Tutorial: Application of Deep Clustering Algorithms

32nd ACM International Conference on Information and Knowledge Management

Collin Leiber[1,2], Lukas Miklautz[3,4], Claudia Plant[3,5], Christian Böhm[3]

[1]Institute of Informatics, Ludwig-Maximilians-Universität München, Munich, Germany
[2]MCML
[3]Faculty of Computer Science, University of Vienna, Vienna, Austria
[4]UniVie Doctoral School Computer Science, Vienna, Austria
[5]ds:UniVie, Vienna, Austria

Presenters

Collin Leiber

Lukas Miklautz

Claudia Plant

Christian Böhm

Hands-On

- Prepared jupyter notebook with examples
- Implemented in PyTorch and ClustPy
- Collab link for jupyter notebook: https://tinyurl.com/cikm23-clustpy
- Download link for material: https://tinyurl.com/cikm23-material

ClustPy Package

- Link: https://github.com/collinleiber/ClustPy

- > 20 recently introduced (deep) clustering algorithms implemented in sklearn style → Easy to use and apply
- > 70 benchmarking data sets (e.g., UCI, UCR, Torchvision, MedicalMNIST)
- Many performance metrics and visualization methods

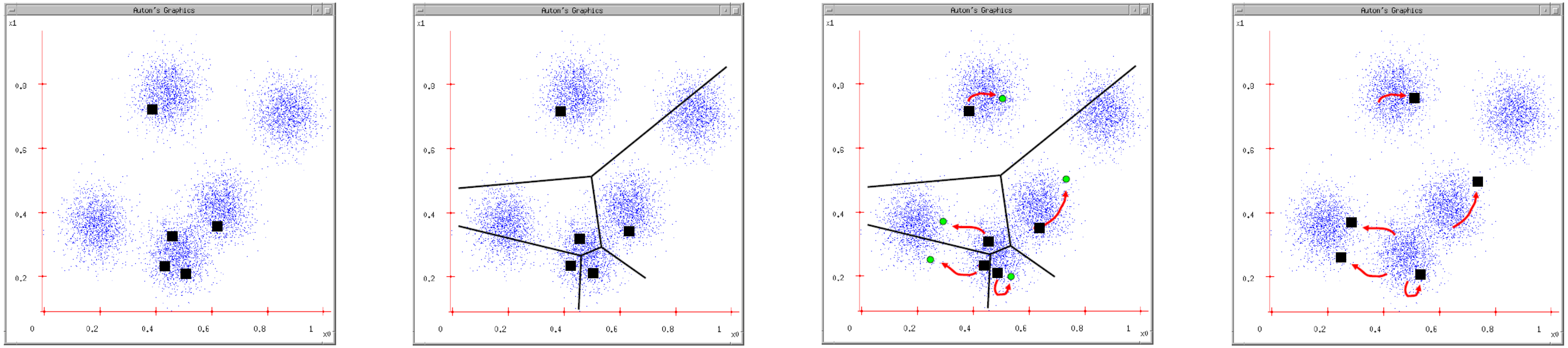
ClustPy

Outline

- Introduction to Clustering
- Introduction to Deep Clustering
- Application of Deep Clustering Algorithms
- Recent Approaches
- Outlook

Clustering – Find a “meaningful” grouping

Recap: K-Means



1. random **initialization** of the K cluster centers

2. **assignment** of objects to the closest center

3. **update** of the centers

4. **iteration** of (2) and (3) until convergence

+ fast convergence,
+ well-defined objective function,
+ based on statistical model.

Recap: K-Means

Algorithm k-Means

Input Parameter: Number K of clusters;

Randomly initialize the K cluster centers $\mu_1 \ldots \mu_K$

Iterate the following steps until convergence:

Assign each object x_i to the nearest centroid μ_j

Update the cluster centroids $\mu = (\mu_1 \ldots \mu_K)$

Objective function:

$$L(\mu; x) = \sum_i L(\mu; x_i) = \sum_i \frac{1}{2}\big(x_i - d_i(\mu)\big)^2$$

Where the function $j \coloneqq d_i(\mu)$ assigns the i^{th} point x_i to its closest centroid μ_j

SGD-K-Means

- Stochastic Gradient Descent Version of K-Means [BB94]
  - Learned parameters for K-Means are the centroids $\mu_j, j \in \{0,1,\dots,\mathrm{K}\}$
  - Runs several times (epochs) over the full data set in randomized order

$$L(\mu; x_i) = \frac{1}{2}\left(x_i - d_i(\mu)\right)^2$$

SGD-K-Means

- Stochastic Gradient Descent Version of K-Means [BB94]
  - Learned parameters for K-Means are the centroids $\mu_j, j \in \{0,1,\dots,\mathrm{K}\}$
  - Runs several times (epochs) over the full data set in randomized order

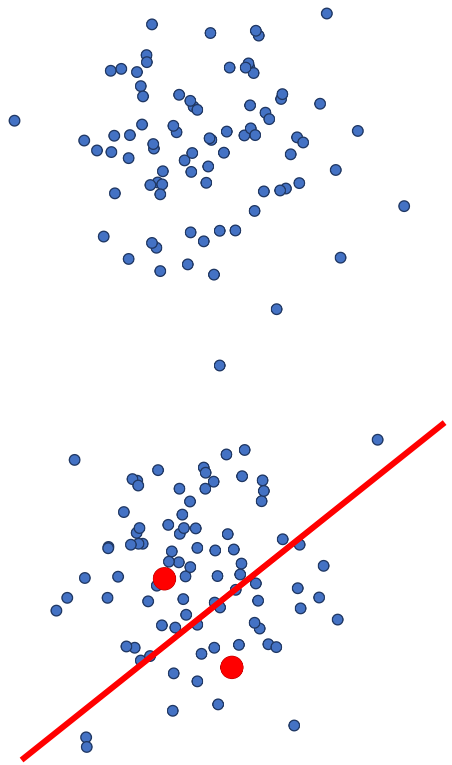
$$L(\mu; x_i) = \frac{1}{2}\left(x_i - d_i(\mu)\right)^2$$

  - The gradient update for the loss function w.r.t. μ

$$\Delta\mu = -\alpha \cdot \frac{\partial L(\mu; x_i)}{\partial \mu} = \begin{cases} \alpha \cdot \left(x_i - \mu_j\right), & \text{if } j = d_i(\mu) \\ 0, & \text{otherwise} \end{cases}$$

