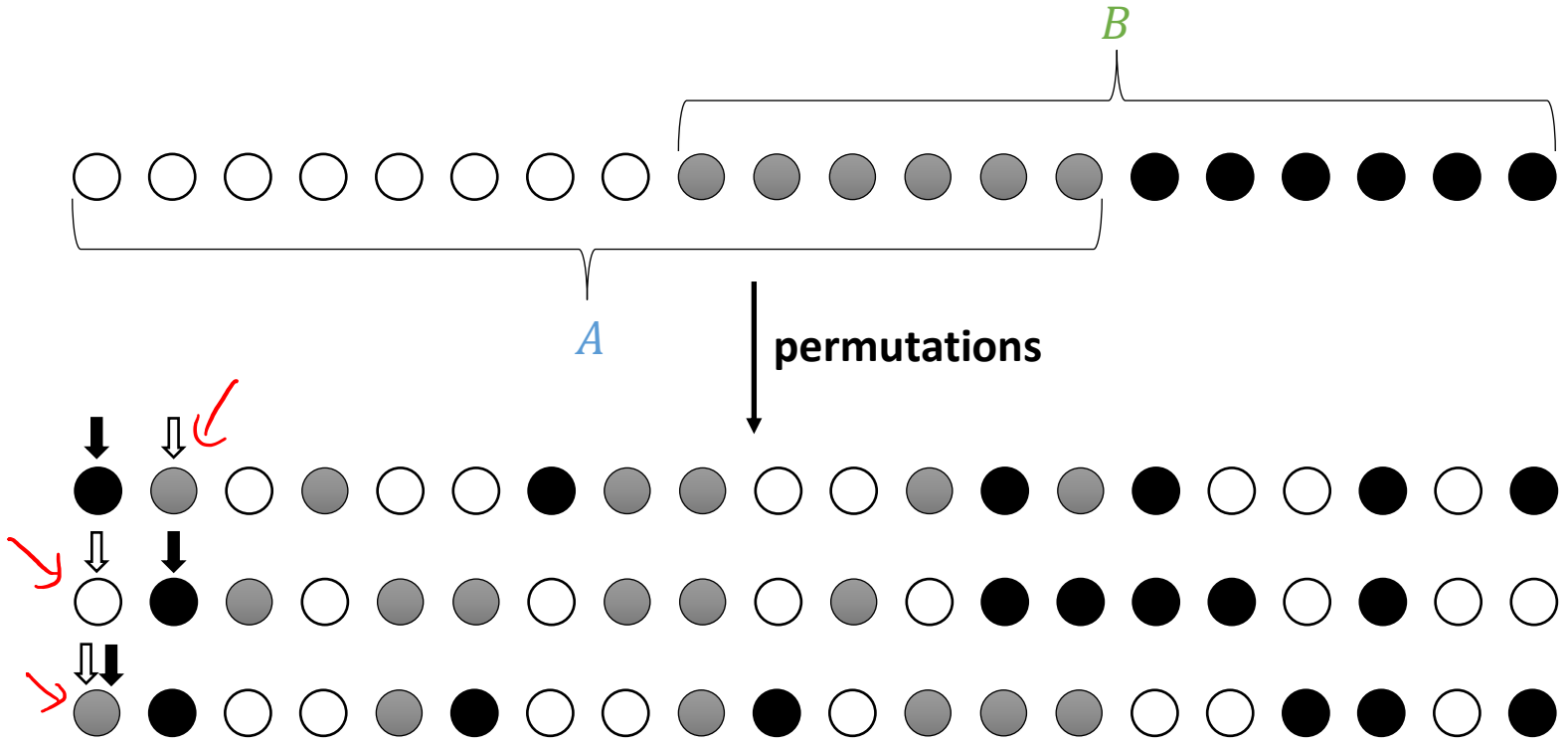
Jaccard index [Jaccard, 1902]

