# SVM
# PCA
# kNN indexes

**Vladislav Goncharenko**
Senior researcher

girafe ai

MSU
spring 2024

# Recap

Lecture 3:
Logistic Regression

- Linear classification
  - margin
  - loss functions
- Logistic regression
  - sigmoid derivation
  - Maximum Likelihood Estimation (MLE)
  - logistic loss
  - probability calibration
- Multiclass aggregation strategies
  - One vs Rest
  - One vs One
- Metrics in classification
  - Accuracy, Balanced accuracy
  - Precision, Recall, F-score
  - ROC curve, PR curve, AUC
  - Confusion matrix
- Hardware for ML

# Outline

1. Support Vector Machine (SVM)
    a. Hinge loss
    b. Kernel trick
2. Dimensionality reduction and PCA
    a. Problem statement
    b. Singular Value Decomposition
    c. Eckart–Young theorem
    d. Equivalent definitions
    e. Data normalization
3. k Nearest Neighbors
    a. kNN indexes
    b. HNSW

# Support Vector Machine

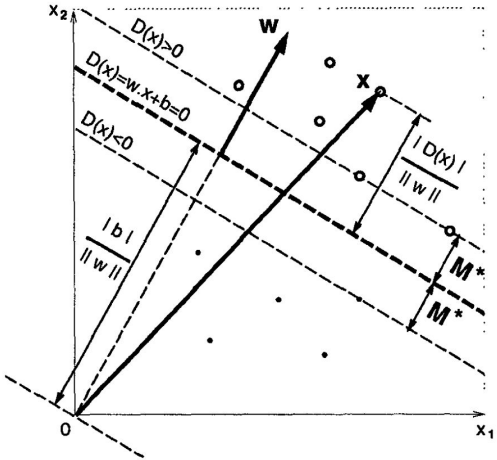
girafe
ai

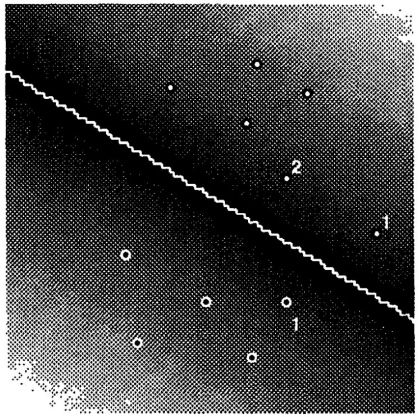
01

# Support Vector Machine

1. History
2. Motivation
3. Solution for separable design
4. Inseparable design, soft margin
5. Kernels
    a. Kernel definition (Hilbert spaces, inner product, positive semidefiniteness)
    b. Kernels properties (addition, infinite sums)
    c. Types of kernels (poly, exponential, gaussian)
6. Current state

# History



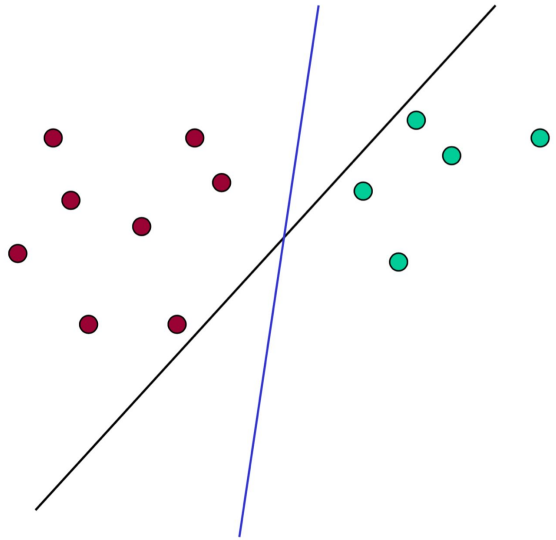




**1963**: SVM introduced by Soviet mathematicians
Vladimir Vapnik and Alexey Chervonenkis

**1992**: kernel trick (Vapnik, Boser, Guyon)

**1995**: soft margin (Vapnik, Cortes)

# Motivation



Linear separable case

Many separating hyperplanes exist

Maximize width

# Margin

$$y \in \{1, -1\}$$

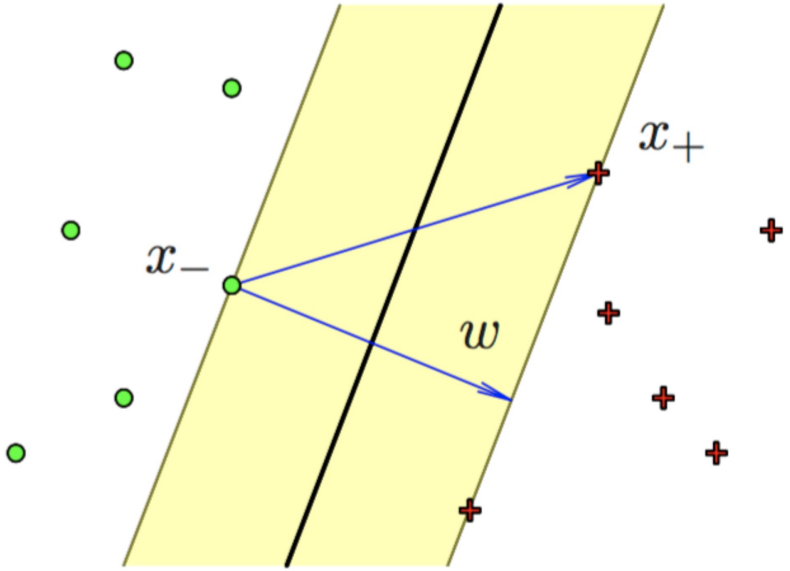
$$y_i = 1 : w^T x_i - c > 0$$

$$y_i = -1 : w^T x_i - c < 0$$

$$c_+(w) = \min_{y_i=1}(w^T x_i)$$

$$c_-(w) = \max_{y_i=-1}(w^T x_i)$$

$$\rho(w) = \frac{c_+(w) - c_-(w)}{2}$$




$$\rho\left(\frac{w_0}{||w_0||}\right) = \frac{1}{||w_0||}$$

# Optimization problem

$$y_i = 1 : w^T x_i - c > 0$$
$$y_i = -1 : w^T x_i - c < 0$$
$$M_i = y_i \cdot (w^T x_i - c)$$

$$\rho(w) = \frac{1}{||w||} \to \max_{w,c}$$
$$s.t.\ y_i(w^T x_i - c) \geq 1$$

Convex problem!

$$L(w, c, \alpha) = \frac{1}{2} w^T w - \sum_i \alpha_i (y_i (w^T x_i - c) - 1)$$

Many of them are zeros

# Hinge loss

$$L(w, c, \alpha) = \frac{1}{2} w^T w - \sum_i \alpha_i (y_i (w^T x_i - c) - 1)$$

$$L^{\text{hinge}} = (1 - M)_+$$

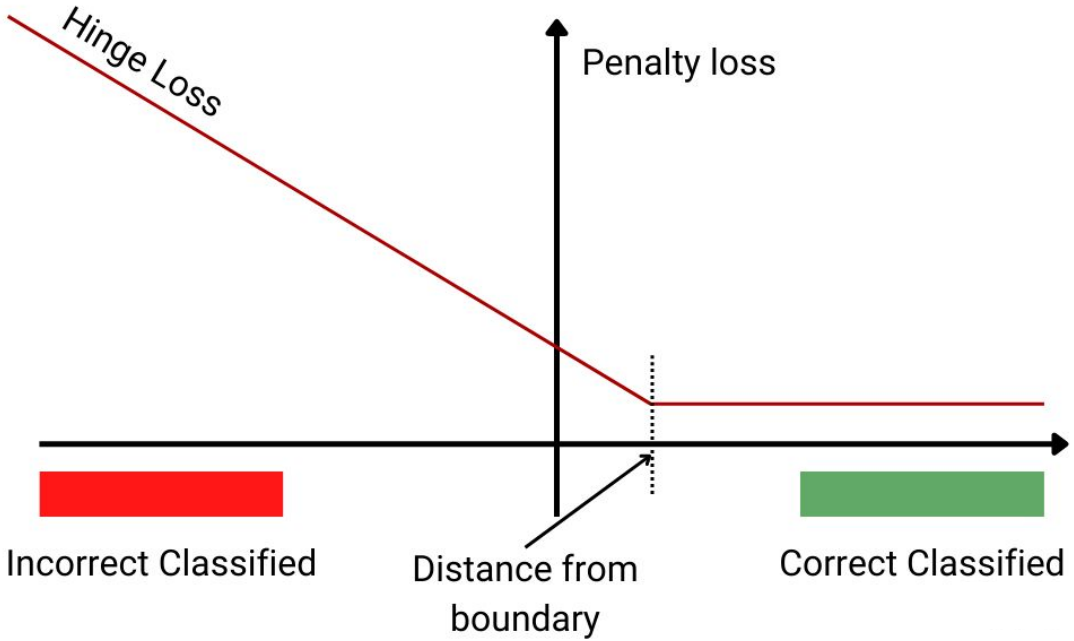
$$L(w, c, \alpha) = \frac{1}{2} ||w||_2^2 + \sum_i \alpha_i L_i^{\text{hinge}}$$

Loss is retrieved from “subject to” part.

Here “regularization” term is responsible for stripe width

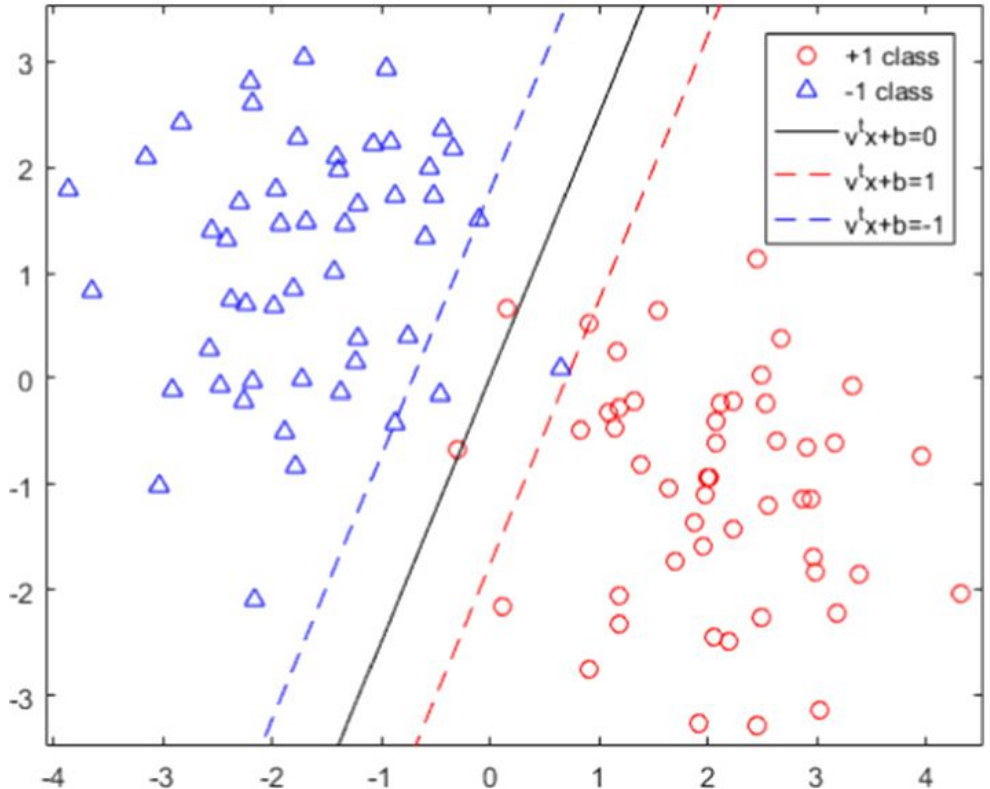
# Hinge loss

$$L^{\text{hinge}} = (1 - M)_+$$



# Inseparable case

Let our model mistake, but penalize that mistakes



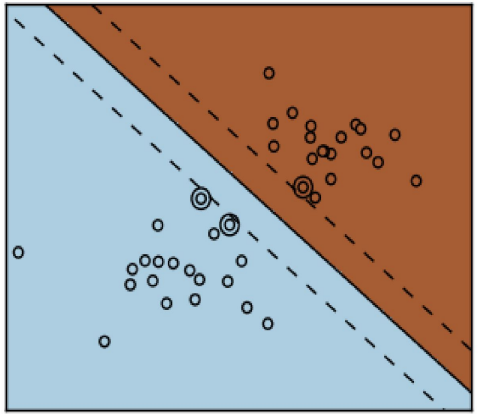
# Inseparable case

Let our model mistake, but penalize that mistakes

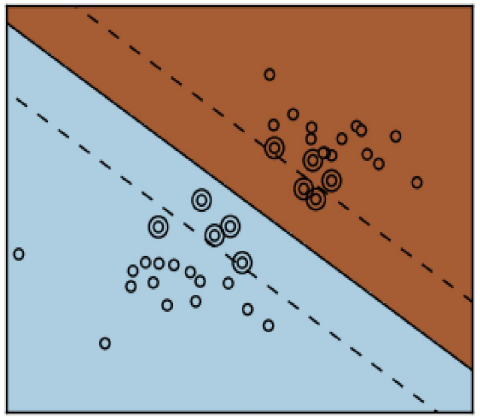
Implemented via margin slack variables

$$\begin{cases} \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{\ell}\xi_i \to \min\limits_{w,w_0,\xi}; \\ y_i\big(\langle w, x_i\rangle - w_0\big) \geqslant 1 - \xi_i, \quad i = 1, \ldots, \ell; \\ \xi_i \geqslant 0, \quad i = 1, \ldots, \ell. \end{cases}$$