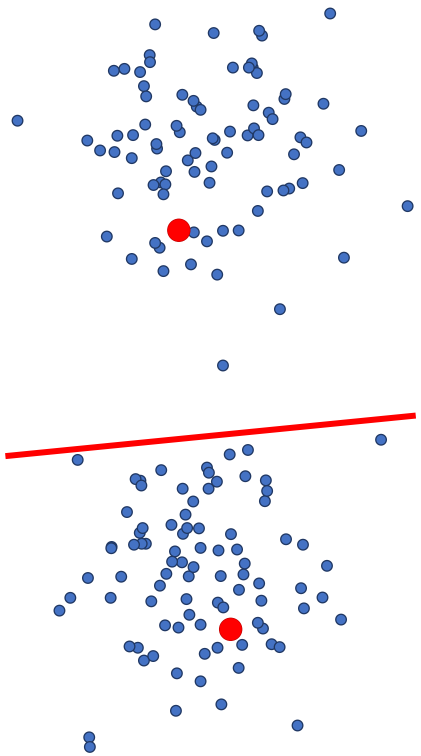
- Each point x_i moves its respective center μ_j closer to x_i by $\Delta\mu$
- Optimal learning rate $\alpha = 1/n_j$ where n_j is number of objects in cluster j

SGD-K-Means converges much faster



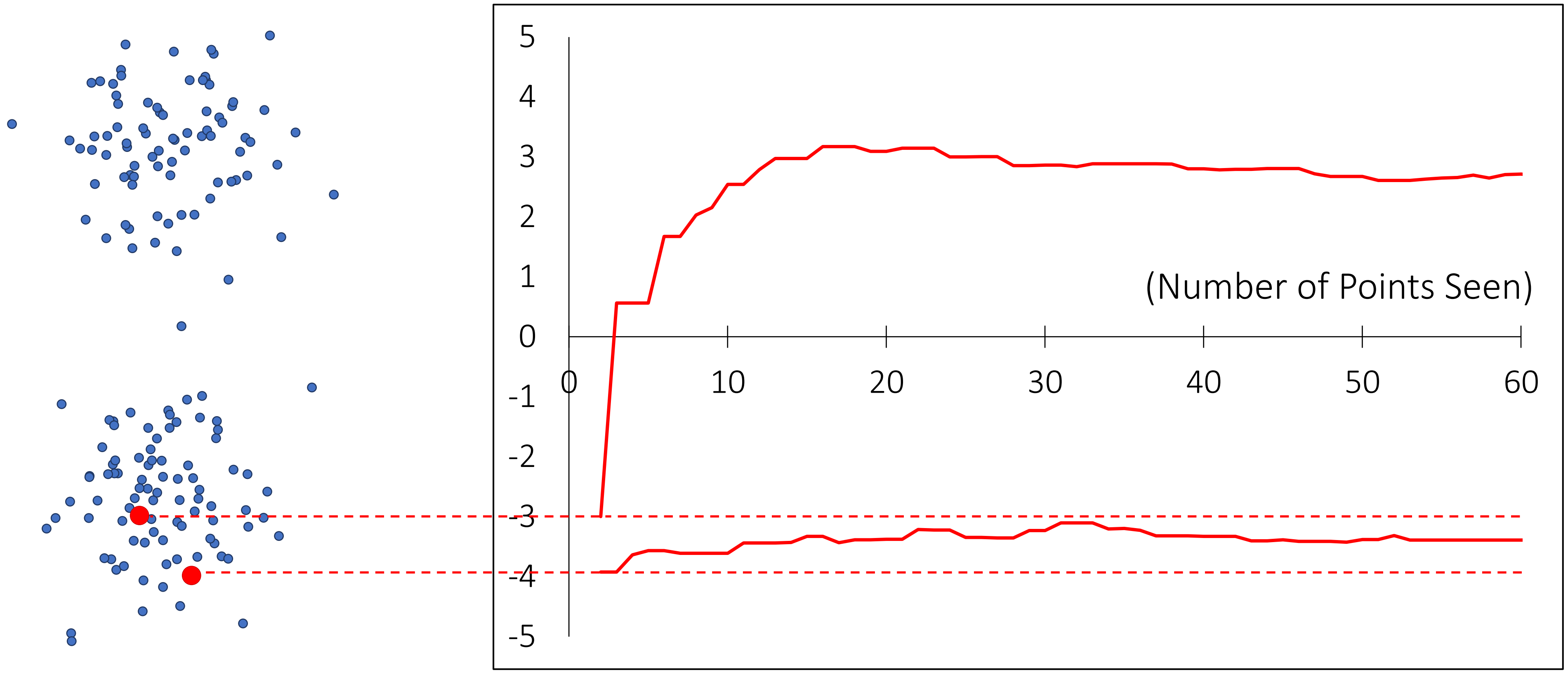
- The random initialization may go wrong
- Classical K-Means would base a complete round of assignment on the resulting boundary

SGD-K-Means converges much faster



- The random initialization may go wrong
- Classical K-Means would base a complete round of assignment on the resulting boundary
- After having seen e.g. 10 points, the centers are already much better with SGD-K-Means
- SGD-K-Means continuously improves centers

SGD-K-Means converges much faster



Minibatch-K-Means

Algorithm Minibatch-K-Means [S10]

Input Parameter: Number K of clusters;

Randomly initialize the K cluster centers $\mu_1 \dots \mu_K$

Iterate the following steps until convergence:

Select a Minibatch M;

Update centroids $\mu_1 \dots \mu_K$ for each x_i in M:

$$\Delta\mu = -\alpha \cdot \frac{\partial L(\mu; x_i)}{\partial \mu} = \begin{cases} \alpha \cdot \left(x_i - \mu_j\right), & \text{if } j = d_i(\mu) \\ 0, & \text{otherwise} \end{cases}$$

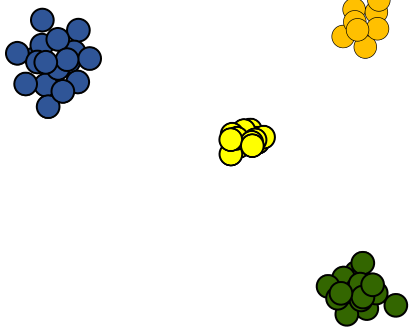
Further improvements, e.g. in [PB10, APB13]:

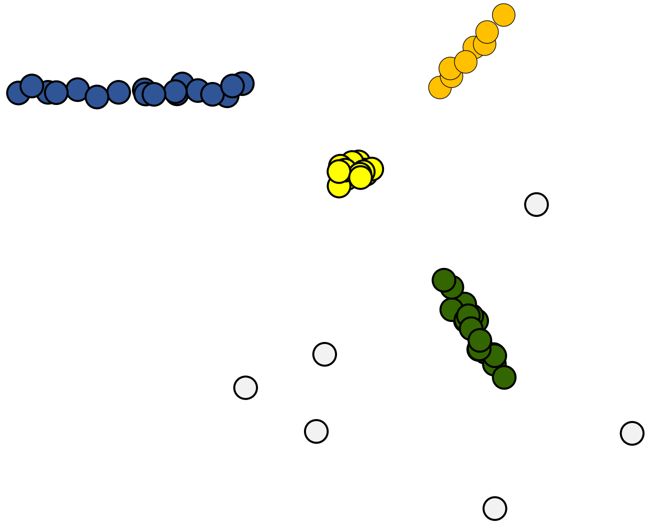
- Consider additional update of center μ whenever the cluster loses a point x_i
- Consider occupation of network/bus when parallel processes exchange information of centers $\mu_1 \dots \mu_K$

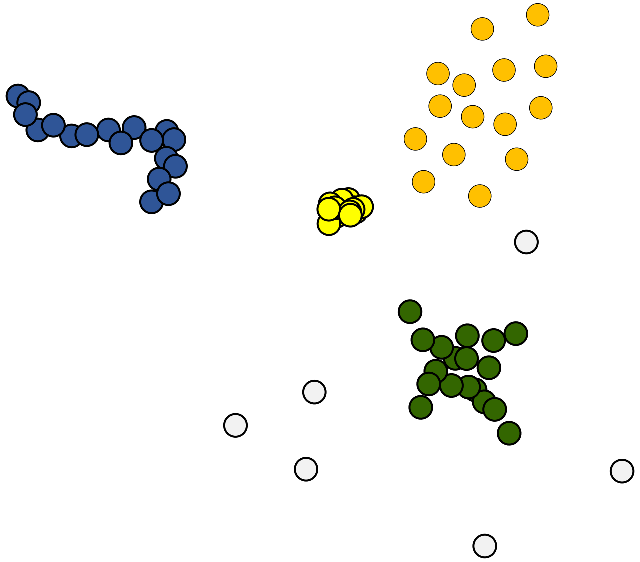
Outline

- Introduction to Clustering
- Introduction to Deep Clustering
- Application of Deep Clustering Algorithms
- Recent Approaches
- Outlook

The Curse of Dimensionality in Clustering







- Full-dimensional
- Gaussian clusters
- without outliers or noise.

- Subspace clusters
- and outliers.

- Arbitrarily shaped subspace clusters,
- of different density,
- noise and outliers.

Deep Representation Learning

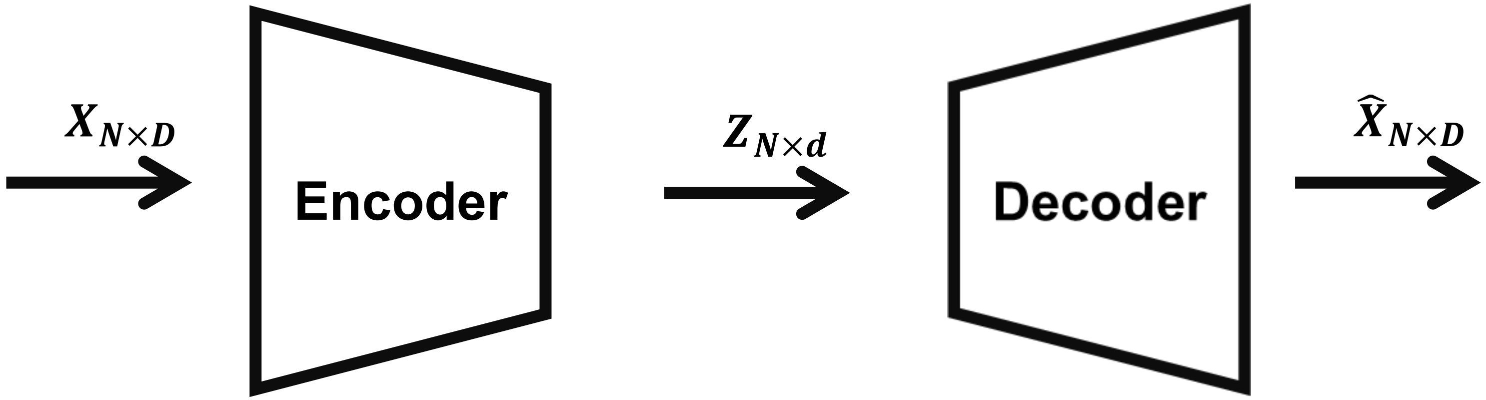
- Successful for image, text, video, audio …
  - Structured data
  - High data volume

- Automated feature extraction (Representation Learning)
  - Feature engineering requires domain knowledge

- Easy to parallelize
  - GPU friendly
  - Works on large amount of data

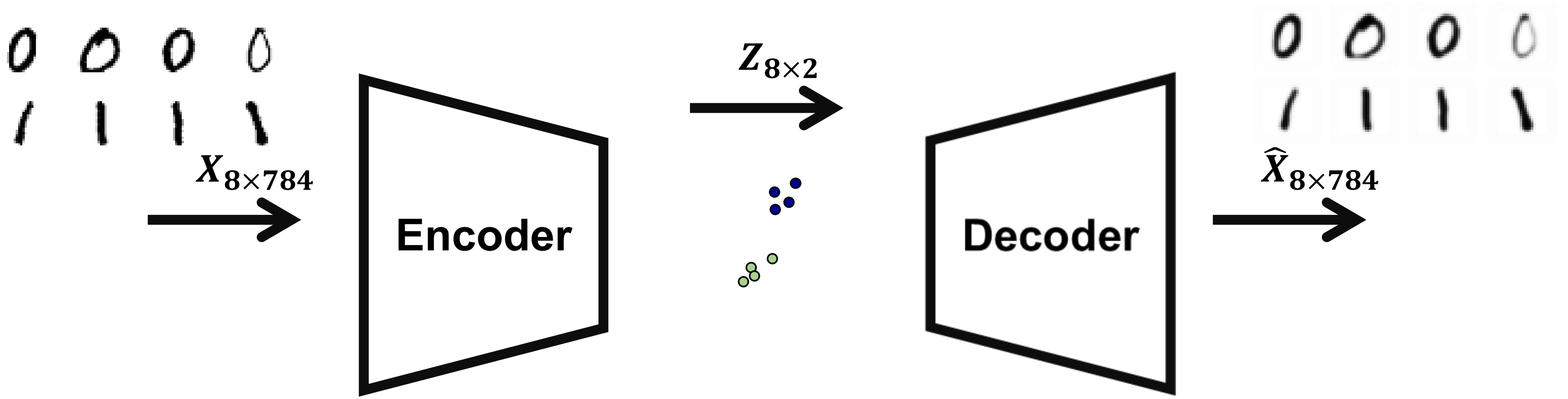
Prerequisite: Autoencoder

- Learning is done via self-supervision – requires no labels
- The prediction (output) is a reconstruction of the input data
- Goal: Low dimensional representation (embedding) of input data