- Measures the similarity between two sets by

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$



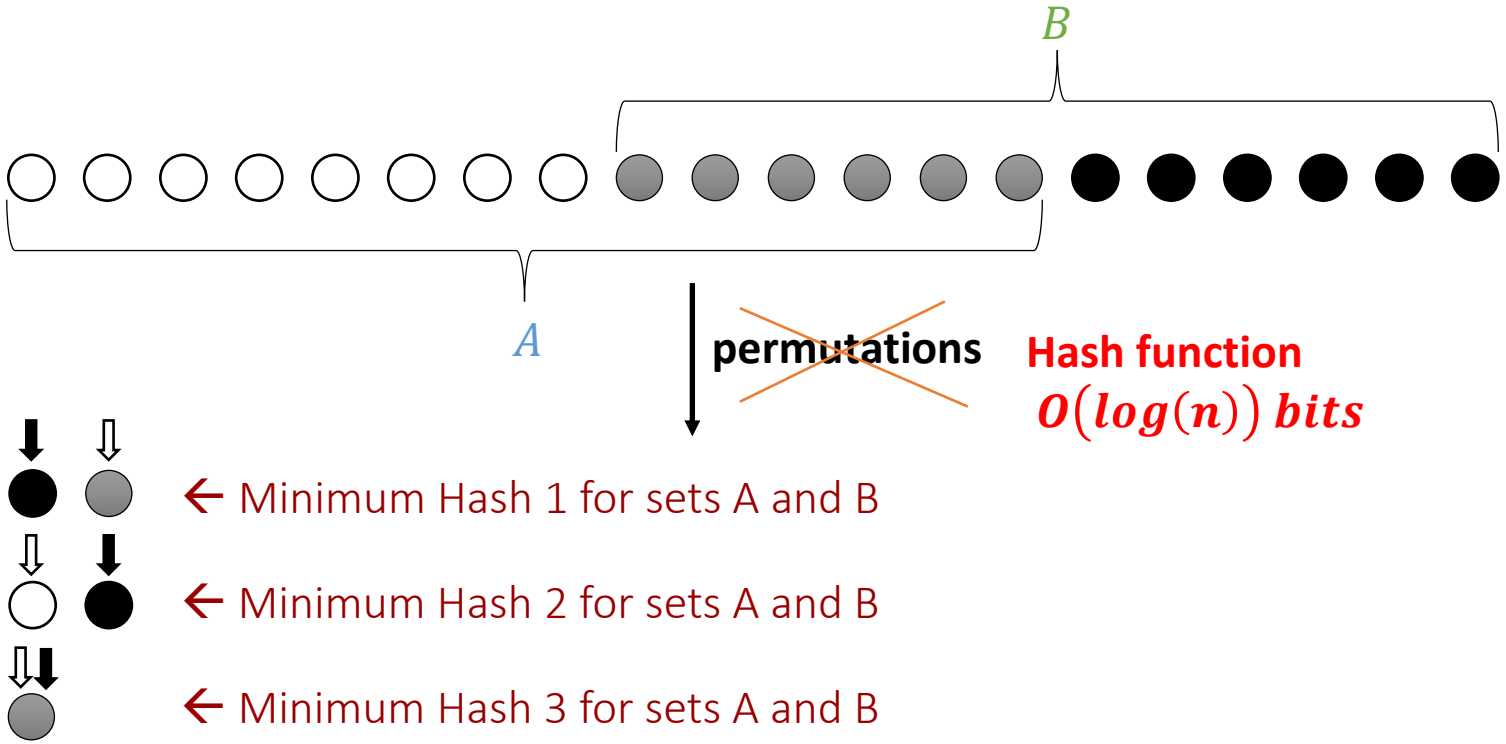
MinHash [Broder, 1997]



$$Prob(\min(A) = \min(B)) = \frac{|A \cap B|}{|A \cup B|} = J(A, B)$$

Can estimate Jaccard index from empirical probabilities!

MinHash [Broder, 1997]



$$Prob(\min(A) = \min(B)) = \frac{|A \cap B|}{|A \cup B|} = J(A, B)$$

Can estimate Jaccard index from empirical probabilities!

MinHash: a worked example

Iterations (buckets)

$\|A\| = 5\mathbf{M}$	$\|A \cap B\|$	$\|B\| = 10\mathbf{M}$
0.1548		0.0358
0.1422		0.0657
0.0559	=	0.0559
0.1287		0.0400
0.0811	=	0.0811
0.1208		0.2649
0.1153		0.0120

$$J(A,B) \approx \frac{2}{7}$$

$\|A\| = 5\mathbf{M}$	$\|A \cup B\|$	$\|B\| = 10\mathbf{M}$
0.1548	0.0358	0.0358
0.1422	0.0657	0.0657
0.0559	0.0559	0.0559
0.1287	0.0400	0.0400
0.0811	0.0811	0.0811
0.1208	0.1208	0.2649
0.1153	0.0120	0.0120

Can merge sketches

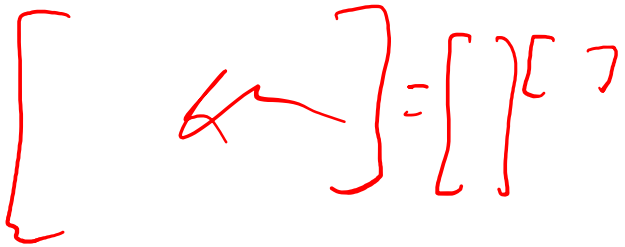
Turnstile streams

- Begin with an n length vector x initialized to 0.
- Stream in a set of updates in the form $\{i, v\}$, where $i \in \{1, \dots, n\}$ and v. For each update set $x_i \leftarrow x_i + v$.
- After streaming, return approximately $f(x)$ for some f. Ideally, for some f, do not need to store all of x. Examples:
  - p-norm
  - Largest entries (heavy hitters)

AMS Sketch: $F_2 = |x|_2^2$

- Let $\Pi = \frac{1}{\sqrt{m}} \begin{bmatrix} \pm 1 & \cdots & \pm 1 \\ \vdots & \ddots & \vdots \\ \pm 1 & \cdots & \pm 1 \end{bmatrix} \in \mathbb{R}^{m \times n}$, where each row is chosen pseudorandomly by a 4-wise independent hash function, so the matrix can be represented in $O(m \log n)$ bits.
- Store $y \leftarrow y + v\Pi_{\mathrm{i}}$ where Π_i is the ith column.
- Then $|y|_2^2$ is an estimator of $|x|_2^2$.
- Requires $O\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta}\right)$ space to estimate with $1 - \delta$ probability to within ϵ relative error.