Big C



Small C



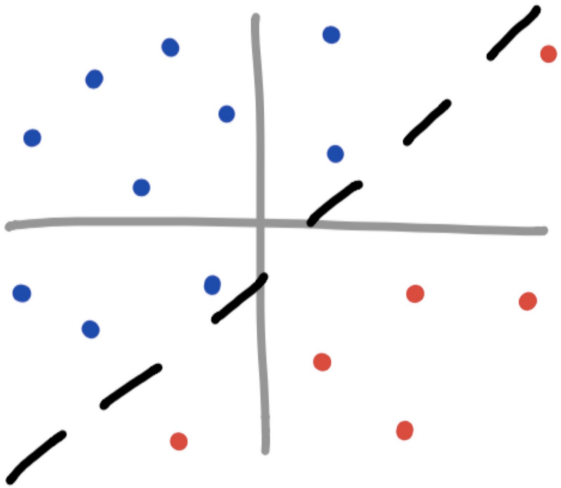
# Kernel trick

$$y_i = 1 : w^T x_i - c > 0$$
$$y_i = -1 : w^T x_i - c < 0$$

$$\begin{matrix} x \mapsto \phi(x) \\ w \mapsto \phi(w) \end{matrix} \implies < w, x > \mapsto < \phi(w), \phi(x) >$$

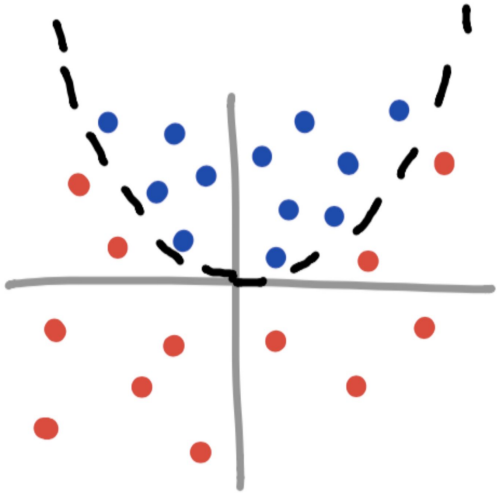
$$K(w, x) = < \phi(w), \phi(x) >$$

# Kernel types



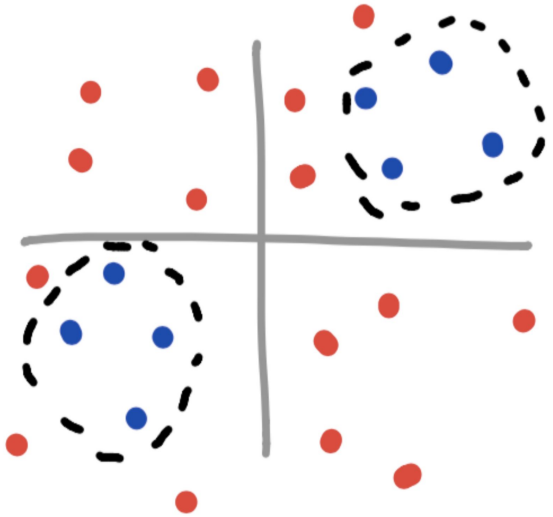
$K(w,x) =< \mathrm{w}, x >$

Linear



$K(w,x) = (\gamma < w,x > +r)^d$

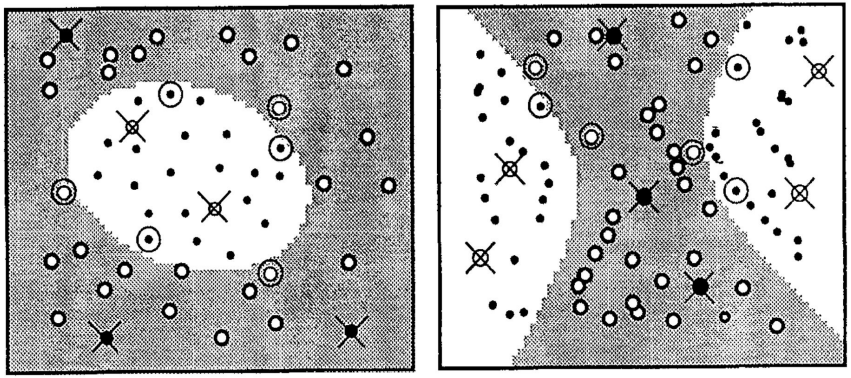
Polynomial



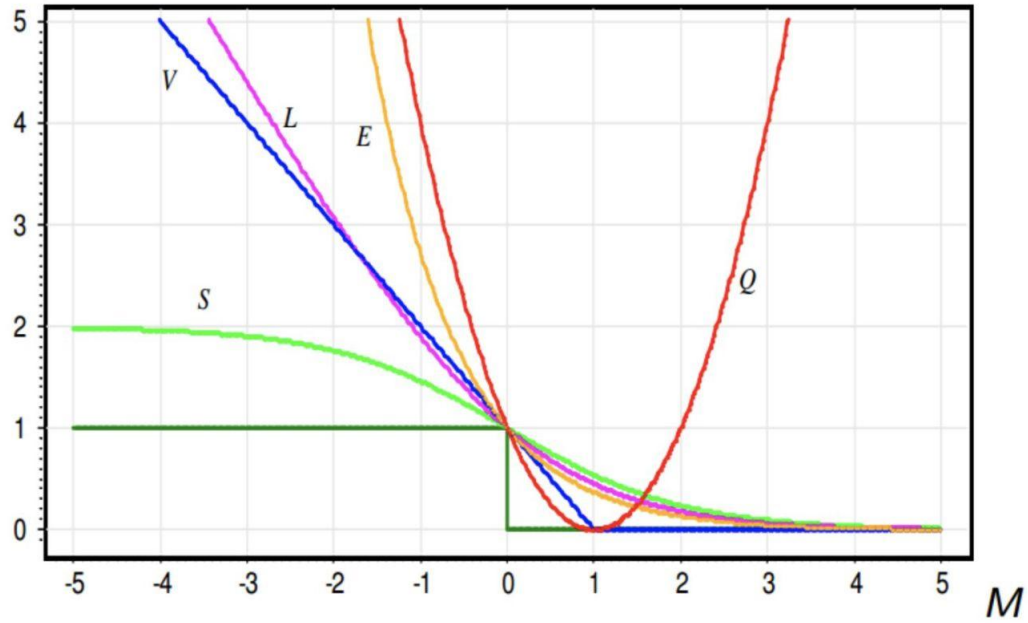
$K(w,x) = e^{-\gamma\|w-x\|^2}$

Gaussian
radial basis function

# Current state



# Classification losses




$$Q(M) = (1 - M)^2$$
$$V(M) = (1 - M)_+$$
$$S(M) = 2(1 + e^M)^{-1}$$
$$L(M) = \log_2(1 + e^{-M})$$
$$E(M) = e^{-M}$$

Loss functions for classification

# Principal Component Analysis

girafe ai

02

# Principal Component Analysis

$$x_1, \ldots, x_n \rightarrow g_1, \ldots, g_k, k \leq n$$

$$U : UU^T = I, G = XU$$

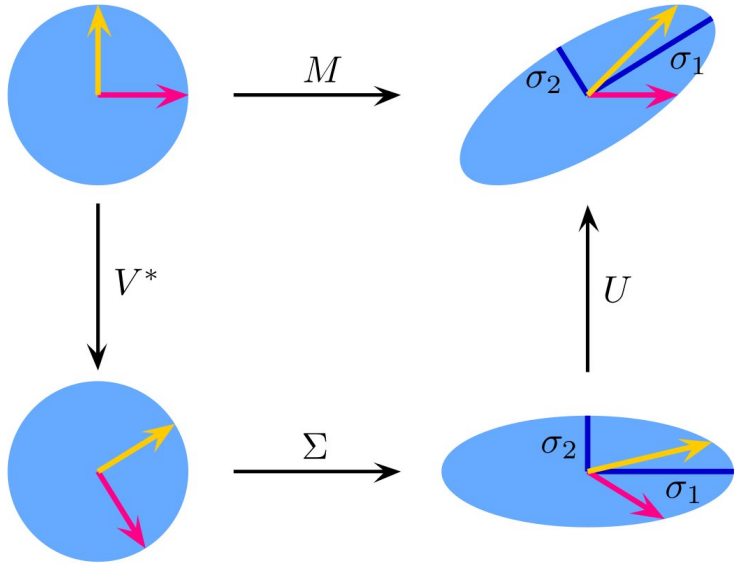
$$\hat{X} = GU^T \approx X$$

$$||GU^T - X|| \rightarrow \min_{G,U} \; s.t. \; rank(G) \leq k$$

# Singular value decomposition

factorization of a real or complex matrix into a rotation, followed by a rescaling followed by another rotation.

It generalizes the eigendecomposition of a square normal matrix with an orthonormal eigenbasis to any m×n matrix. It is related to the polar decomposition.




$$M = U \cdot \Sigma \cdot V^*$$

# PCA derivation

$$||GU^T - X|| \to \min_{G,U} \; s.t.\; rank(G) \leq k$$

$$X = V\Sigma U^T : ||GU^T - V\Sigma U^T||_2 = ||G - V\Sigma||_2$$

$$G = V\Sigma' : ||V\Sigma' - V\Sigma||_2 = ||\Sigma' - \Sigma||_2$$

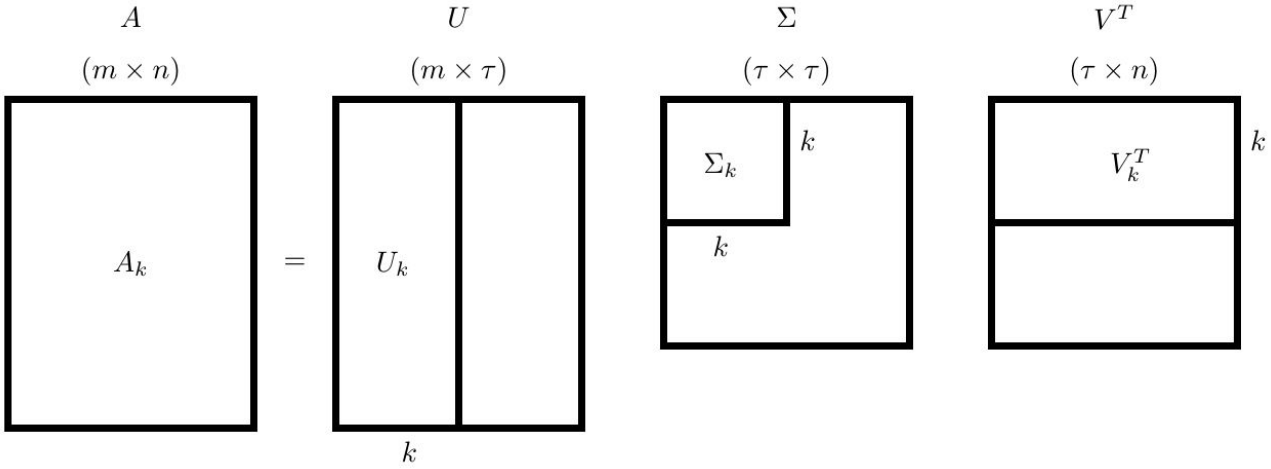
$$||A||_2 = \sigma_{max}(A) : ||\Sigma' - \Sigma||_2 = \sigma_k(\Sigma) = \sigma_k(X)$$