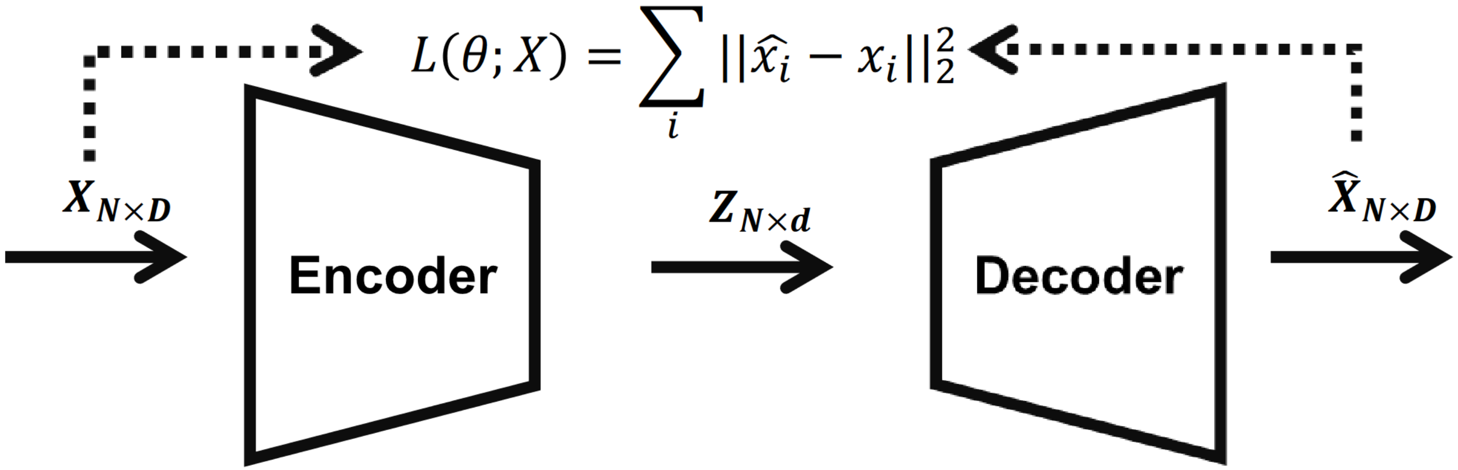
Prerequisite: Autoencoder

- Learning is done via self-supervision – requires no labels
- The prediction (output) is a reconstruction of the input data
- Goal: Low dimensional representation (embedding) of input data



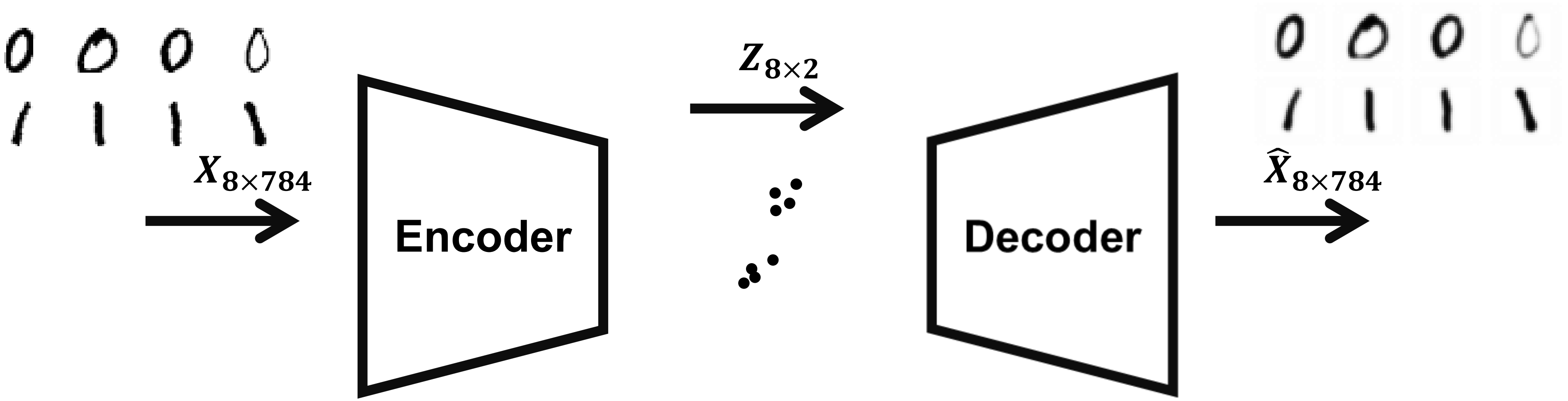
Autoencoder – Loss Function

- Compares the reconstruction $\hat{x}$ with the input x
- Quantifies the reconstruction loss which we want to minimize
- Common choices: Cross Entropy, Sum of Squared Differences



Sequential Deep Clustering Approach

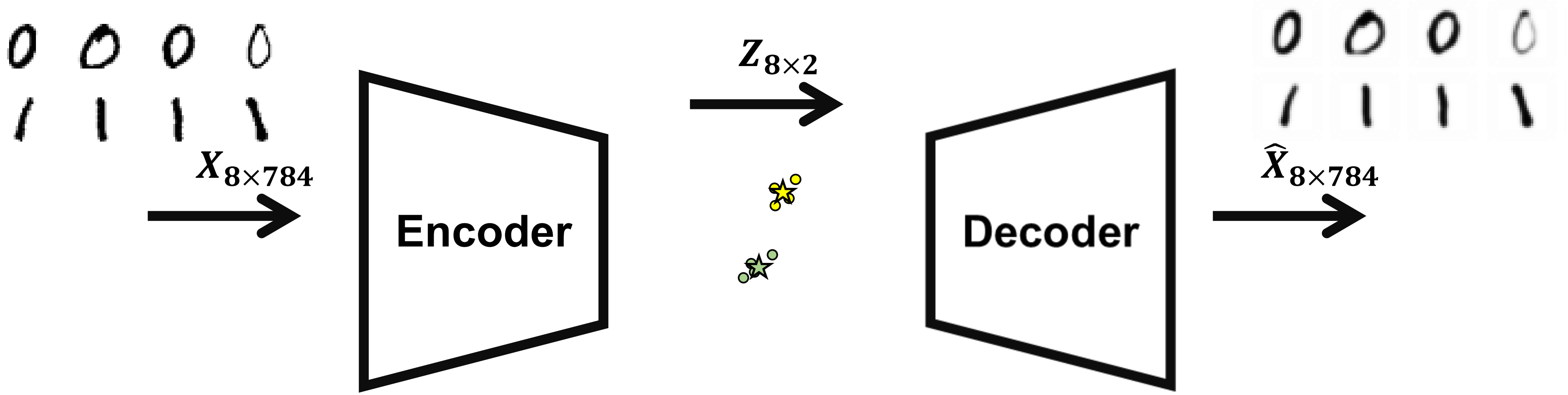
1) Use an autoencoder to learn a non-linear embedding of your data i.e., Feature learning/Representation learning



