

Unsupervised Learning

Andrea Passerini
passerini@disi.unitn.it

Machine Learning

Setting

- Supervised learning requires the availability of labelled examples
- Labelling examples can be an extremely expensive process
- Sometimes we don't even know how to label examples
- *Unsupervised* techniques can be employed to group examples into **clusters**

k-means clustering

Setting

- Assumes examples should be grouped into k clusters
- Each cluster i is represented by its mean μ_i

Algorithm

- 1 Initialize cluster means $\mu_1, \dots, \mu_k$
- 2 Iterate until no mean changes:
 - 1 Assign each example to cluster with nearest mean
 - 2 Update cluster means according to assigned examples

How can we define (dis)similarity between examples ?

(Dis)similarity measures

- Standard Euclidean distance in $\mathbb{R}^d$:

$$d(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{i=1}^d (x_i - x'_i)^2}$$

- Generic Minkowski metric for $p \geq 1$:

$$d(\mathbf{x}, \mathbf{x}') = \left(\sum_{i=1}^d |x_i - x'_i|^p \right)^{1/p}$$

- Cosine similarity (cosine of the angle between vectors):

$$s(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^T \mathbf{x}'}{\|\mathbf{x}\| \|\mathbf{x}'\|}$$

How can we define quality of obtained clusters ?

Sum-of-squared error criterion

- Let n_i be the number of samples in cluster $\mathcal{D}_i$
- Let μ_i be the cluster sample mean:

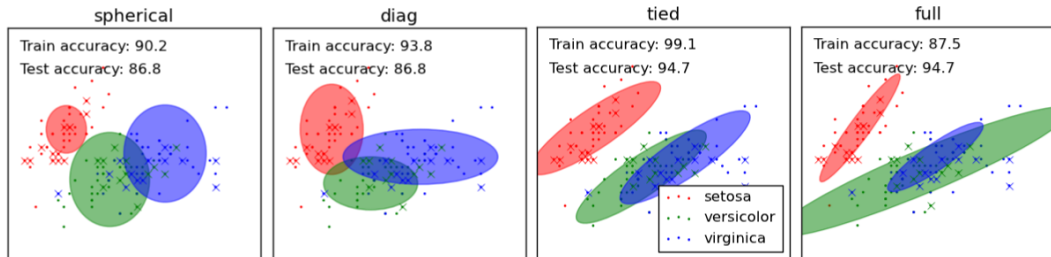
$$\mu_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}$$

- The sum-of-squared errors is defined as:

$$E = \sum_{i=1}^k \sum_{\mathbf{x} \in \mathcal{D}_i} \|\mathbf{x} - \mu_i\|^2$$

- Measures the squared error incurred in representing each example with its cluster mean

Gaussian Mixture Model (GMM)



Setting

- Cluster examples using a mixture of Gaussian distributions
- Assume number of Gaussians is given
- Estimate mean and possibly variance of each Gaussian

Parameter Estimation

- Maximum likelihood estimation cannot be applied as cluster assignment of examples is unknown
- Expectation-Maximization approach:
 - 1 Compute expected cluster assignment given current parameter setting
 - 2 Estimate parameters given cluster assignment
 - 3 Iterate

Example: estimating means of k univariate Gaussians

Setting

- A dataset of $x_1, \dots, x_n$ examples is observed
- For each example x_i , cluster assignment is modelled as $z_{i1}, \dots, z_{ik}$ binary latent (i.e. unknown) variables
- $z_{ij} = 1$ if Gaussian j generated x_i , 0 otherwise.
- Parameters to be estimated are the $\mu_1, \dots, \mu_k$ Gaussians means
- All Gaussians are assumed to have the same (known) variance σ^2

Example: estimating means of k univariate Gaussians

Algorithm

- 1 Initialize $h = \langle \mu_1, \dots, \mu_k \rangle$
- 2 Iterate until difference in maximum likelihood (ML) is below a certain threshold:
 - E-step** Calculate expected value $E[z_{ij}]$ of each latent variable assuming current hypothesis $h = \langle \mu_1, \dots, \mu_k \rangle$ holds
 - M-step** Calculate a new ML hypothesis $h' = \langle \mu'_1, \dots, \mu'_k \rangle$ assuming values of latent variables are their expected values just computed. Replace $h \leftarrow h'$