Nonnegative matrix factorization



- Consider the Topic Modelling problem
  - Suppose there exist r topics and and n documents, which are a mixture of the topics, determining the probability distribution of words (or phrases) in the document.
  - We want to determine both the topics that exist, as well as what topics the documents are mixtures of.

$$\begin{array}{c} \\ \text{TERM} \end{array} \overset{\text{DOCUMENT}}{\begin{pmatrix} \\ P \\ \\ \end{pmatrix}} = \begin{array}{c} \\ \text{TERM} \end{array} \overset{\text{TOPIC}}{\begin{pmatrix} \\ B \\ \\ \end{pmatrix}} \begin{array}{c} \text{TOPIC} \end{array} \overset{\text{DOCUMENT}}{\begin{pmatrix} \\ C \\ \\ \end{pmatrix}}$$

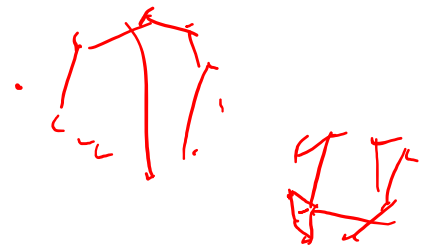
NMF (continued)

- Cannot use SVD for topic modelling because some of the low-rank "topics" will have negative numbers of particular words.
- Hence, we need non-negative matrix factorization
- i.e. decompose a matrix $A = BC$, where all entries are non-negative, and the columns of B and C sum to 1.
- Algorithms for NMF are much more complicated than SVD, but can be done in $O\big(poly(r)\big)$ time, where r is the rank of the matrix.

Random graphs

- Networks of connected nodes show up often
  - Electrical grids
  - Social networks
  - Protein interaction networks
- Real-world networks can be analyzed using things like graph partitioning, or random walks on a particular network. However, we may also be interested in modelling the generation of the network itself.

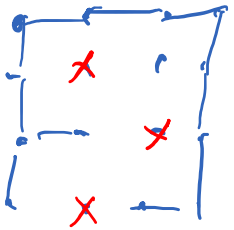
Random graphs – Erdos-Renyi



- $G(n,p)$ model, where n is the number of vertices, and p is the edge probability.
- Degree is tightly concentrated around np, and in fact binomial with mean np.
- Has sudden phase transition in number of connected components at expected degree $d = 1$.
- Degree of separation also has a sharp threshold.
- Has related applications in designing CNF solvers for SAT problems.

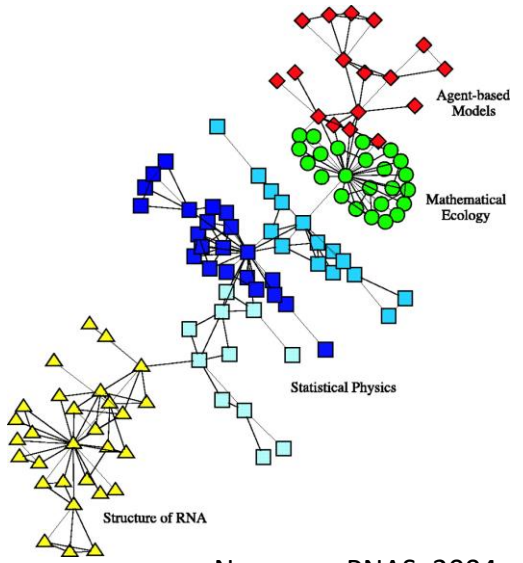
Percolation Theory and random graphs

- Suppose you have an alloy of an electrical conductor and an insulator, with the atoms intermingled in the form of a cable.
- So long as you have a path from one end of the cable to the other composed entirely of conductive atoms, the cable itself is conductive.
- How much insulator can you put into the cable while it remains conductive?

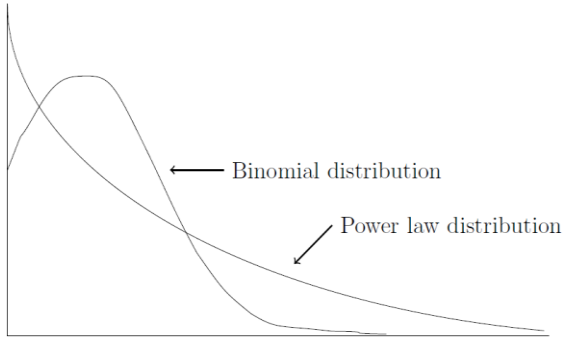


Random graphs – Preferential attachment

- Real social networks however do not look like Erdos-Renyi graphs.
- One easy way to see this is to look at the degree distribution.
- Preferential attachment (rich get richer) is one common model that promotes both small-world graphs and the long-tail behavior of real social networks.



Newman, PNAS, 2004



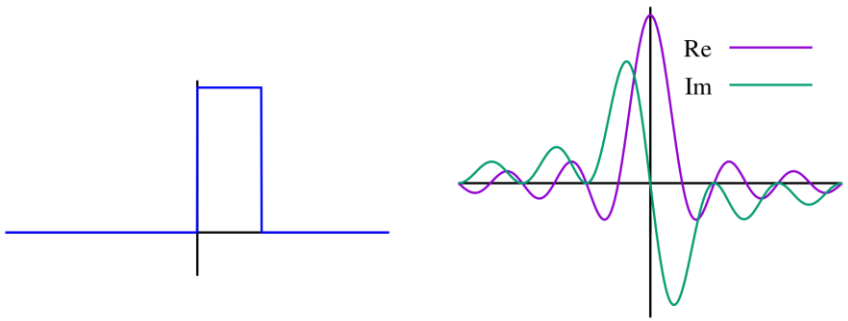
Blum, Hopcroft, Kannan, 2020

Random graphs – Conf-model

- But what about networks whose properties we have trouble approximating using a simple model?
- Can we still generate a random network with e.g. the same degree distribution as a real one?
- One way to do this is the configuration random model. Start with an existing network (or set of degree distributions), cut each edge in half, and then randomly reattach edge-halves.
- Used in the null model for Girvan-Newman modularity clustering score.