Sequential Deep Clustering Approach

1) Use an autoencoder to learn a non-linear embedding of your data i.e., Feature learning/Representation learning

2) Cluster that data with some algorithm of your choice



Sequential Deep Clustering Approach

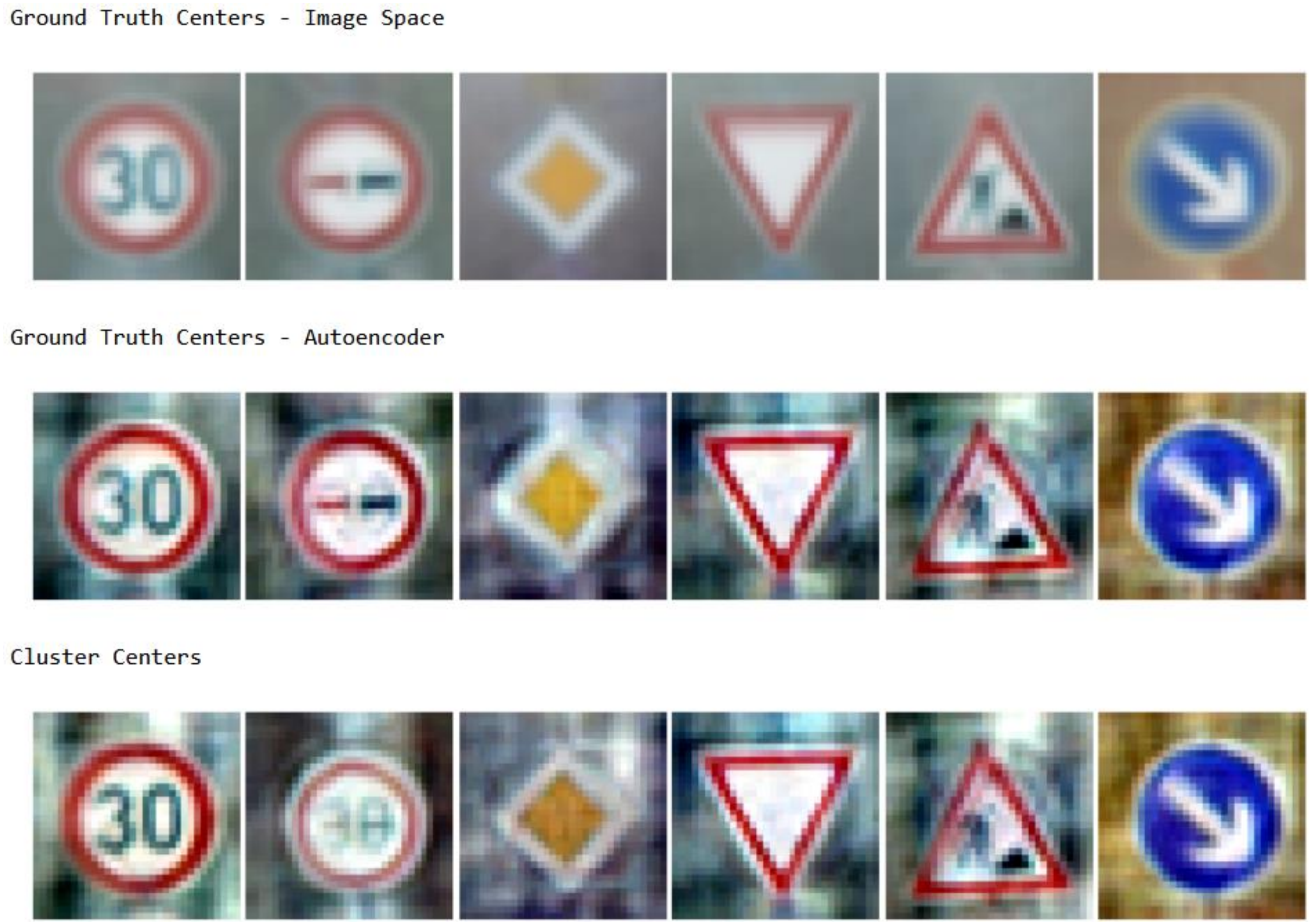
1) Use an autoencoder to learn a non-linear embedding of your data i.e., Feature learning/Representation learning

2) Cluster that data with some algorithm of your choice

Note: This is not necessarily a bad idea and often useful, but it might limit our solution -> We are stuck to the initial representation

Notebook Example

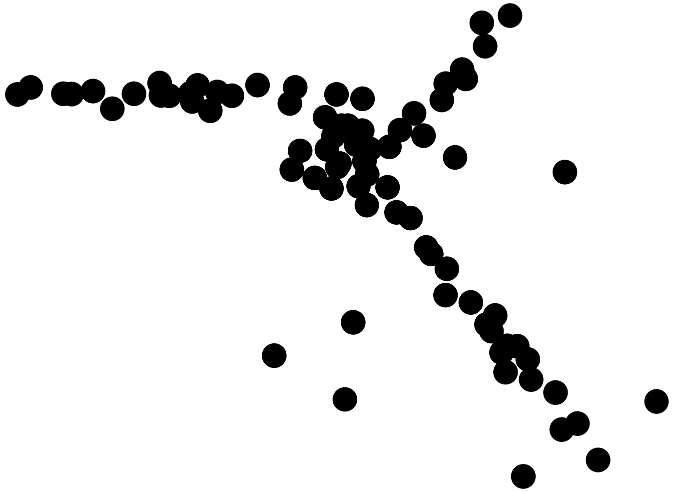
- Clustering of Autoencoder embedded space



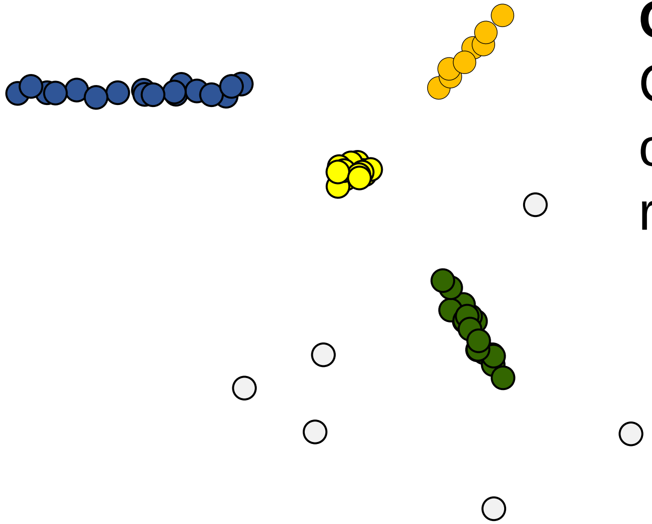
Can we do better?