Eckart–Young–Mirsky theorem

# Singular value decomposition

$$||GU^T - X|| \to \min_{G,U} \; s.t.\; rank(G) \leq k$$

$$X = V\Sigma U^T \qquad \sigma_k(\Sigma) = \sigma_k(X)$$



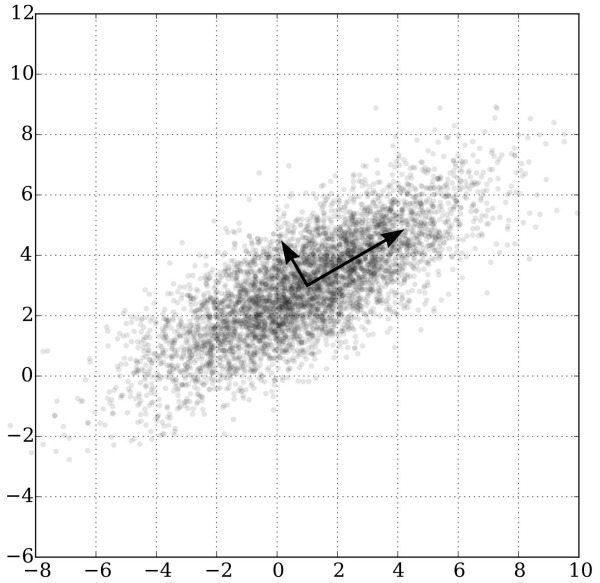

Eckart–Young–Mirsky theorem

# Another approach

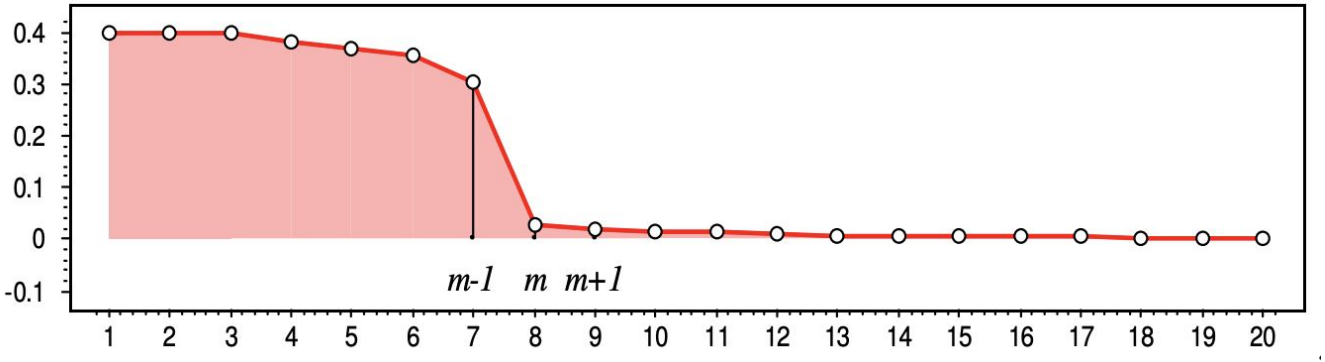
Residual variance maximization

Take new basis vectors greedy

Same result for G and U



# Relative variance threshold



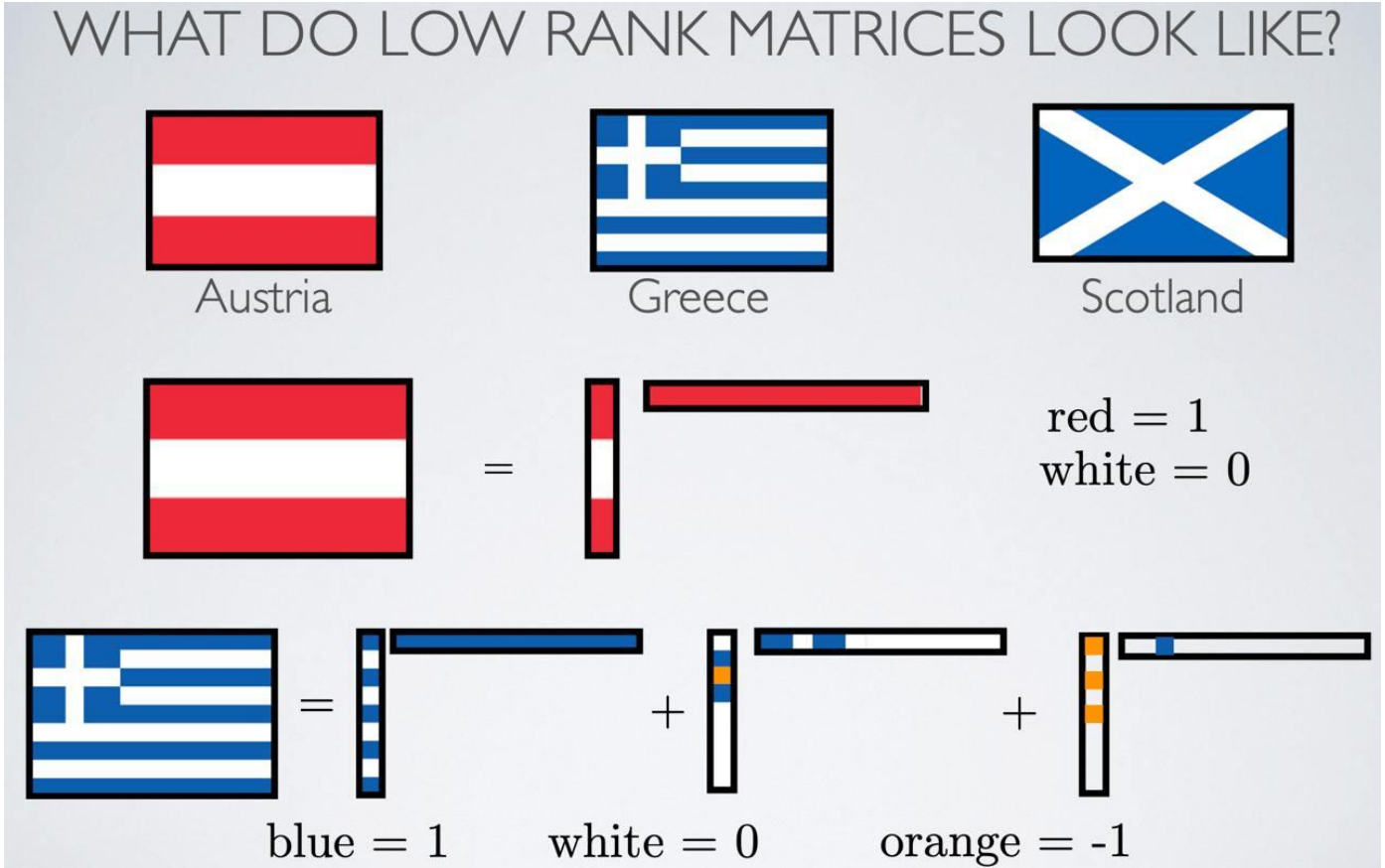

Get rid of low-variance components

$$E_m = \frac{\|GU^\intercal - F\|^2}{\|F\|^2} = \frac{\lambda_{m+1} + \cdots + \lambda_n}{\lambda_1 + \cdots + \lambda_n} \leqslant \varepsilon.$$

# PCA for visualization

Let’s walk through space…

# PCA on images



# PCA on images


16 base components

# Dimensionality reduction


50 components restoration

# Dimensionality reduction


250 components restoration

# Problem with PCA

Flat price prediction

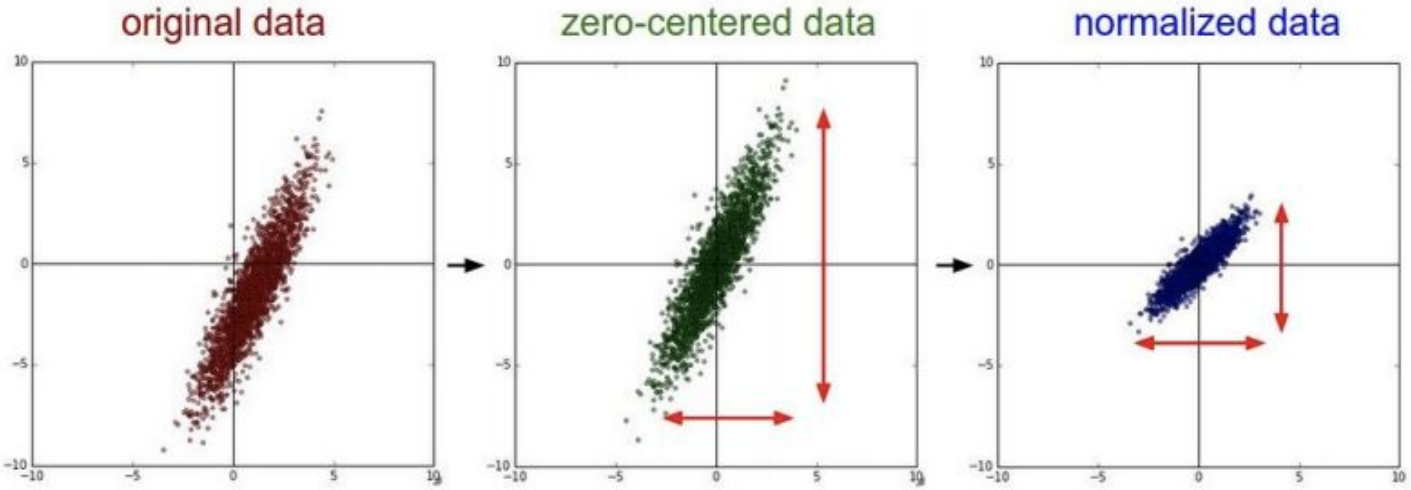
Features:

- square (sq. meters)
- distance to downtown (meters)

Solution: normalize data

# Data normalization

Always normalize data before PCA!!!