Wavelets: background (FT)

- Recall the Fourier transform, which gives a basis in terms of sines and cosines for the space of functions.
  - Each of the basis functions contains information localized in frequency, but not in space/time.
  - Hard to represent discontinuities.



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

Wavelets: motivation

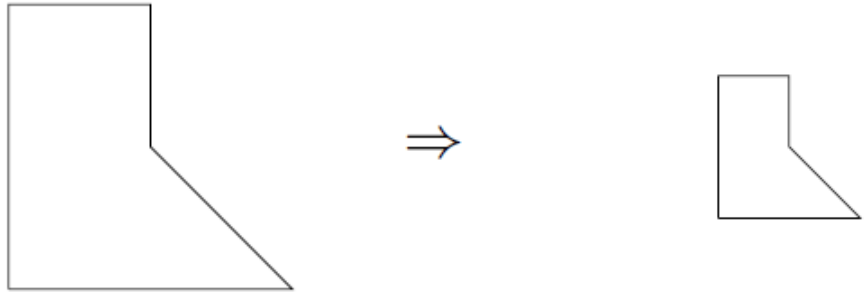
- Want an easy-to-compute-with orthogonal basis set of functions that have finite support.
- Finite support makes it easier to represent functions that have discontinuities.
- The basis should be composed of simple pieces, like sines and cosines for the Fourier transform.

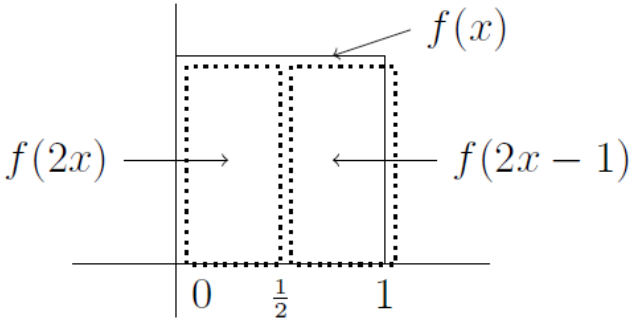
Wavelets: dilations

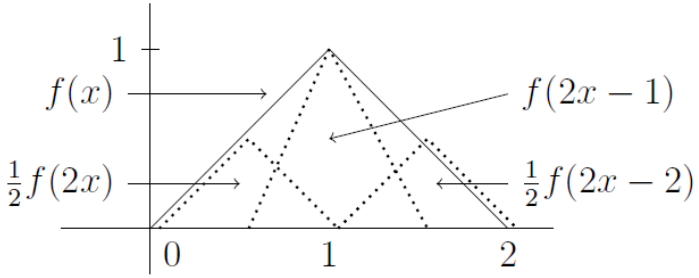
- Dilations are mappings that scale all distances by the same factor.
- A dilation equation is a function defined in terms of linear, scaled, shifted versions of itself.

$$f(x) = f(2x) + f(2x-1)$$

$$f(x) = \frac{1}{2}f(2x) + f(2x-1) + \frac{1}{2}f(2x-2)$$







Blum, Hopcroft, Kannan, 2020

Wavelets: construction

- Start from a dilation equation, and a solution $\phi(x)$
- We define a 2D set of scaling functions

$$\phi_{jk}(x) = \phi(2^j x - k)$$

- For a fixed value of j, the ϕ_{jk} span a space V_j.
- If $\phi(x)$ satisfies a dilation equation of the form

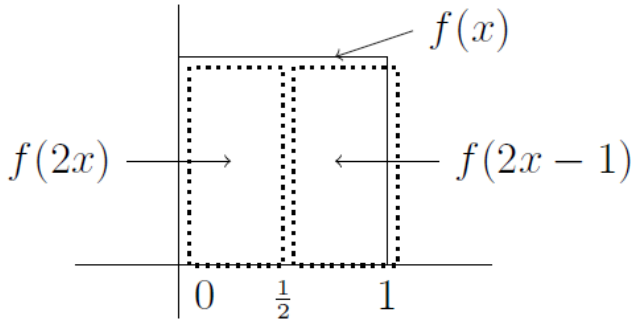
$$\phi(x) = \sum_{k=0}^{d-1} c_k \phi(2x - k)$$

Then each ϕ_{jk} is a linear combination of $\phi_{j+1,k}$'s.

- Thus $V_0 \subseteq V_1 \subseteq V_2 \subseteq \cdots$
- We can then approximate a function by choosing V_k