Example: estimating means of k univariate Gaussians

Algorithm

E-step The expected value of z_{ij} is the probability that x_i is generated by Gaussian j assuming hypothesis $h = \langle \mu_1, \dots, \mu_k \rangle$ holds:

$$E[z_{ij}] = \frac{p(x_i | \mu_j)}{\sum_{l=1}^k p(x_i | \mu_l)} = \frac{\exp - \frac{1}{2\sigma^2} (x_i - \mu_j)^2}{\sum_{l=1}^k \exp - \frac{1}{2\sigma^2} (x_i - \mu_l)^2}$$

M-step The maximum-likelihood mean μ_j is the weighted sample mean, each instance being weighted by its probability of being generated by Gaussian j :

$$\mu'_j = \frac{\sum_{i=1}^n E[z_{ij}] x_i}{\sum_{i=1}^n E[z_{ij}]}$$

Expectation-Maximization (EM)

Formal setting

- We are given a dataset made of an observed part X and an unobserved part Z
- We wish to estimate the hypothesis maximizing the expected log-likelihood for the data, with expectation taken over unobserved data:

$$h^* = \operatorname{argmax}_h \mathbb{E}_Z[\ln p(X, Z|h)]$$

Problem

The unobserved data Z should be treated as random variables governed by the distribution depending on X and h

Expectation-Maximization (EM)

Generic algorithm

- 1 Initialize hypothesis h
- 2 Iterate until convergence

E-step Compute the expected likelihood of an hypothesis h' for the full data, where the unobserved data distribution is modelled according to the current hypothesis h and the observed data:

$$Q(h'; h) = \mathbb{E}_Z[\ln p(X, Z|h') | h, X]$$

M-step replace the current hypothesis with the one maximizing $Q(h'; h)$

$$h \leftarrow \operatorname{argmax}_{h'} Q(h'; h)$$

Example: estimating means of k univariate Gaussians

Derivation

- the likelihood of an example is:

$$p(x_i, z_{i1}, \dots, z_{ik} | h') = \frac{1}{\sqrt{2\pi}\sigma} \exp \left[- \sum_{j=1}^k z_{ij} \frac{(x_i - \mu'_j)^2}{2\sigma^2} \right]$$

- the dataset log-likelihood is:

$$\ln p(X, Z | h') = \sum_{i=1}^n \left(\ln \frac{1}{\sqrt{2\pi}\sigma} - \sum_{j=1}^k z_{ij} \frac{(x_i - \mu'_j)^2}{2\sigma^2} \right)$$

Example: estimating means of k univariate Gaussians

E-step

- the expected log-likelihood (remember linearity of the expectation operator):

$$\begin{aligned} E_Z[\ln p(X, Z|h')] &= E_Z \left[\sum_{i=1}^n \left(\ln \frac{1}{\sqrt{2\pi}\sigma} - \sum_{j=1}^k z_{ij} \frac{(x_i - \mu'_j)^2}{2\sigma^2} \right) \right] \\ &= \sum_{i=1}^n \left(\ln \frac{1}{\sqrt{2\pi}\sigma} - \sum_{j=1}^k E[z_{ij}] \frac{(x_i - \mu'_j)^2}{2\sigma^2} \right) \end{aligned}$$

- The expectation given current hypothesis h and observed data X is computed as:

$$E[z_{ij}] = \frac{p(x_i|\mu_j)}{\sum_{l=1}^k p(x_i|\mu_l)} = \frac{\exp -\frac{1}{2\sigma^2}(x_i - \mu_j)^2}{\sum_{l=1}^k \exp -\frac{1}{2\sigma^2}(x_i - \mu_l)^2}$$