Yes! – Learn A Cluster Friendly Representation

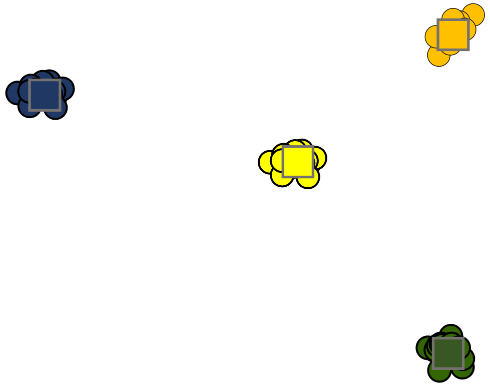
Initial representation



Deep Clustering



Option A:
Correlation and outlier preserving representation



Option B:
k-Means friendly representation

Deep Clustering - Overview

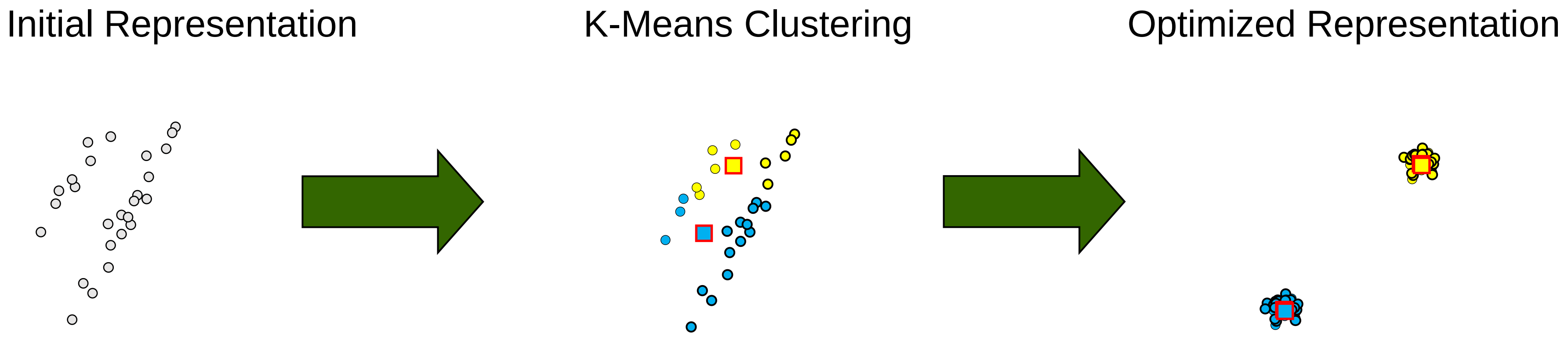
- Idea: Include the notion of clustering already during the autoencoder training
- Goal: We want to find all relevant cluster structure and improve it!

Problems:

- We need to specify a cluster model (inherit assumptions)
- We face circular dependency problem
  - In order to learn a good representation we need to know what clusters we have
  - In order to learn a good clustering we need to have already a good representation
  - Deep Learning is not a magic bullet that solves this problem

Deep Clustering – Toy Example

- **Problems**: We still face circular dependency problem
  - In order to learn a good representation we need to know what clusters we have
  - In order to learn a good clustering we need to have already a good representation
- Here: Clusters are ripped apart



Deep Clustering - Approaches

- Alternating optimization
  - Alternate between optimizing the representation and updating the clustering assignments
- Joint optimization
  - Cluster assignments and representation are updated together

Deep Clustering - Approaches

- Alternating optimization
  - Alternate between optimizing the representation and updating the clustering assignments
- Joint optimization
  - Cluster assignments and representation are updated together

- Overall Goal: Learn a cluster friendly embedding
  - Cluster friendly = Enhanced separation of clusters, Cluster structure is more distinct
  - Increase inter-cluster distance and decrease intra-cluster distance
  - Include structural constraints to avoid the “destruction” of structure, i.e. ripped apart clusters