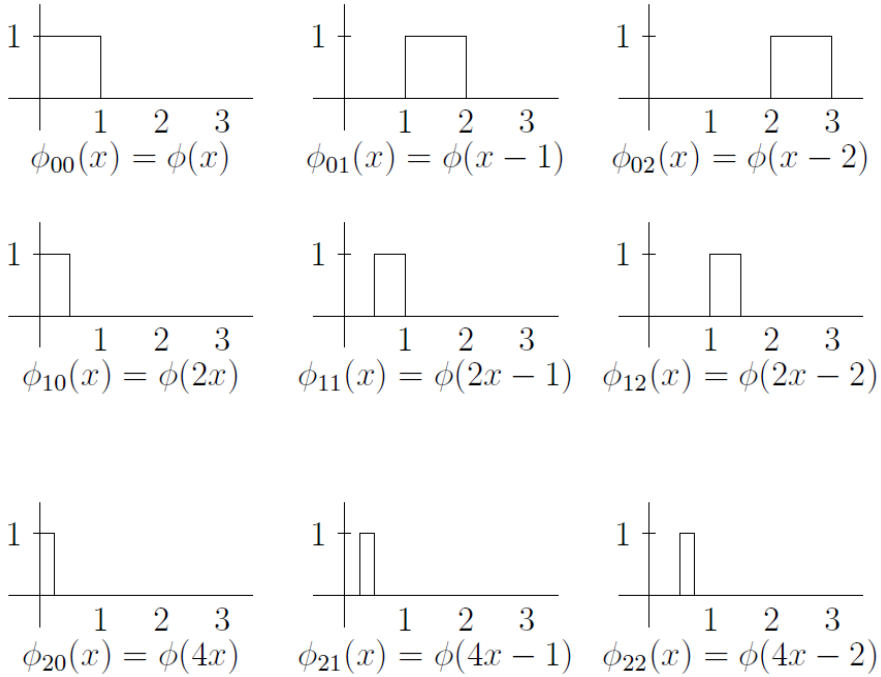
Wavelets: Haar wavelet

$$f(x) = f(2x) + f(2x-1)$$



$$\phi(x) = 1 \text{ if } x \in [0,1]$$

$$\phi_{jk}(x) = \phi(2^j x - k)$$



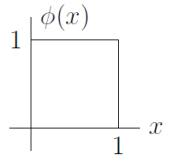
Blum, Hopcroft, Kannan, 2020

Wavelets: Haar wavelet

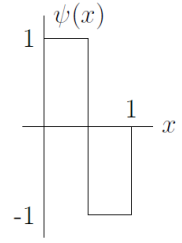
- But the set of functions given above is not orthogonal, so reduce set to a linearly ind. set.
- The Haar wavelet is defined by the following basic functions, but with certain members that are linearly dependent removed.

The Haar Wavelet

$$\phi(x) = \begin{cases} 1 & 0 \le x < 1 \\ 0 & \text{otherwise} \end{cases}$$



$$\psi(x) = \begin{cases} 1 & 0 \le x < \frac{1}{2} \\ -1 & \frac{1}{2} \le x < 1 \\ 0 & \text{otherwise} \end{cases}$$



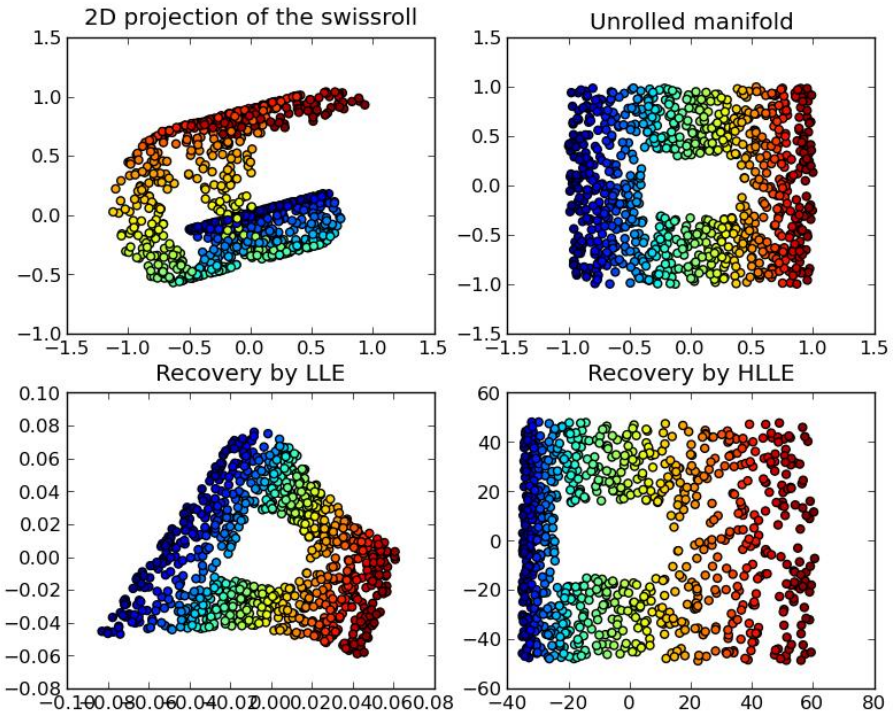
Blum, Hopcroft, Kannan, 2020

Wavelets: applications

- Applications
  - Data compression
  - Signal processing
  - Power-line communication protocols
- Issues
  - Often need to design a wavelet system specific to the problem. i.e. Haar is often not the most natural.
  - Smoothness of the basis functions can sometimes be desirable.

Nonlinear dimensionality reduction

- SVD and related methods like PCA allows you to reduce the dimensionality of a dataset down linearly.
- What if your dataset is actually nonlinear?
- What techniques do we have to reduce dimensionality while preserving structure?
  - E.g. t-SNE and UMAP, among others.



https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction#/media/File:Lle_hlle_swissroll.png