Example: estimating means of k univariate Gaussians

M-step

- The likelihood maximization gives:

$$\begin{aligned}\operatorname{argmax}_{h'} Q(h'; h) &= \operatorname{argmax}_{h'} \sum_{i=1}^n \left(\ln \frac{1}{\sqrt{2\pi}\sigma} - \sum_{j=1}^k E[z_{ij}] \frac{(x_i - \mu'_j)^2}{2\sigma^2} \right) \\ &= \operatorname{argmin}_{h'} \sum_{i=1}^n \sum_{j=1}^k E[z_{ij}] (x_i - \mu'_j)^2\end{aligned}$$

- zeroing the derivative wrt to each mean we get:

$$\frac{\partial}{\partial \mu'_j} = -2 \sum_{i=1}^n E[z_{ij}] (x_i - \mu'_j) = 0$$

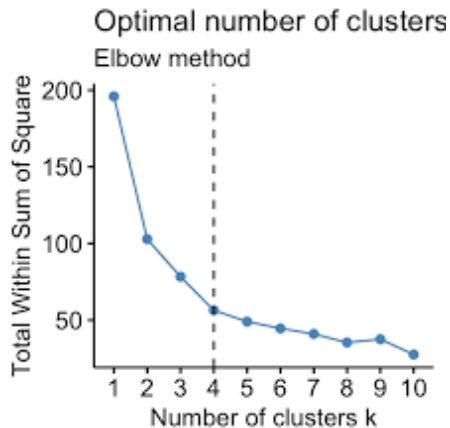
$$\mu'_j = \frac{\sum_{i=1}^n E[z_{ij}] x_i}{\sum_{i=1}^n E[z_{ij}]}$$

How to choose the number of clusters?

Elbow method: idea

- Increasing number of clusters allows for better modeling of data
- Needs to trade-off quality of clusters with quantity
- Stop increasing number of clusters when advantage is limited

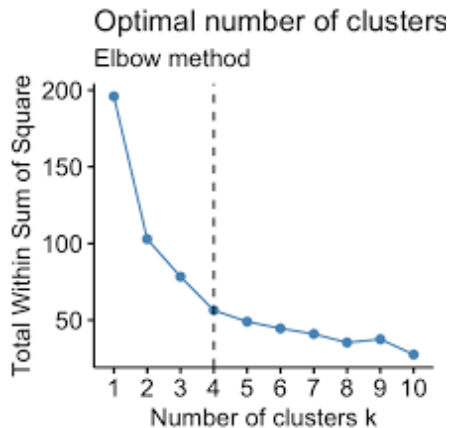
How to choose the number of clusters?



Elbow method: approach

- 1 Run the clustering algorithm for increasing values of k .
- 2 Plot a clustering evaluation metric, such as the within-cluster sum of squared errors, against k .
- 3 Choose the value of k near the “elbow,” where adding more clusters produces only a small additional improvement.

How to choose the number of clusters?



Elbow method: problem

The Elbow method can be ambiguous, with multiple candidate points (e.g. $k=2$ and $k=4$ in the figure).

How to choose the number of clusters?

Average silhouette method: idea

- Increasing the numbers of clusters makes **each cluster** more homogeneous
- Increasing the number of clusters can make **different clusters** more similar
- Use quality metric that trades-off intra-cluster similarity and inter-cluster dissimilarity

How to choose the number of clusters?

Silhouette coefficient for example i

- 1 Compute the average dissimilarity between i and examples of its cluster C :

$$a_i = d(i, C) = \frac{1}{|C|} \sum_{j \in C} d(i, j)$$

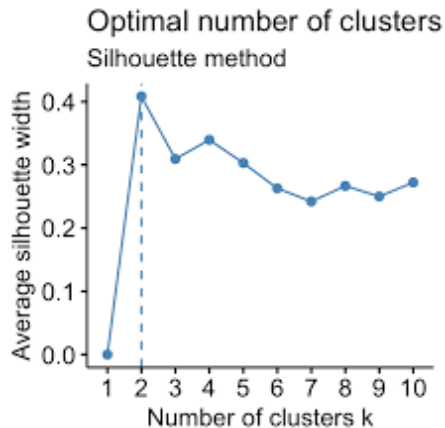
- 2 Compute the average dissimilarity between i and examples of each cluster $C' \neq C$, take the minimum:

$$b_i = \min_{C' \neq C} d(i, C')$$

- 3 The silhouette coefficient is:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

How to choose the number of clusters?