# Normalization types

To follow sklearn notation:

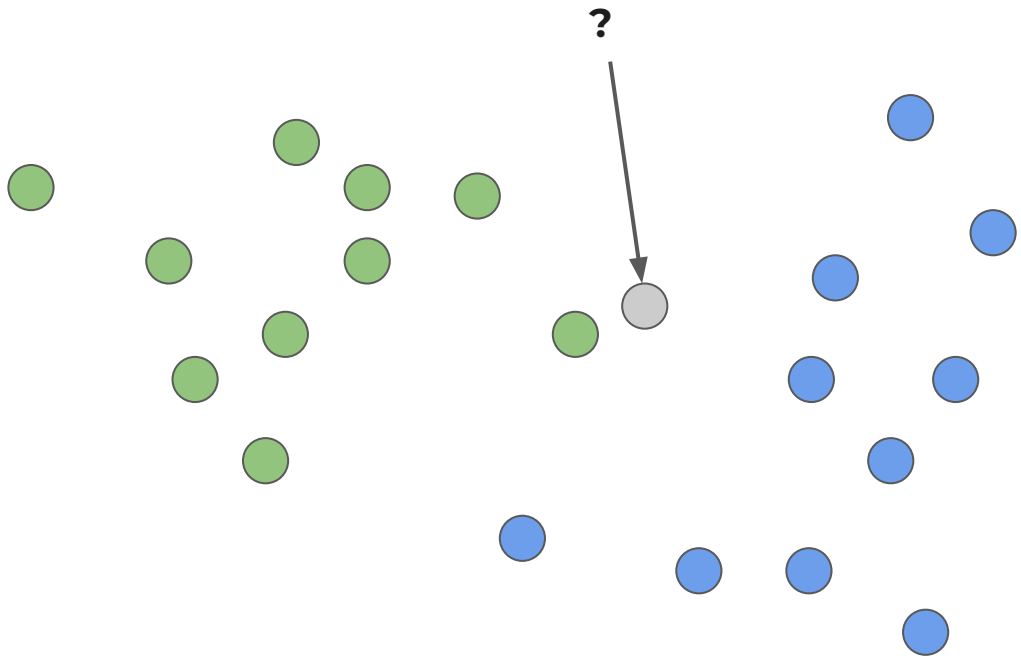
- MinMaxScaler
  - scales to 0 to 1 range
- StandardScaler
  - removes mean and std from values distribution

# k Nearest Neighbors

girafe ai

03

# Intuition



# kNN algorithm

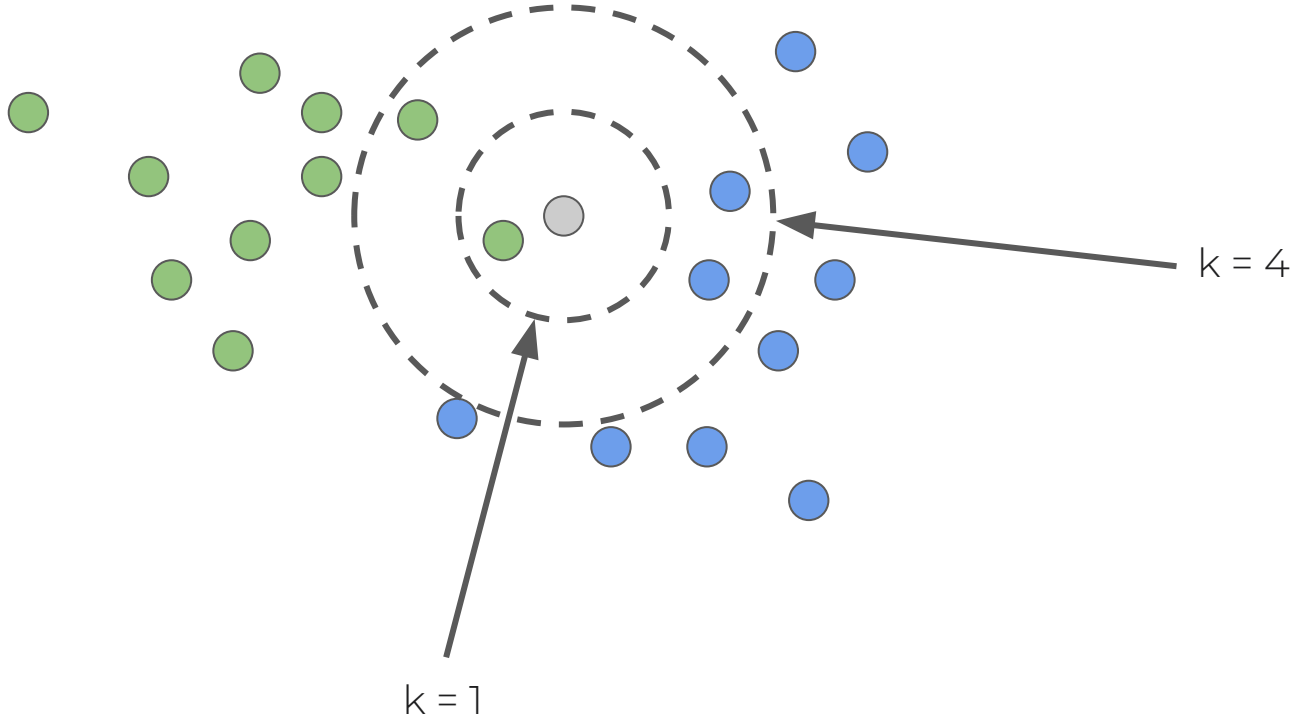
Given a new observation:

1. Calculate the distance to each of the samples in the dataset
2. Select samples from the dataset with the minimal distance to them
3. The label of the new observation will be the most frequent label among those nearest neighbors

How to answer to regression problem?

# How to make it better?

1. The number of neighbors k

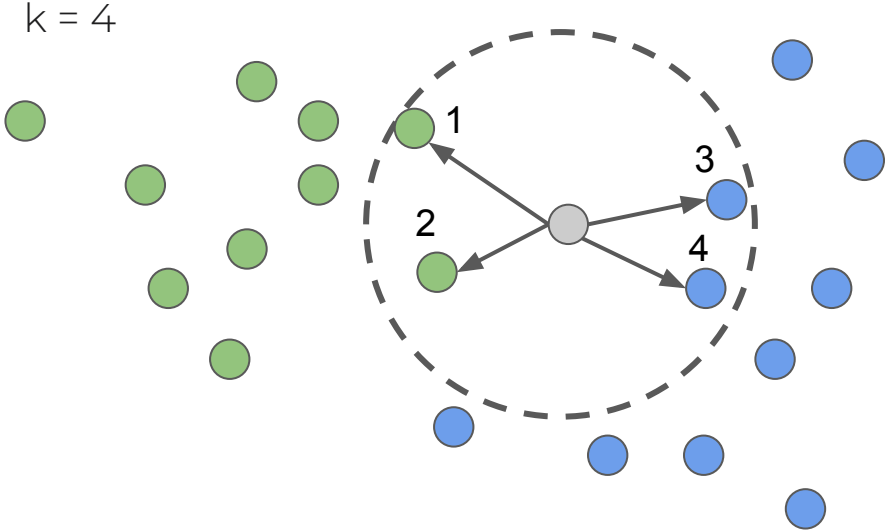


# How to make it better?

1. The number of neighbors k
2. The distance measure between samples
    a. Euclidean (L2)
    b. city block distance (aka Manhattan, L1)
    c. Minkowski distances
    d. cosine
    e. Hamming
        i. number of substitutions
    f. etc
3. Weighted neighbours
    a. fake! (never used in practice)

They are hyperparameters for kNN model

# Weighted kNN




- Weights can be adjusted according to the neighbors order

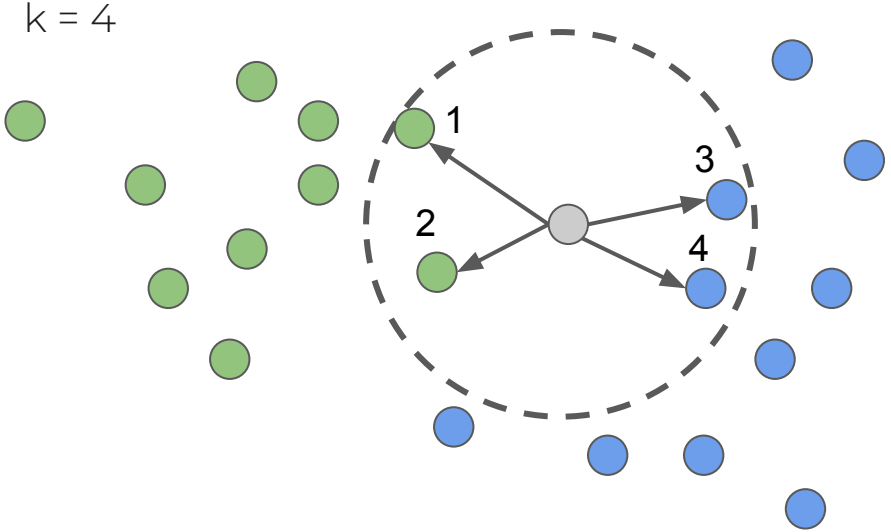
$$w(\mathbf{x}_{(i)}) = w_i$$

- or on the distance itself

$$w(\mathbf{x}_{(i)}) = w(d(\mathbf{x}, \mathbf{x}_{(i)}))$$

$$p_{\text{green}} = \frac{w(\mathbf{x}_1) + w(\mathbf{x}_2)}{w(\mathbf{x}_1) + w(\mathbf{x}_2) + w(\mathbf{x}_3) + w(\mathbf{x}_4)}$$

# Weighted kNN




- Weights can be adjusted according to the neighbors order

$$w(\mathbf{x}_{(i)}) = w_i$$

- or on the distance itself

$$w(\mathbf{x}_{(i)}) = w(d(\mathbf{x}, \mathbf{x}_{(i)}))$$

$$p_{\text{blue}} = \frac{w(\mathbf{x}_3) + w(\mathbf{x}_4)}{w(\mathbf{x}_1) + w(\mathbf{x}_2) + w(\mathbf{x}_3) + w(\mathbf{x}_4)}$$

# kNN indexes

girafe
ai

04

# Naive compute approach

Common object for kNN nowadays is embedding (usually by neural networks).

Let's suppose we have **million (10^6)** of floating point **embeddings in 100 dimensional** space.

We have a computer with **1 GHz (10^9) operations** frequency.

Then we want to find **10 nearest neighbours** of a new given embedding.

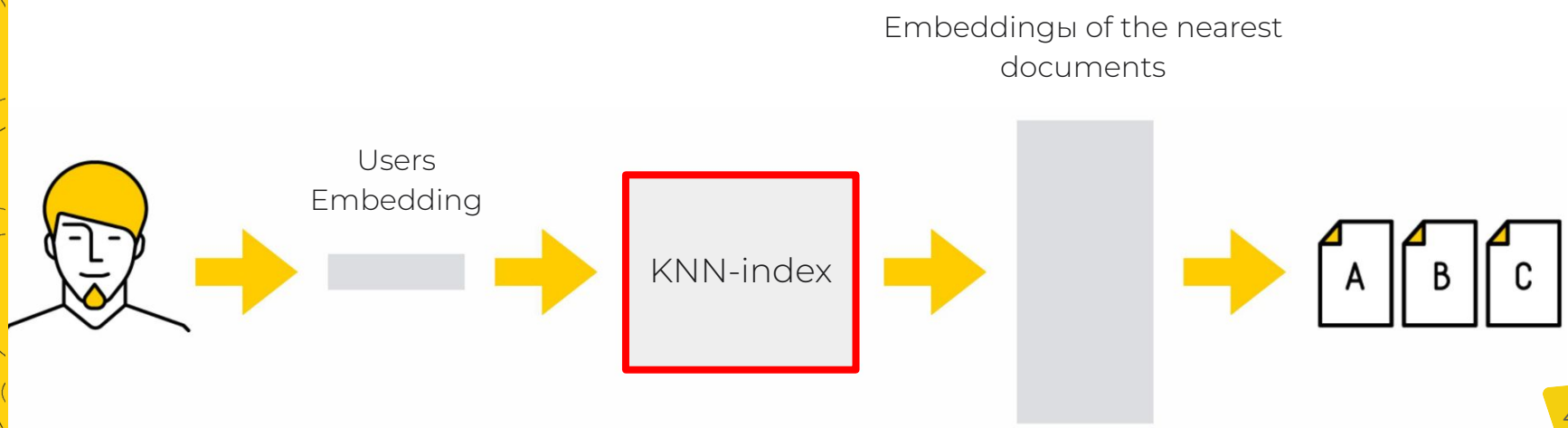
How much time will it take?

Is this time acceptable for realtime operation?

# kNN index

KNN index is a data structure
that allows you to quickly find the N nearest vectors for a given vector.

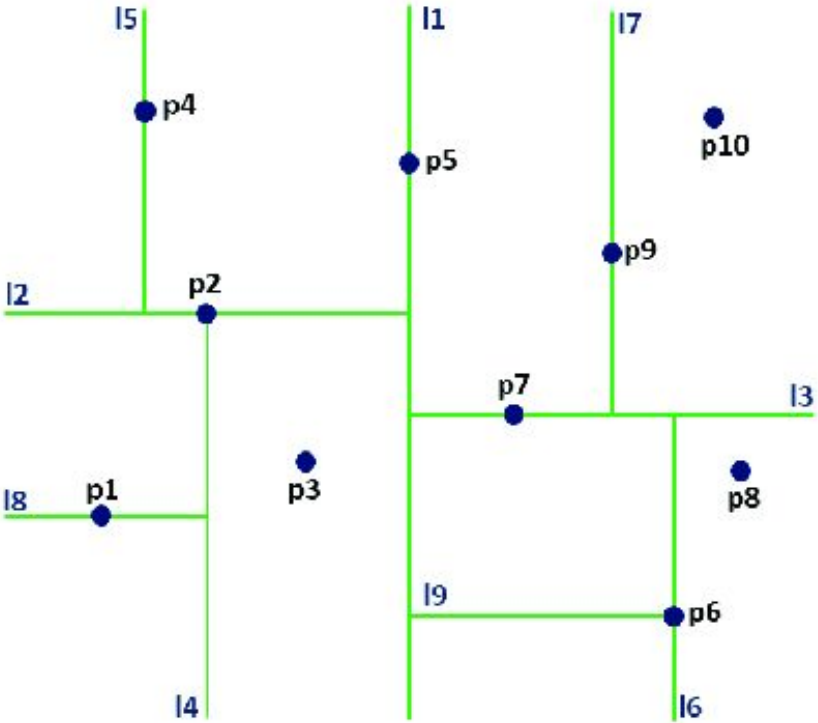
We perform some precompute beforehand to increase speed of final search.