Persistent homology

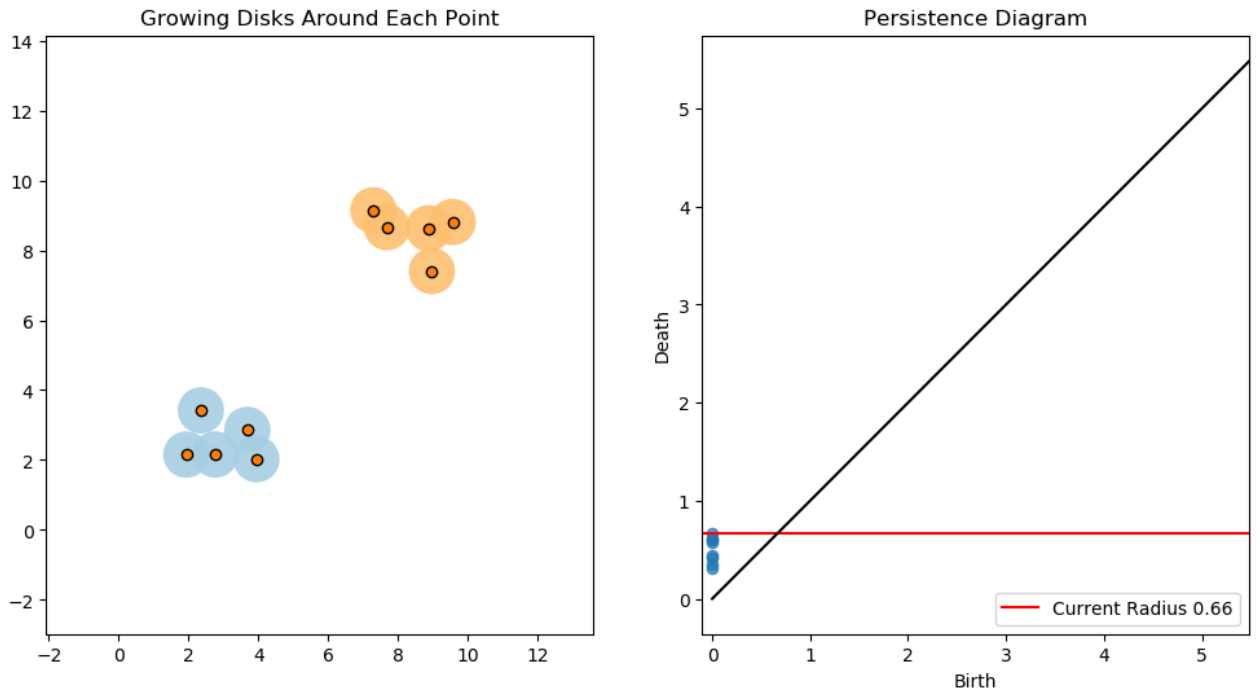
- Method for computing topological features of a space at different spatial resolutions.
- Represent a data cloud as a simplicial complex.
- A distance function specifying links between neighboring points corresponds to a filtration on the simplicial complex.
- We can then ask questions about the simplicial homology at a particular resolution.
- Persistent homologies are the long-lived features.

Persistent homology: math

- Let S be a simplicial complex.
- A simplicial k-chain: $\sum_{i=1}^{N} c_i \, \sigma_i$ where $c_i \in \mathbb{Z}$ and σ_i is an oriented k-simplex (and $-\sigma_i$ is the opposite oriented simplex).
- The free abelian group of k-chains on S is written C_k, and has basis in 1-1 correspondence with k-simplices.
- The boundary operator $d_k \colon C_k \to C_{k-1}$ is a homomorphism given by $d_k(\sigma) = \sum_{i=1}^{k} (-1)^i (\sigma_{\hat{\imath}})$, where $\sigma_{\hat{\imath}}$ is the ith face of σ, obtained by deleting its ith vertex.
- Let $Z_k = \ker \delta_k$, the subgroup of cycles.
- Let $B_k = \operatorname{im} \delta_{k+1}$, the subgroup of boundaries.
- The kth homology group is defined as the quotient abelian group $H_k(S) = Z_k / B_k$, which is nonzero when there are k-cycles on S which are not boundaries. (i.e. k-dim holes in the complex)
- The kth Betti number of S is $\beta_k = \operatorname{rank}\big(H_k(S)\big)$.