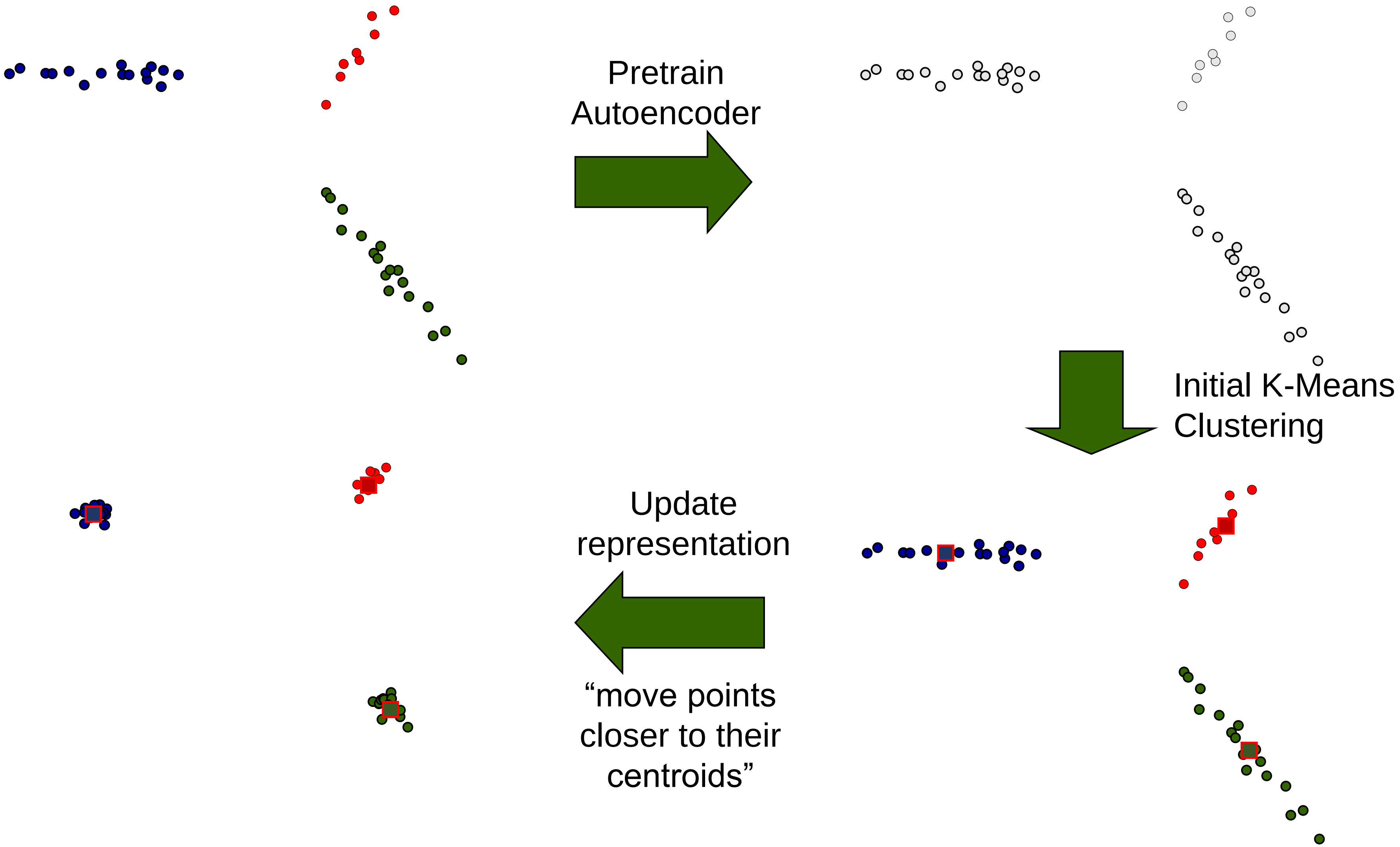
Outline

- Introduction to Clustering
- Introduction to Deep Clustering
- Application of Deep Clustering Algorithms
- Recent Approaches
- Outlook

Alternating Optimization

1) Pretrain an autoencoder to learn a non-linear embedding of your data
   a) Set the dimensionality to $\min(k, \#\ \text{features})$. This “upper bound” avoids losing too much information. For a motivation of this rule of thumb see e.g. the connections of K-means and PCA [DH04]
2) Initialize clustering with some algorithm (e.g. K-means)

Toy Example – 1 Iteration



Alternating Optimization

1) Pretrain an autoencoder to learn a non-linear embedding of your data
   a) Set the dimensionality to $\min(k, \#\text{ features})$. This "upper bound" avoids losing too much information. For a motivation of this rule of thumb see e.g. the connections of K-means and PCA [DH04]
2) Initialize clustering with some algorithm (e.g. K-means)

While cluster labels change:

a) Fix centroids and update the autoencoder parameters
   - Move points closer to their centroids
b) Fix autoencoder parameters and update centroids and assignments

DCN-Deep Clustering Networks

- Deep Clustering Network (DCN) [YFSH17]
  - Based on Mini-Batch K-means [S10]
  - Centroids are not optimized via SGD, but are updated explicitly
  - They use hard cluster assignments which are not differentiable

DCN-Deep Clustering Networks

- Deep Clustering Network (DCN) [YFSH17]
  - Based on Mini-Batch K-means [S10]
  - Centroids are not optimized via SGD, but are updated explicitly
  - They use hard cluster assignments which are not differentiable
- Alternating optimization between clustering and autoencoder
  - Because the calculation of cluster assignments is non-differentiable
  - Alternate between
    1) K-Means Step
       1) Assignments
       2) Centroid updates
    2) Autoencoder Step

       Preserve Global structure via Reconstruction and make clusters more “K-Means friendly” [YFSH17] by “moving” points closer to their centroids

DCN-Deep Clustering Networks

- Deep Clustering Network (DCN) [YFSH17]
- Alternating optimization between clustering and autoencoder
  - Alternate between
    1) K-Means Step
    2) Autoencoder Step (Reconstruction + Compression)

Overall Loss Function $l = \lambda l_c + l_R$

Compression loss: $l_C = ||z_i - \mu_i||_2^2$

Reconstruction loss: $l_R = ||\widehat{x_i} - x_i||_2^2$

where λ is a hyperparameter weighing the importance of cluster structure

Notebook Example

- Deep clustering with DCN

Joint Optimization

1) Pretrain an autoencoder to learn a non-linear embedding of your data
   a) Set the dimensionality to $\min(k, \# \text{ features})$.

2) Initialize clustering with some algorithm (here K-means)

Joint Optimization

1) Pretrain an autoencoder to learn a non-linear embedding of your data
   a) Set the dimensionality to $\min(k, \#\ \text{features})$.

2) Initialize clustering with some algorithm (here K-means)

3) While cluster labels change

   Jointly optimize the clustering parameters (update centroids and assignments), together with the autoencoder

Joint Optimization

1) Pretrain an autoencoder to learn a non-linear embedding of your data
   a) Set the dimensionality to $\min(k, \#\ \mathrm{features})$.
2) Initialize clustering with some algorithm (here K-means)
3) While cluster labels change

   Jointly optimize the clustering parameters (update centroids and assignments), together with the autoencoder

- Cluster procedure need to be differentiable
- Assignments need to be soft e.g. assignment probabilities
- Usually faster, because we can completely parallelize the procedure

DKM -Deep k-Means

- Deep k-Means (DKM) [FTG19]
- Truly joint learning of the representation and the k-Means clustering parameters
- Builds directly on the k-Means loss:

$$\sum_{x \in X} ||x - c(x, M)||_2^2, \quad \text{where} \quad c(x, M) = \underset{\mu \in M}{\operatorname{argmin}} ||x - \mu||_2^2$$

=> For Deep Clustering: $\mathcal{L} = \sum_{x \in X} \mathcal{L}_{rec}(x) + \lambda \underbrace{||\text{enc}(x) - c(\text{enc}(x), M)||_2^2}_{f}$

- Problem: *f* must be continuously differentiable!