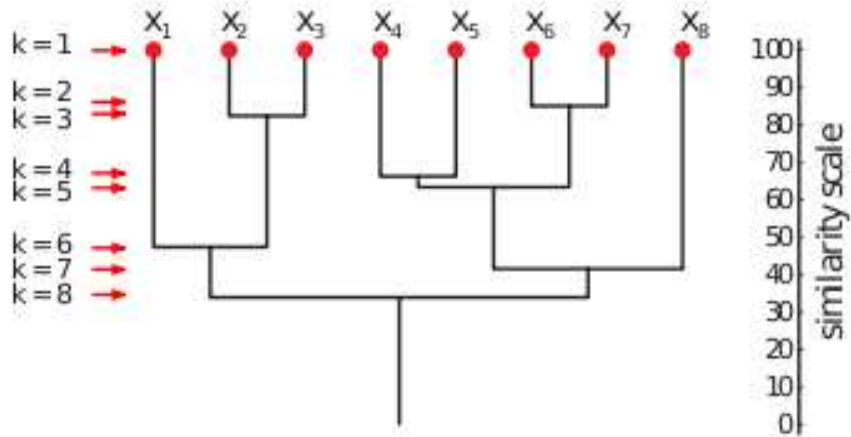
Average silhouette method: approach

- 1 Run clustering algorithm for increasing number of clusters
- 2 Plot average (over examples) silhouette coefficient for different k
- 3 Choose k where the average silhouette coefficient is maximal

Setting

- Clustering does not need to be *flat*
- Natural grouping of data is often hierarchical (e.g. biological taxonomy, topic taxonomy, etc.)
- A hierarchy of clusters can be built on examples
- *Top-down* approach:
 - start from a single cluster with all examples
 - recursively split clusters into subclusters
- *Bottom-up* approach:
 - start with n clusters of individual examples (singletons)
 - recursively aggregate pairs of clusters

Dendrograms



Agglomerative hierarchical clustering

Algorithm

- 1 Initialize:
 - Final cluster number k (e.g. $k=1$)
 - Initial cluster number $\hat{k} = n$
 - Initial clusters $\mathcal{D}_i = \{x_i\}, i \in 1, \dots, n$
- 2 while $\hat{k} > k$:
 - 1 find pairwise nearest clusters $\mathcal{D}_i, \mathcal{D}_j$
 - 2 merge $\mathcal{D}_i$ and $\mathcal{D}_j$
 - 3 update $\hat{k} = \hat{k} - 1$

Note

Stopping criterion can be threshold on pairwise similarity

Measuring cluster similarities

Similarity measures

- Nearest-neighbour

$$d_{min}(\mathcal{D}_i, \mathcal{D}_j) = \min_{\mathbf{x} \in \mathcal{D}_i, \mathbf{x}' \in \mathcal{D}_j} \|\mathbf{x} - \mathbf{x}'\|$$

- Farthest-neighbour

$$d_{max}(\mathcal{D}_i, \mathcal{D}_j) = \max_{\mathbf{x} \in \mathcal{D}_i, \mathbf{x}' \in \mathcal{D}_j} \|\mathbf{x} - \mathbf{x}'\|$$