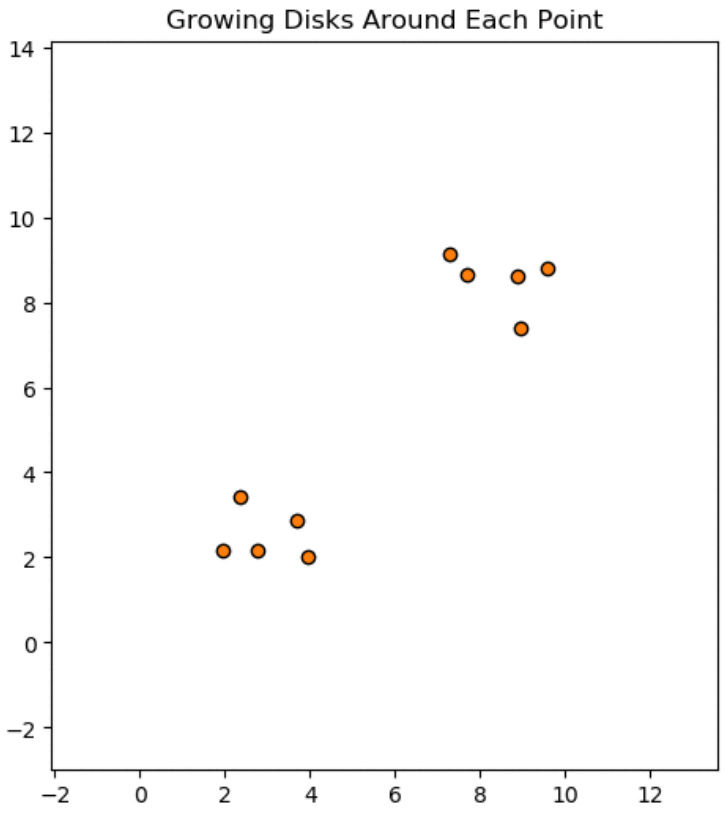
Persistent homology: visualization

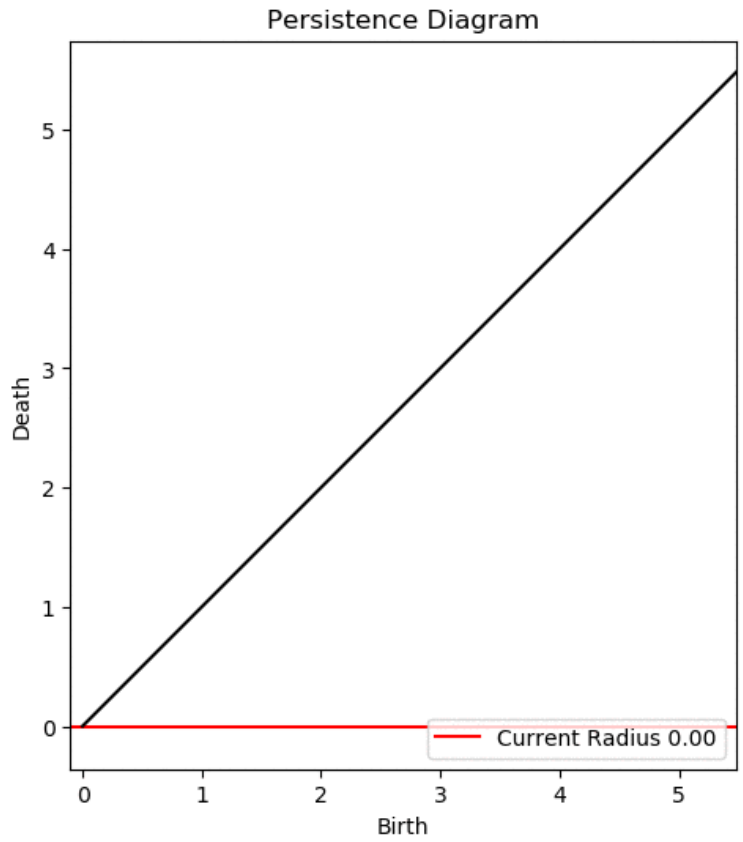
- Example: connected components (β_0), loops (β_1), higher-dimensional holes (β_i).



https://towardsdatascience.com/persistent-homology-with-examples-1974d4b9c3d0

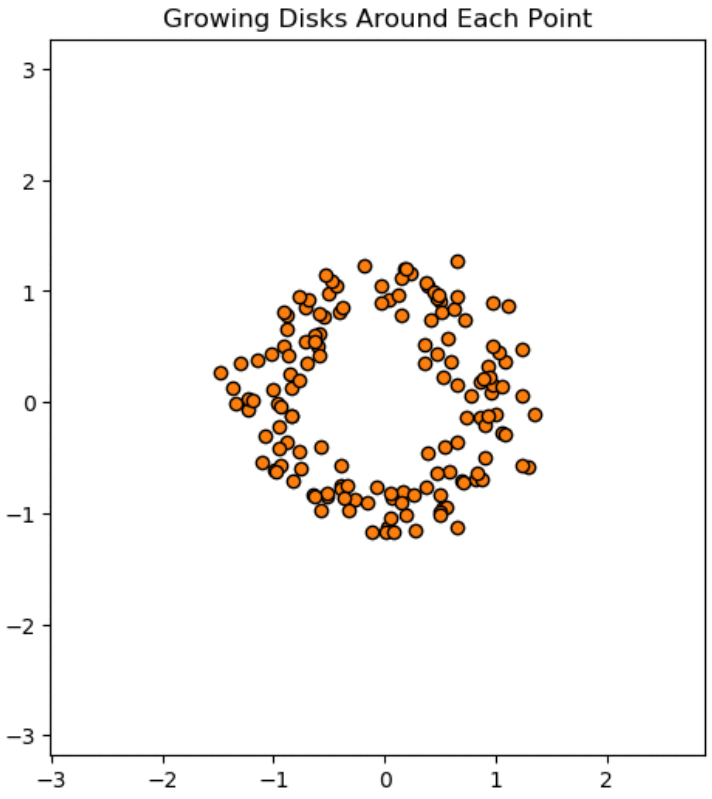
Persistent homology: connected components