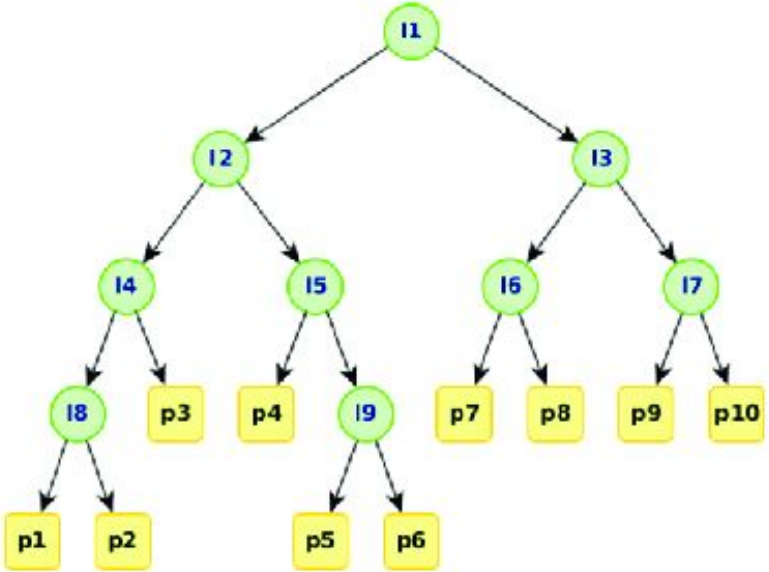


# kNN index ideas

1. Clustering
2. k-d tree
   a. default for scikit learn
   b. ball tree is extension for high dimensional cases
3. HNSW
   a. most common nowadays
4. many more...

# k-d tree

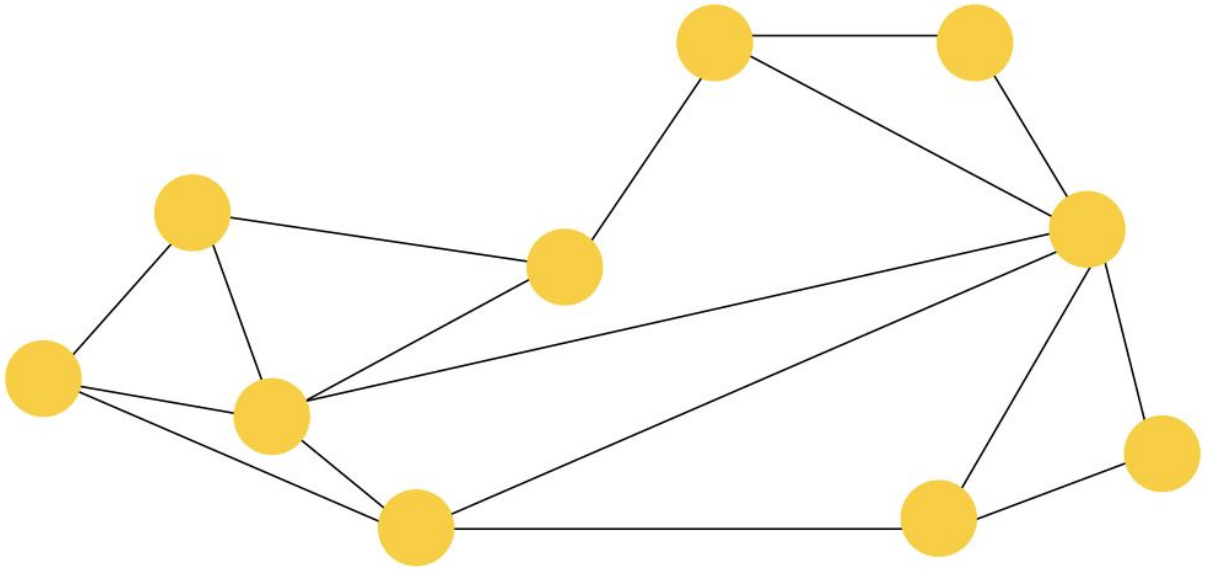

l5
l1
l7
p4
p10
p5
p9
l2
p2
p7
l3
p3
p8
l8
p1
l9
p6
l4
l6


l1
l2
l3
l4
l5
l6
l7
l8
p3
p4
l9
p7
p8
p9
p10
p1
p2
p5
p6

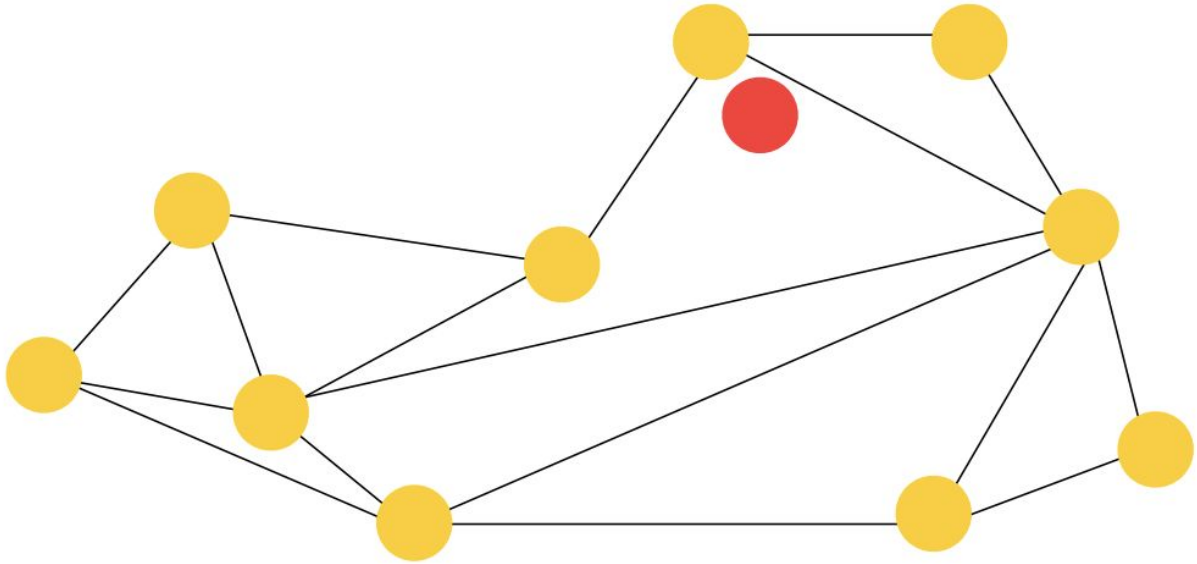
# Navigable Small World

- A graph where vertices are embeddings, edges are distances
- Adding nearby and remote vertices as edges
- We start from a random vertex, count the distances and move along the graph closer to the query point

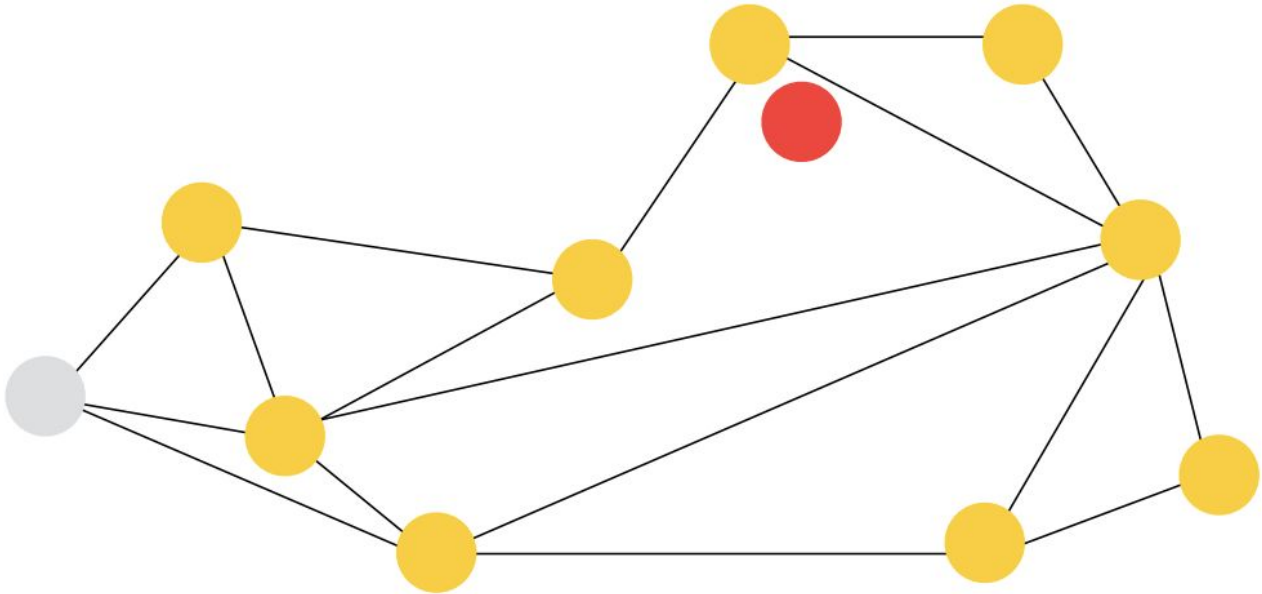
# NSW (Navigable Small World)



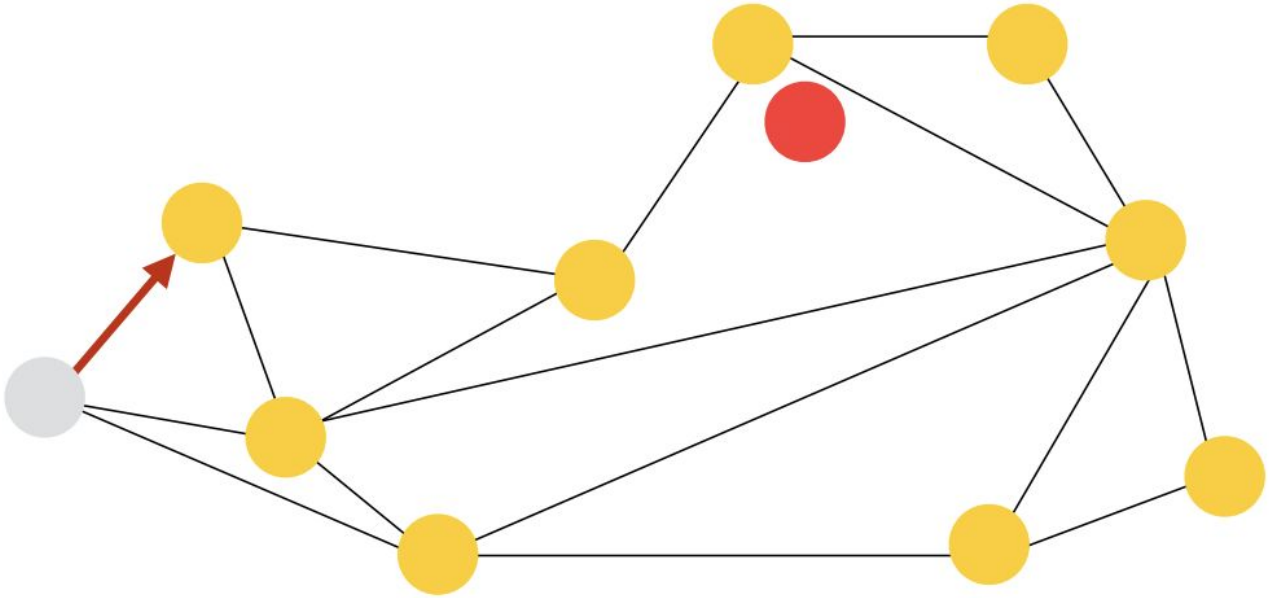
# NSW (Navigable Small World)