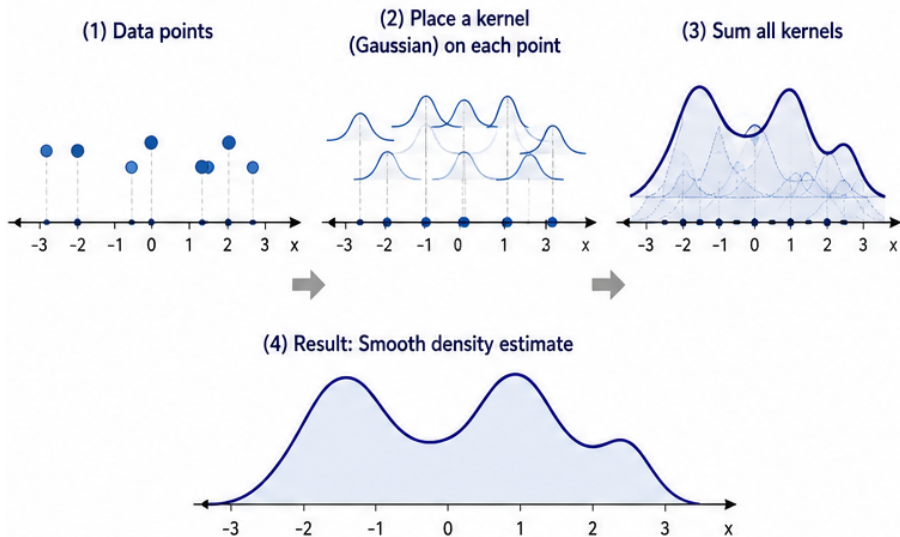
- Average distance

$$d_{avg}(\mathcal{D}_i, \mathcal{D}_j) = \frac{1}{n_i n_j} \sum_{\mathbf{x} \in \mathcal{D}_i} \sum_{\mathbf{x}' \in \mathcal{D}_j} \|\mathbf{x} - \mathbf{x}'\|$$

- Distance between means

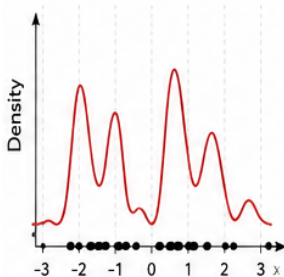
$$d_{mean}(\mathcal{D}_i, \mathcal{D}_j) = \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|$$

Kernel Density Estimation: From Points to Density



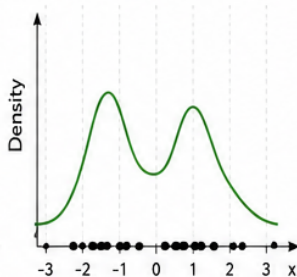
Bandwidth Selection

Small bandwidth
(h too small)



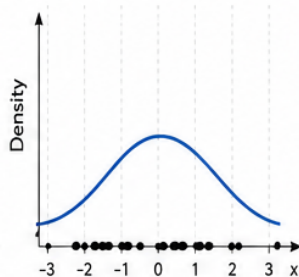
Overfits the data
Very wiggly

Good bandwidth
(just right)



Captures the structure
Good bias-variance trade-off

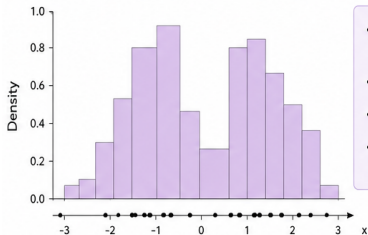
Large bandwidth
(h too large)



Oversmooths the data
Misses important details

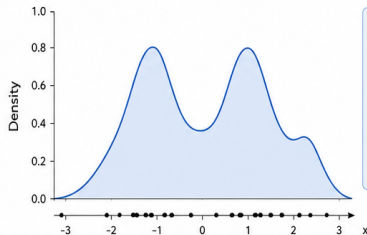
Histogram or KDE?

Histogram (depends on bins)



- Depends on bin locations
- Piecewise constant
- Discontinuous
- Sensitive to bin width

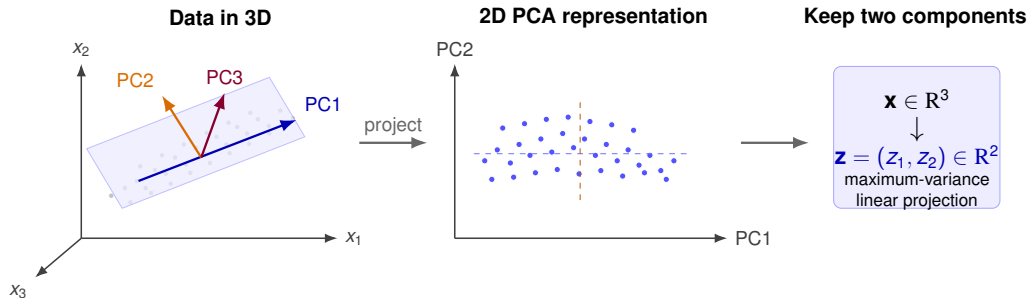
Kernel Density Estimate (KDE)