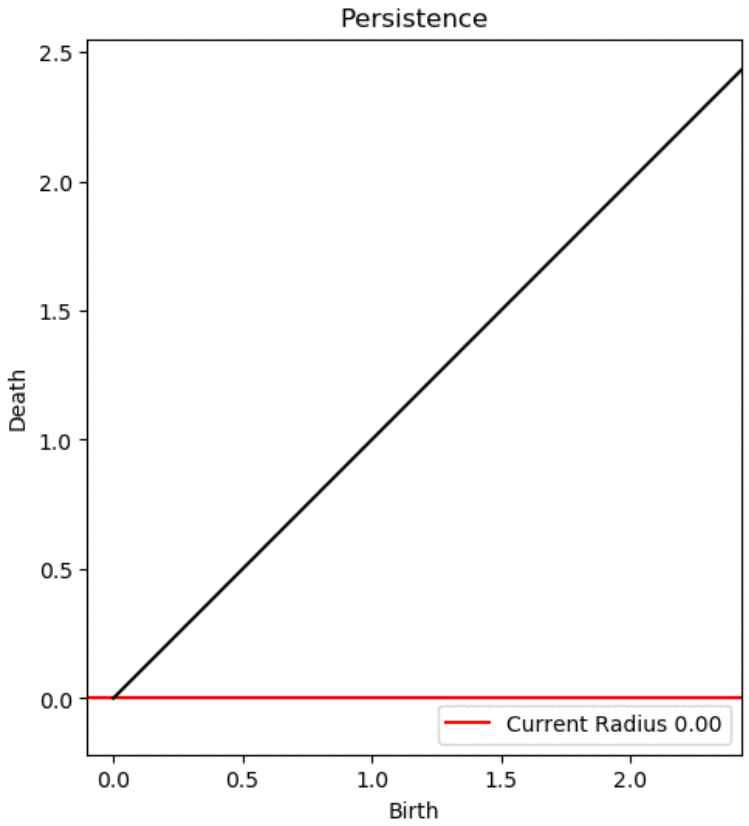




https://towardsdatascience.com/persistent-homology-with-examples-1974d4b9c3d0

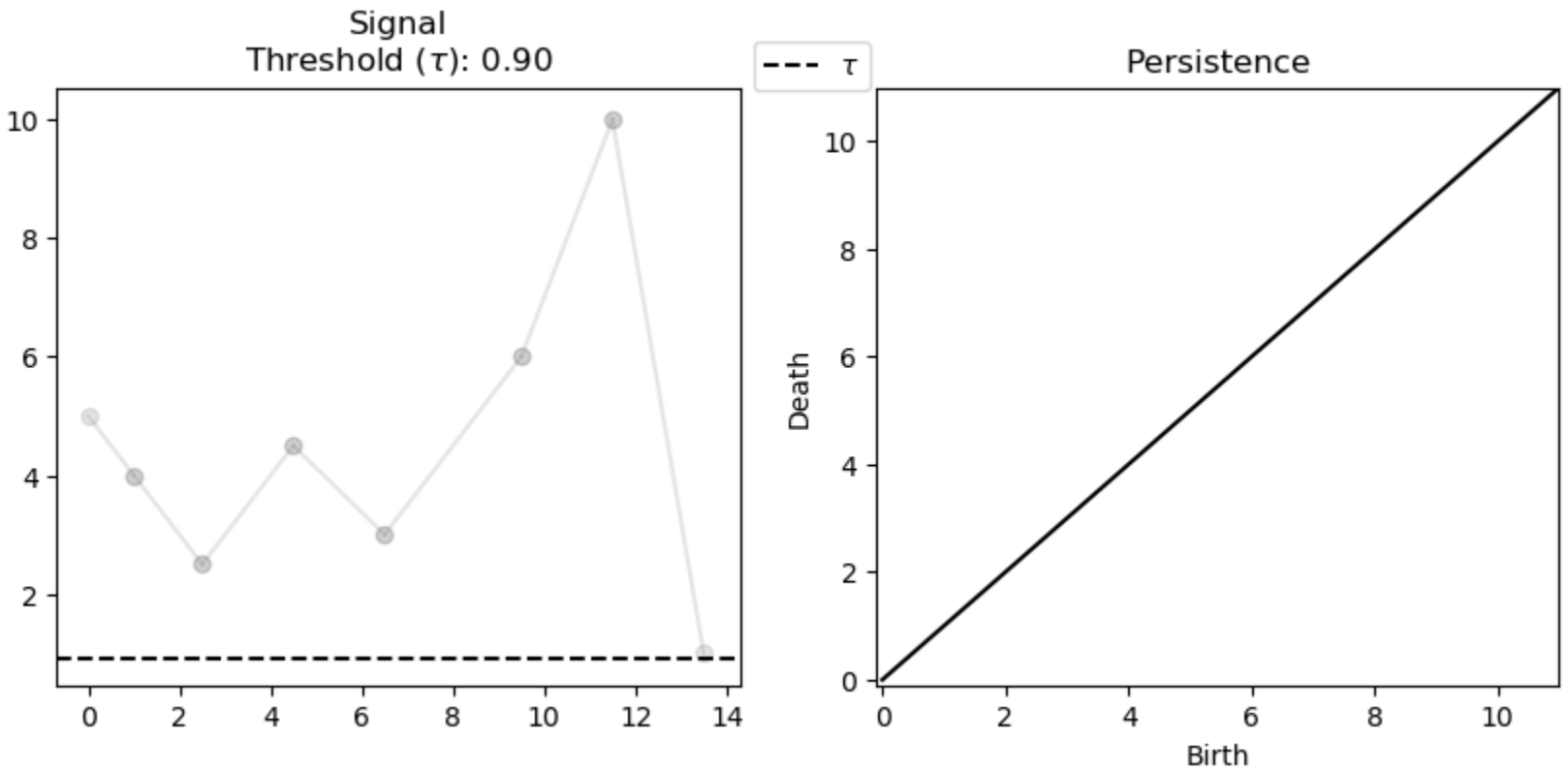
Persistent homology: loops





https://towardsdatascience.com/persistent-homology-with-examples-1974d4b9c3d0

Persistent homology: signals



https://towardsdatascience.com/persistent-homology-with-examples-1974d4b9c3d0

Persistent homology: applications

- Compression of signals and images via storing only persistence diagram (keeping track of critical points in the signal), or maybe even only a subset of the persistence values of highest magnitude.
- Using the persistence diagram as an additional global feature of a dataset, e.g. as input into a machine learning pipeline.
  - i.e. are the persistent features of a data cloud of gut microbiome compositions correlated with the healthiness of the individual.