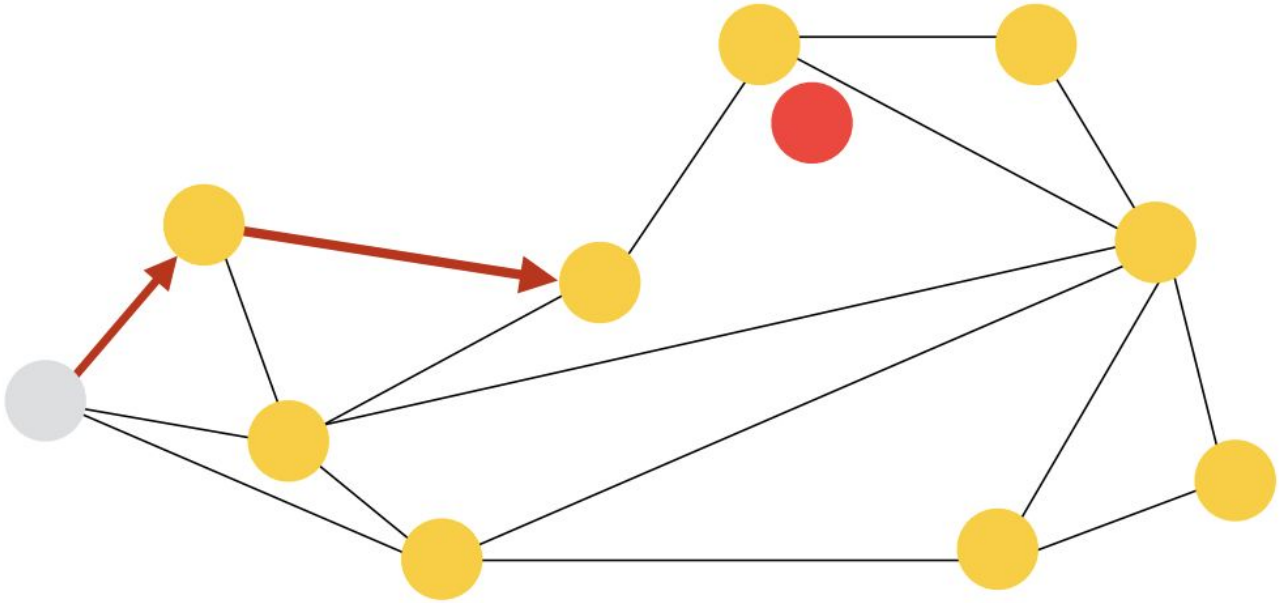
# NSW (Navigable Small World)



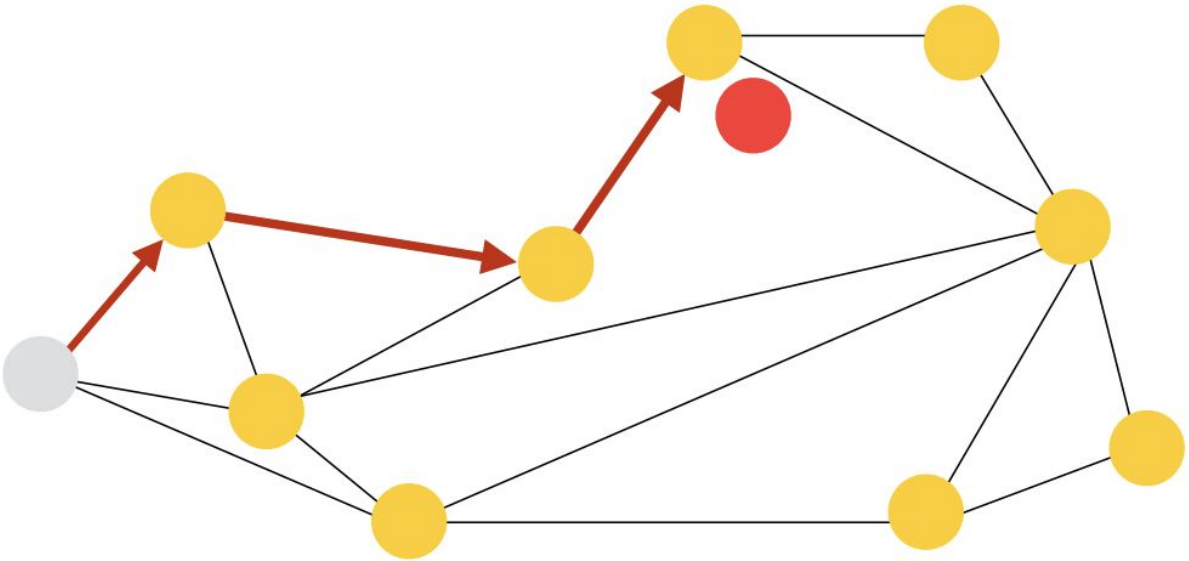
# NSW (Navigable Small World)