- Smooth estimate
- Continuous
- Often more informative
- Controlled by bandwidth (h)

Dimensionality Reduction: PCA

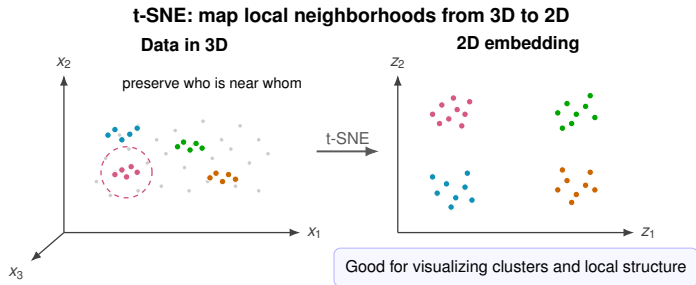
PCA: project 3D data onto a 2D principal subspace



Key idea

PCA finds the two-dimensional linear subspace that retains the most variance. Projecting the 3D data onto PC1 and PC2 gives an axis-aligned two-dimensional representation.

Dimensionality Reduction: t-SNE



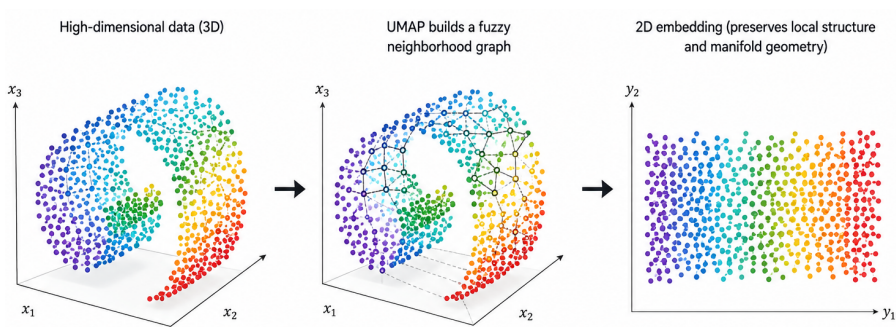
Key idea

t-SNE is designed for visualization: it preserves local neighborhoods, so nearby points in three dimensions remain nearby in two dimensions.

Caveat

Distances between well-separated clusters are often not meaningful.

Dimensionality Reduction: UMAP



Key idea

UMAP builds a neighborhood graph on the three-dimensional manifold and finds a two-dimensional embedding that preserves this graph structure.

- Often faster and more scalable than t-SNE.
- Preserves local neighborhoods and some global organization.

Dimensionality Reduction: Comparison

Method	PCA	t-SNE	UMAP
Type	Linear	Non-linear	Non-linear
What it preserves	Global variance (Linear structure)	Local structure (Neighborhoods)	Local + some global (Manifold structure)
Captures clusters	Sometimes (if linear)	Excellent	Very good
Speed	Very fast	Slow	Fast
Scalability	Excellent	Limited	Excellent
Out-of-sample extension	Yes	No	Yes
Best use	Preprocessing, noise reduction, interpretability	Visualization of clusters and local relationships	Visualization of complex manifolds, large datasets, balanced choice

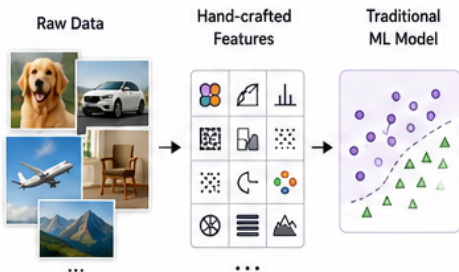


Takeaway

Use PCA for linear structure and preprocessing, t-SNE for discovering clusters and local structure, and UMAP for scalable visualization that preserves both local and global patterns.

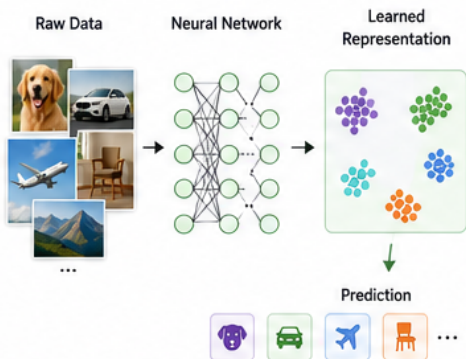
From Features to Representations

Traditional Machine Learning



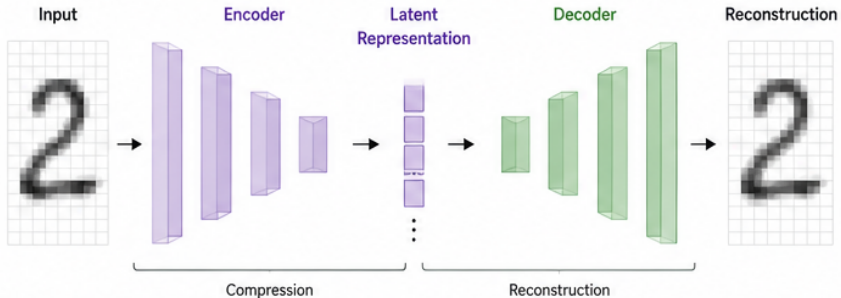
- Features designed by humans
- Limited flexibility
- May not generalize well

Modern Representation Learning



- Learns useful representations from data
- More flexible
- Better generalization

Autoencoders

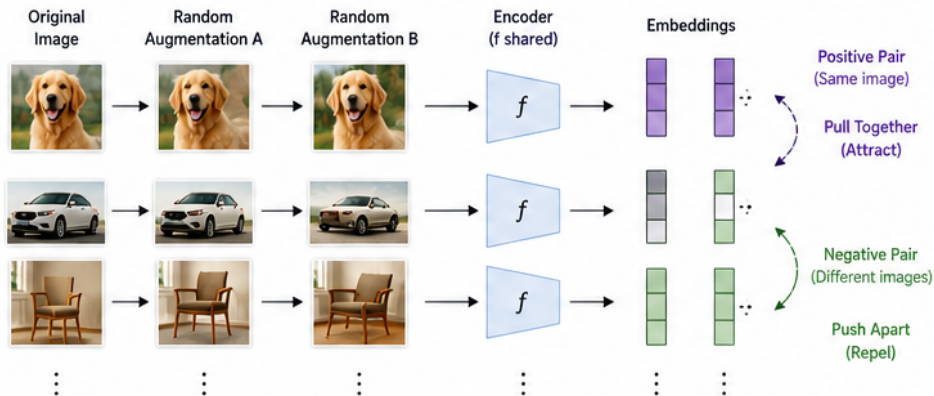


Representation learning

The network is trained to reconstruct its input $x \rightarrow \hat{x}$ while compressing the information into a low-dimensional latent representation.

- Encoder learns compact features
- Decoder reconstructs the input
- No labels are required

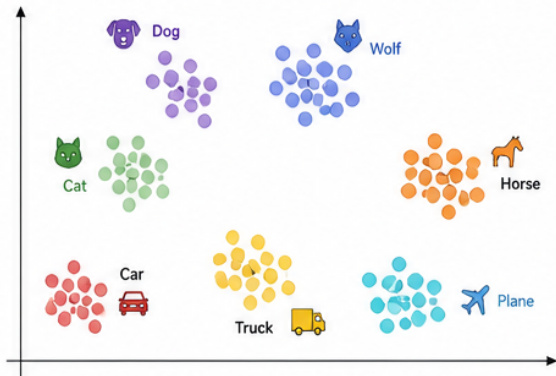
Contrastive Learning



Learning by comparison

Contrastive learning discovers an embedding space in which similar examples are close together and different examples are far apart.

Embeddings



Applications

-  Nearest neighbor search
-  Recommendation
-  Retrieval
-  LLMs
-  Semantic search



Embedding

An embedding is a continuous vector representation in which similar objects are mapped to nearby points.