# NSW (Navigable Small World)



# NSW (Navigable Small World)

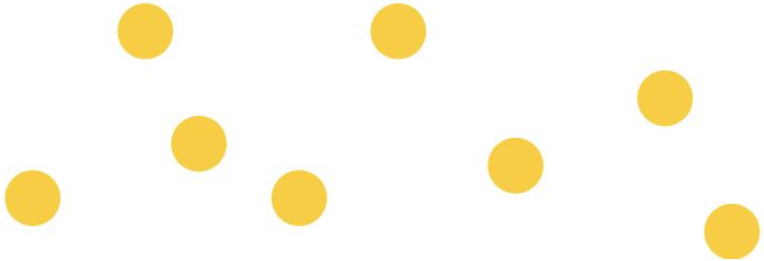


# Hierarchical Navigable Small World (HNSW)

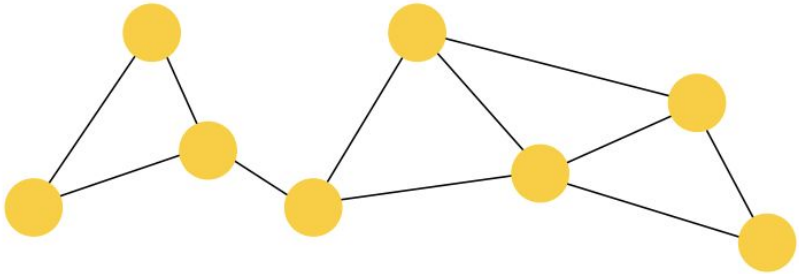
girafe ai

05

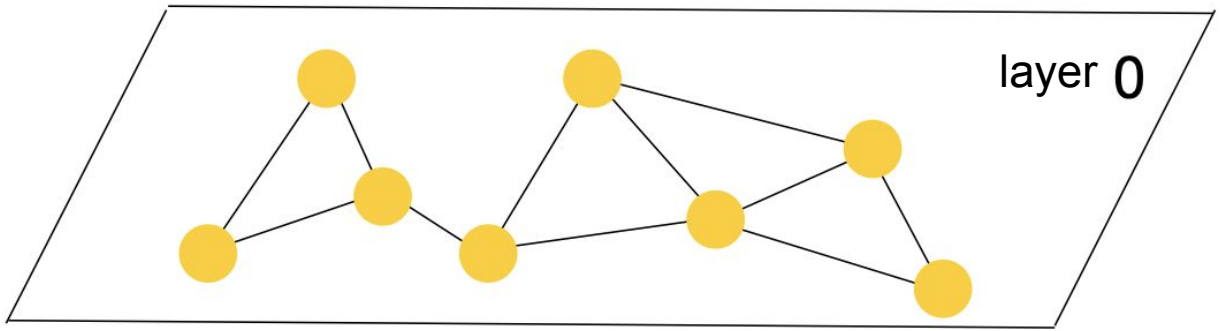
# HNSW



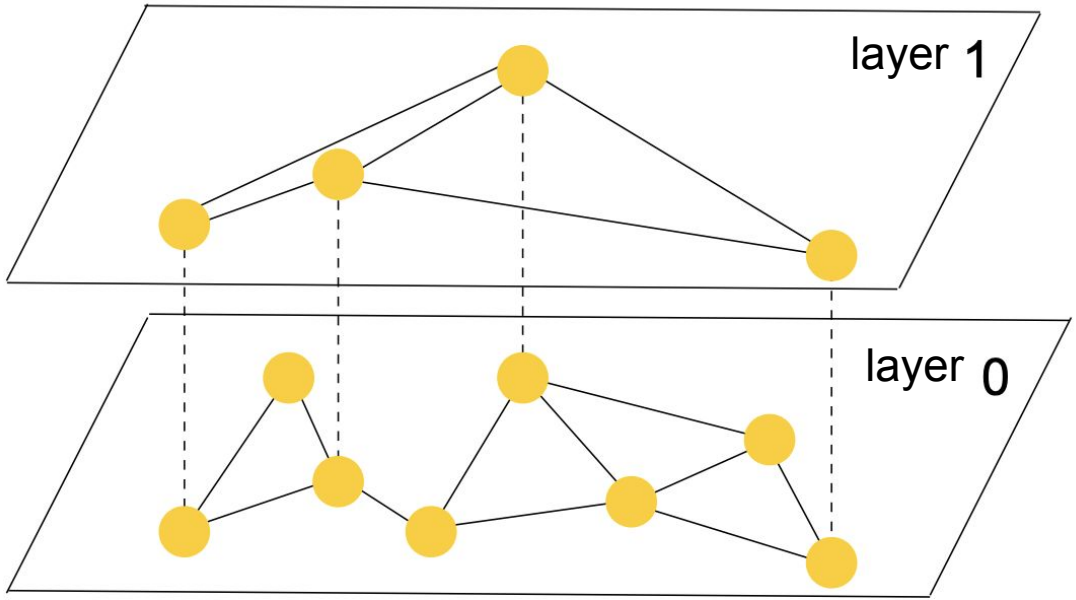
# HNSW



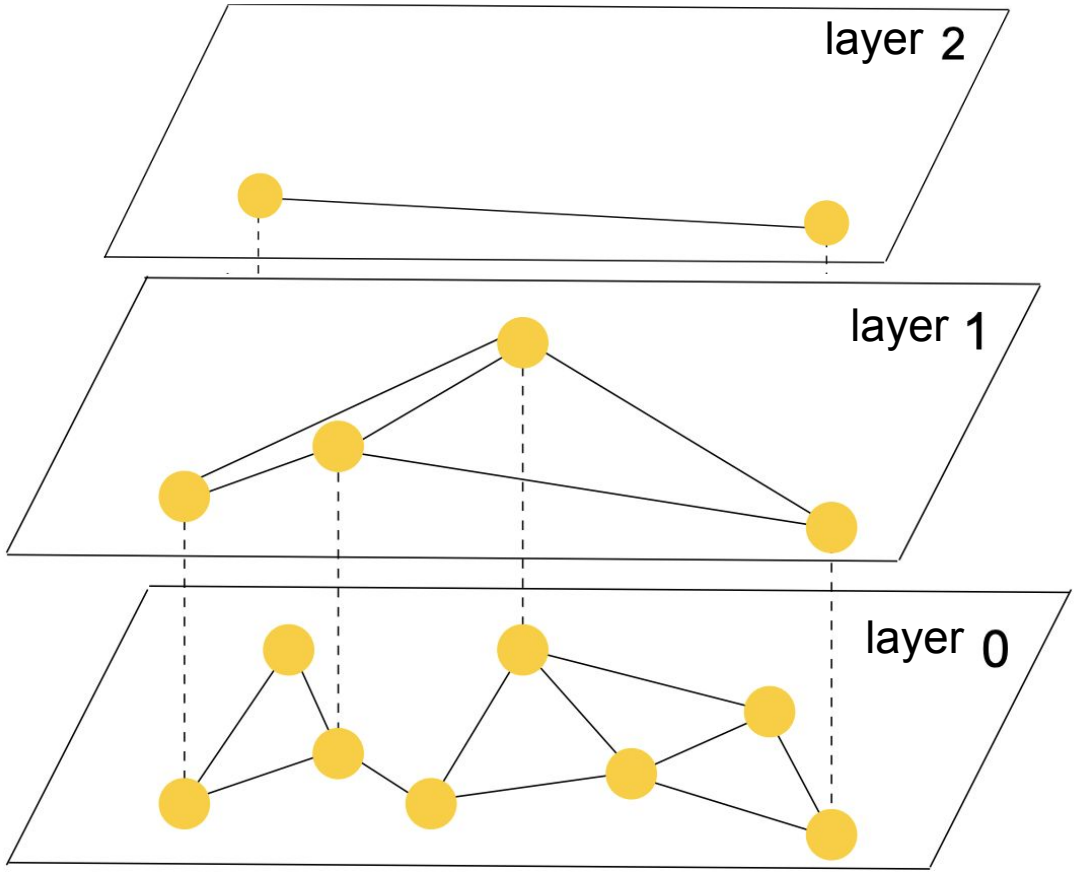
# HNSW



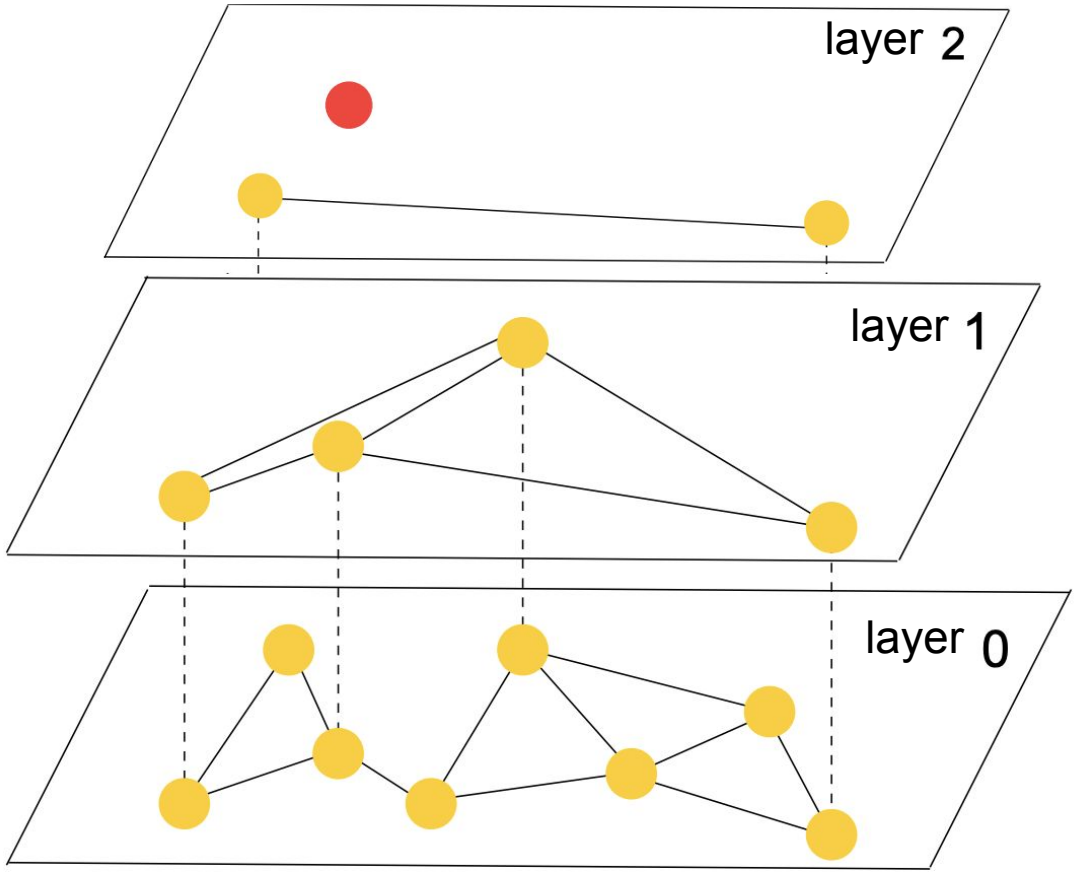
# HNSW



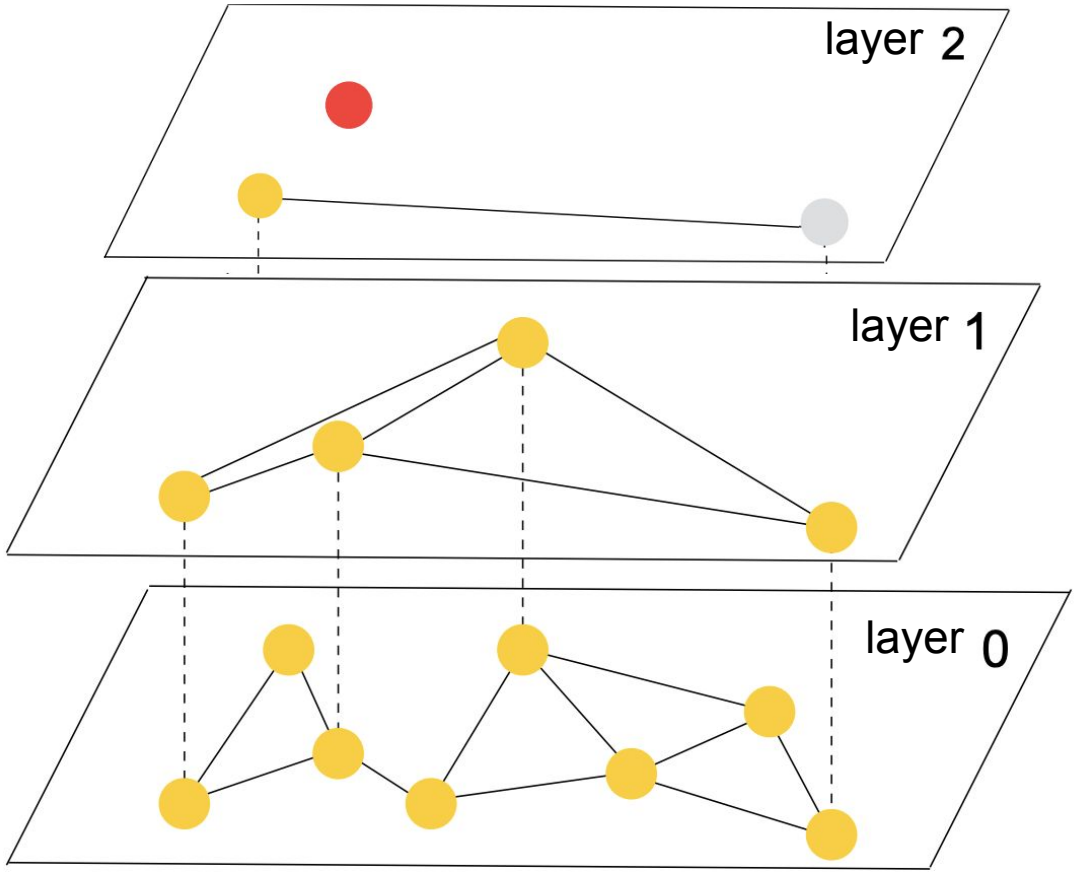
# HNSW



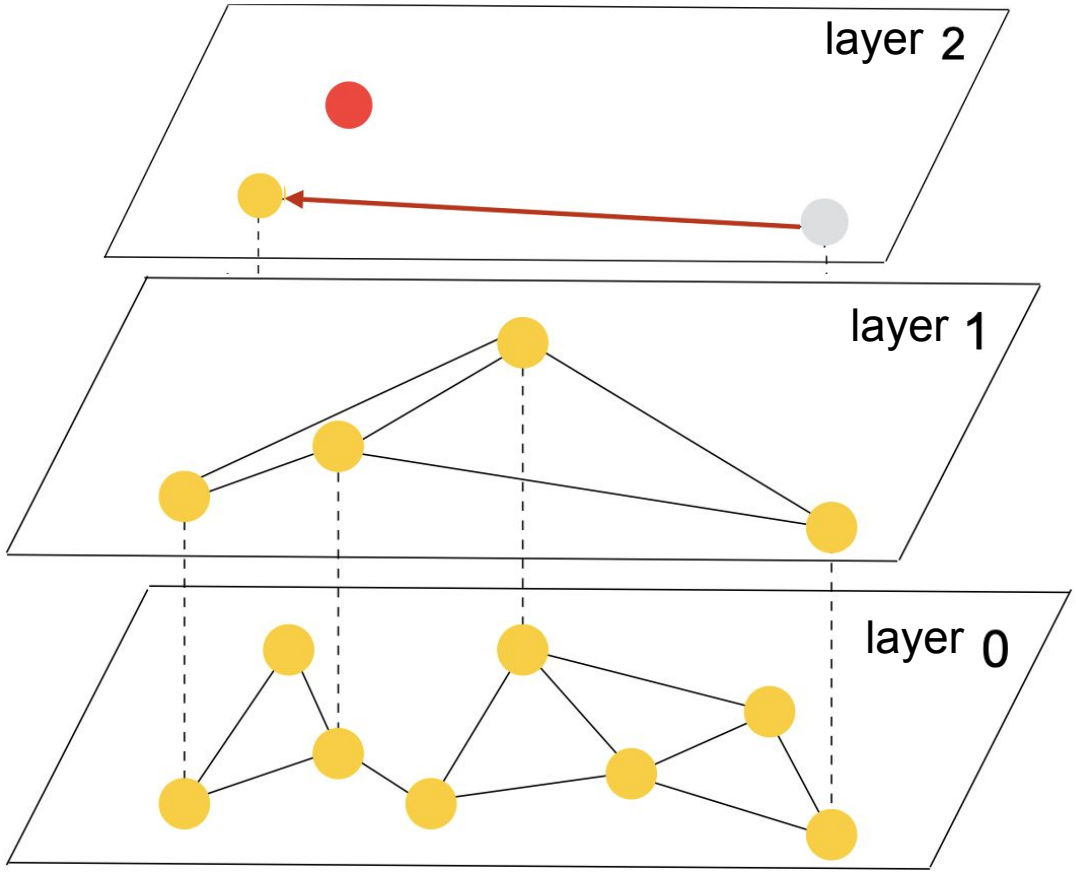
# HNSW



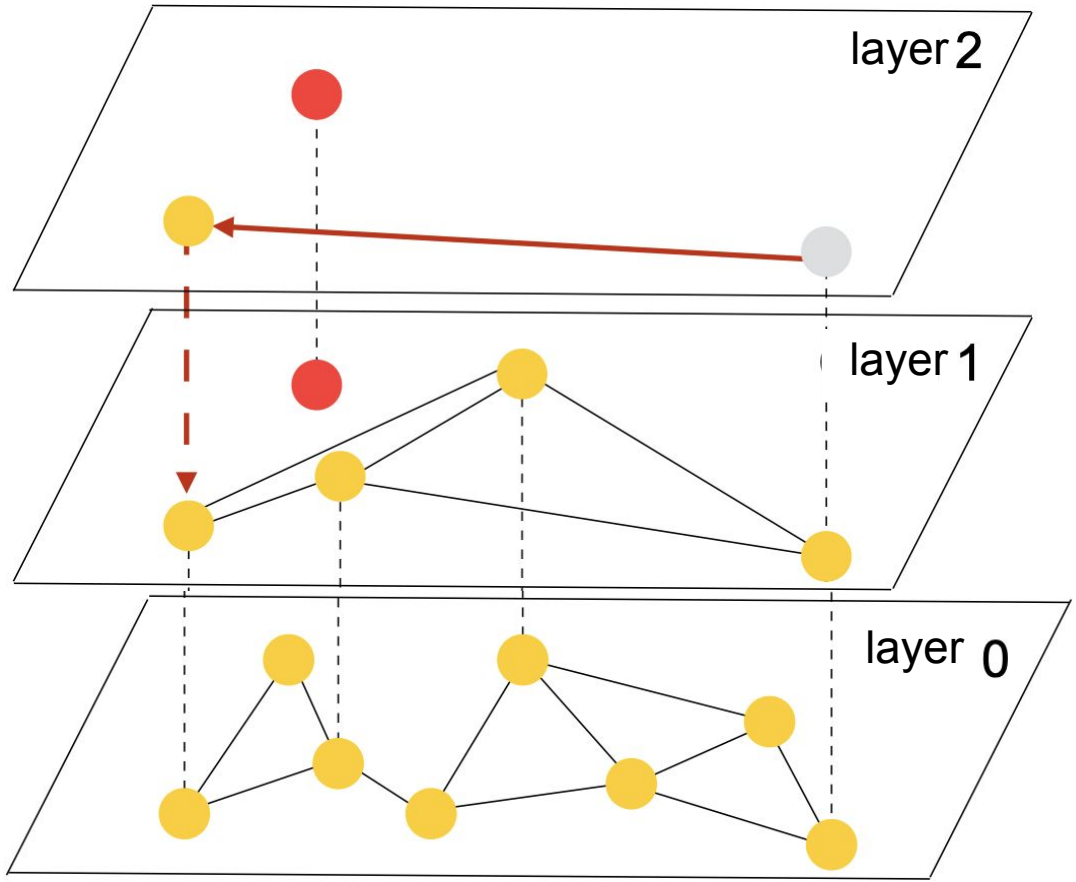
# HNSW



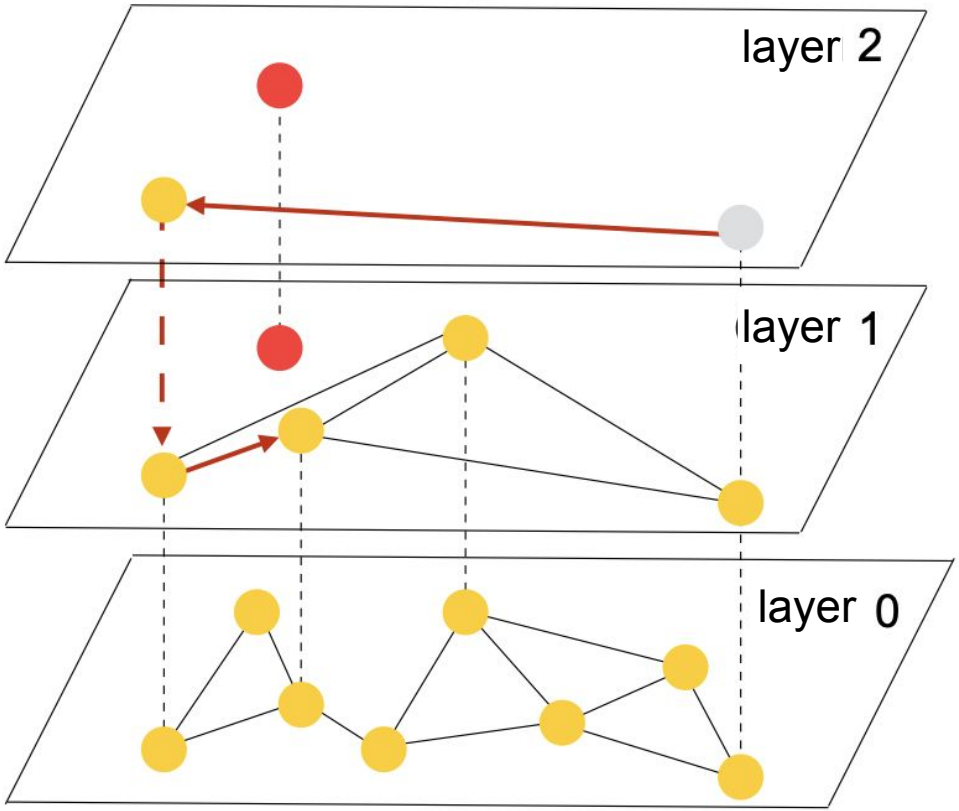
# HNSW



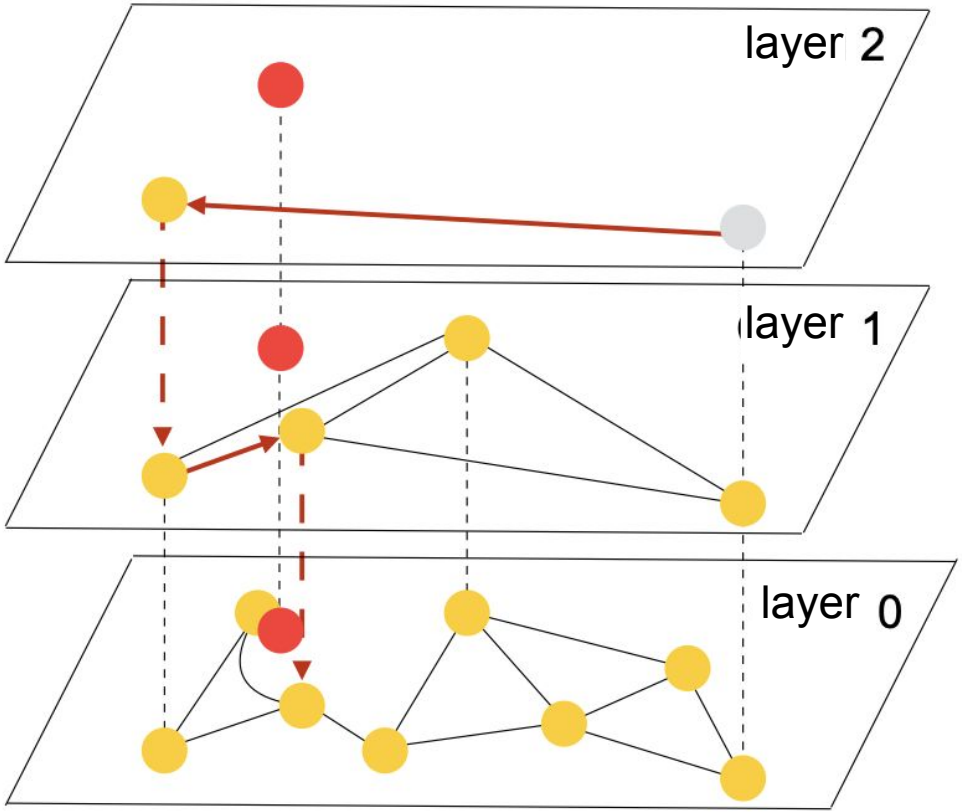
# HNSW



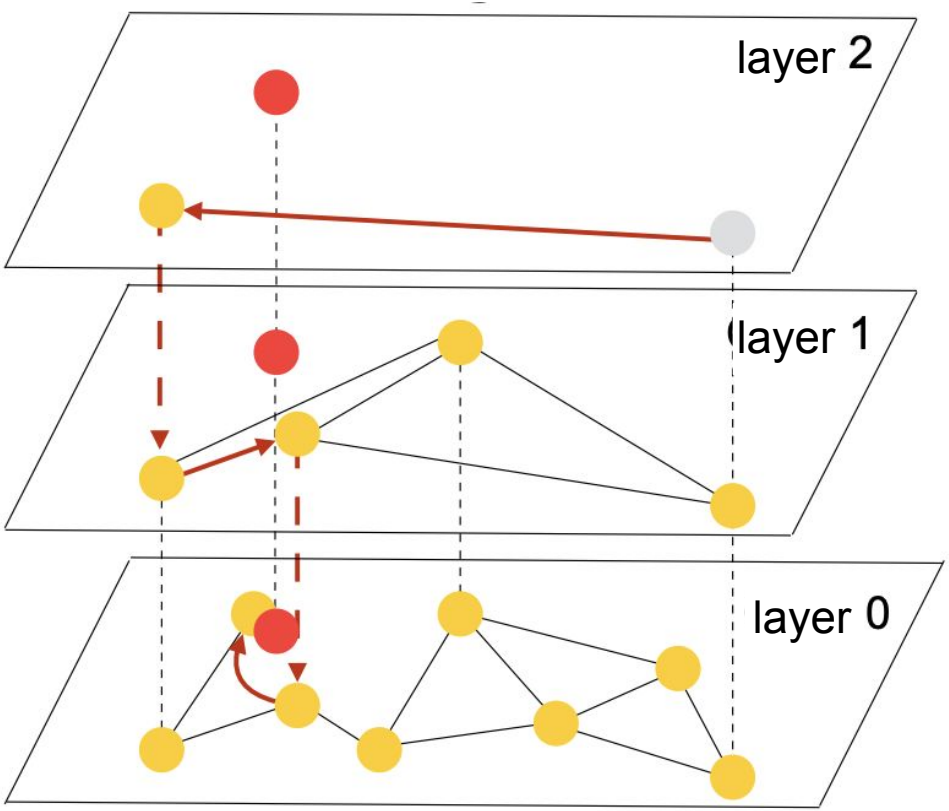
# HNSW



# HNSW

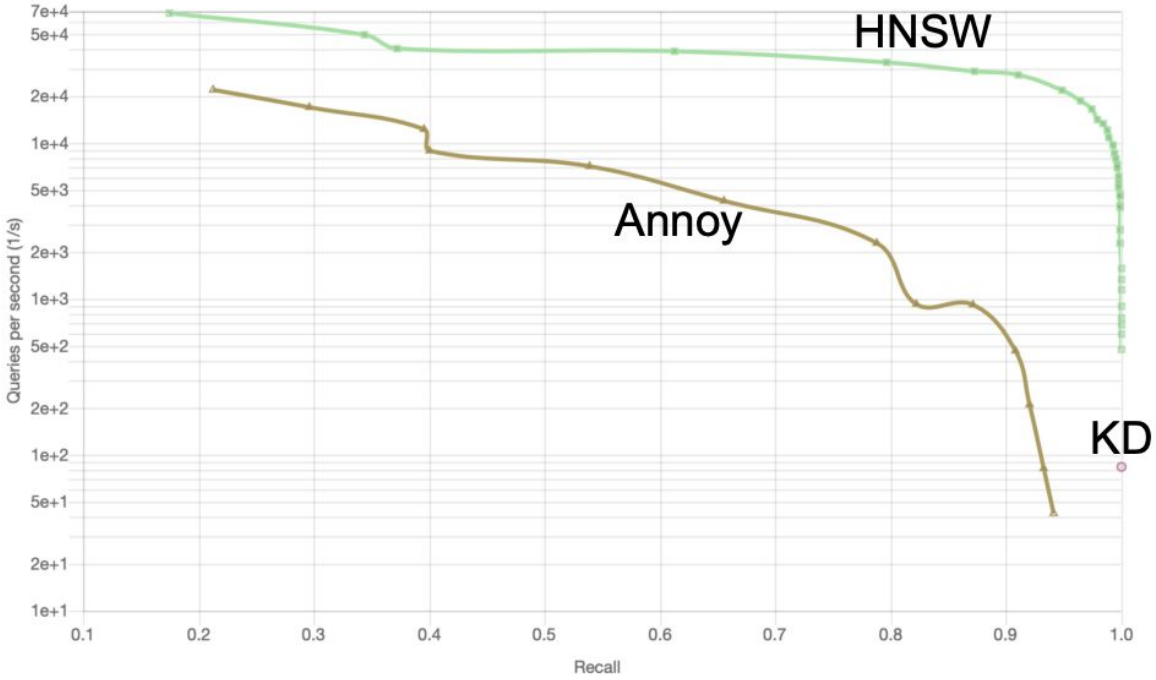


# HNSW



# Accuracy of approximate search

- In addition to speed, you need to find accurate nearby objects
- Comparison on ALS embedding last fm-64-dot, 360K



# kNN index implementations

- https://github.com/facebookresearch/faiss
  - https://habr.com/ru/companies/okkamgroup/articles/509204/
- https://github.com/nmslib/nmslib
- https://github.com/flann-lib/flann
- https://github.com/spotify/annoy
- https://lucene.apache.org/core/9_1_0/core/org/apache/lucene/util/hnsw/package-summary.html

# Vector databases

Also list of vector databases from openAI
https://platform.openai.com/docs/guides/embeddings/how-can-i-retrieve-k-nearest-embedding-vectors-quickly

Ready to production solutions

- https://github.com/qdrant/qdrant
  Has helm chart + distributed deployment
- https://www.trychroma.com/

# Revise

- Support Vector Machine (SVM)
  - History
  - Motivation
  - Solution for separable design
  - Inseparable design, soft margin
  - Kernels
- Dimensionality reduction and PCA
  - Problem statement
  - Eckart–Young theorem
  - Equivalent definitions
- kNN
  - kNN indexes
  - HNSW

# Next time

- Decision trees and thresholds

# Thanks for attention!

Questions?