DKM -Deep k-Means

- We need a function $G_k(\mathrm{x}, \alpha, M) = \begin{cases} 1 & \text{if } \mu_k = c(\mathrm{enc}(c), M) \\ 0 & \text{otherwise} \end{cases}$
- This would lead to:

$$\mathcal{L} = \sum_{x \in X} \mathcal{L}_{rec}(x) + \lambda \sum_{k}^{|M|} ||\mathrm{enc}(x) - \mu_k||_2^2 G_k(\mathrm{x}, \alpha, M)$$

DKM -Deep k-Means

- We need a function $G_k(\mathrm{x}, \alpha, M) = \begin{cases} 1 & \text{if } \mu_k = c(\mathrm{enc}(c), M) \\ 0 & \text{otherwise} \end{cases}$
- This would lead to:

$$\mathcal{L} = \sum_{x \in X} \mathcal{L}_{rec}(x) + \lambda \sum_{k}^{|M|} ||\mathrm{enc}(x) - \mu_k||_2^2 G_k(\mathrm{x}, \alpha, M)$$

- Use a parameterized softmax function

$$G_k(\mathrm{x}, \alpha, M) = \frac{e^{-\alpha||\mathrm{enc}(x) - \mu_k||_2^2}}{\sum_{k'}^{|M|} e^{-\alpha||\mathrm{enc}(x) - \mu_{k'}||_2^2}}$$

- Formulation is fully differentiable regarding the parameters of the autoencoder and the cluster centers M

DKM -Deep k-Means

- $\mathcal{L} = \sum_{x \in X} \mathcal{L}_{rec}(x) + \lambda \sum_{k}^{|M|} ||\text{enc}(x) - \mu_k||_2^2 \frac{e^{-\boxed{\alpha} ||\text{enc}(x) - \mu_k||_2^2}}{\sum_{k'}^{|M|} e^{-\boxed{\alpha} ||\text{enc}(x) - \mu_{k'}||_2^2}}$

- For α close to 0 all centroids are equally weighted, for very large α it simulates hard cluster assignments
- How to choose a good value for α?
  1. Possibility
     - Pretrain the autoencoder
     - Start clustering process with a large α (e.g., 1000)
  2. Possibility
     - Do not use pretraining
     - Use an annealing strategy for α.
       Start with small values and increase α after a certain amount of epochs

Notebook Example

- Deep clustering with DKM

Coffee Break

Welcome back. Any questions?

DEC - Deep Embedded Clustering

- Deep Embedded Clustering (DEC) [XGF16]
  - Based on SGD-K-means with a student t-kernel for measuring the distance of an embedded data point z_i to centroid μ_j in relation to its distance to all other centroids $\mu_{j'}$, except μ_j:

$$q_{i,j} = \frac{\left(1 + ||z_i - \mu_j||_2^2\right)^{-1}}{\sum_{j'}\left(1 + ||z_i - \mu_{j'}||_2^2\right)^{-1}} = \frac{distance\ to\ \mu_j}{summed\ distance\ to\ all\ other\ centroids}$$

- $q_{i,j}$ are soft assignments of the i^{th} data point to the j^{th} cluster centroid

DEC - Deep Embedded Clustering

- $Q_{N \times K}$ is the matrix of soft assignments $q_{i,j}$ of N data points to the K centroids
  - Achieved by measuring the distance with the Student's t-kernel between all embedded points z_i and centroids μ_i.

DEC - Deep Embedded Clustering

- $Q_{N\times K}$ is the matrix of soft assignments $q_{i,j}$ of N data points to the K centroids
  - Achieved by measuring the distance with the Student's t-kernel between all embedded points z_i and centroids μ_i.
- Target distribution $P_{N\times K}$:

  [XGF16] define the following desirable properties for the target distribution P:
  - strengthen predictions on data points assigned with high confidence
  - normalize loss contribution for each centroid

$$p_{i,j} = \frac{\frac{q_{i,j}^2}{f_j}}{\sum_{j'} \frac{q_{i,j'}^2}{f_{j'}}}$$

DEC - Deep Embedded Clustering

- $Q_{N \times K}$ is the matrix of soft assignments $q_{i,j}$ of N data points to the K centroids
  - Achieved by measuring the distance with the Student's t-kernel between all embedded points z_i and centroids μ_i.
- Target distribution $P_{N \times K}$:

  [XGF16] define the following desirable properties for the target distribution P:
  - strengthen predictions on data points assigned with high confidence
  - normalize loss contribution for each centroid
  - $\boxed{q_{i,j}^2}$ strengthens high confidence predictions

  → Assignments close to one will be kept higher than undecided ones that are close to 0.5

$$p_{i,j} = \frac{\frac{\boxed{q_{i,j}^2}}{f_j}}{\sum_{j'} \frac{q_{i,j'}^2}{f_{j'}}}$$

DEC - Deep Embedded Clustering

- $Q_{N \times K}$ is the matrix of soft assignments $q_{i,j}$ of N data points to the K centroids
    - Achieved by measuring the distance with the Student's t-kernel between all embedded points z_i and centroids μ_i.
- Target distribution $P_{N \times K}$:

  [XGF16] define the following desirable properties for the target distribution P:
    - strengthen predictions on data points assigned with high confidence
    - normalize loss contribution for each centroid
    - $q_{i,j}^2$ strengthens high confidence predictions
    - $\boxed{f_j} := \sum_i q_{i,j}$ (soft) frequency per cluster

  → Dividing by $\boxed{f_j}$ renormalizes by cluster size to avoid that large clusters distort the embedding

$$p_{i,j} = \frac{\frac{q_{i,j}^2}{\boxed{f_j}}}{\sum_{j'} \frac{q_{i,j'}^2}{f_{j'}}}$$

DEC - Deep Embedded Clustering

- Minimize the KL divergence between the target distribution P and the cluster assignment Matrix Q:

$$l = l_C = KL(P||Q) = \sum_i \sum_j p_{i,j} \log\left(\frac{p_{i,j}}{q_{i,j}}\right)$$

- Measures how closely the assignment matrix Q matches the target distribution P

DEC - Deep Embedded Clustering

- Minimize the KL divergence between the target distribution P and the cluster assignment Matrix Q:

$$l = l_C = KL(P||Q) = \sum_i \sum_j p_{i,j} \log\left(\frac{p_{i,j}}{q_{i,j}}\right)$$

- Measures how closely the assignment matrix Q matches the target distribution P
- Overall Intuition – Increase separation of clusters by moving embedded points closer to their centroids μ_i and repelling points from other centroids $\mu_j, j \neq i$.
- Note that DEC does not use the reconstruction loss l_R during the joint optimization process

Notebook Example

- Deep clustering with DEC

Ground Truth Centers - Image Space

Ground Truth Centers - Autoencoder

Cluster Centers

IDEC-Improved Deep Embedded Clustering

- [GGLY17] proposed to keep the reconstruction loss during joint optimization with DEC to avoid such distorted solutions
- Their approach IDEC uses during the joint optimization both losses

Overall loss function $l = l_R + \lambda l_c$

Compression loss: $l_C = KL(P||Q)$

Reconstruction loss: $l_R = ||\widehat{x_i} - x_i||_2^2$

IDEC-Improved Deep Embedded Clustering

- [GGLY17] proposed to keep the reconstruction loss during joint optimization with DEC to avoid such distorted solutions
- Their approach IDEC uses during the joint optimization both losses

Overall loss function $l = l_R + \lambda l_c$

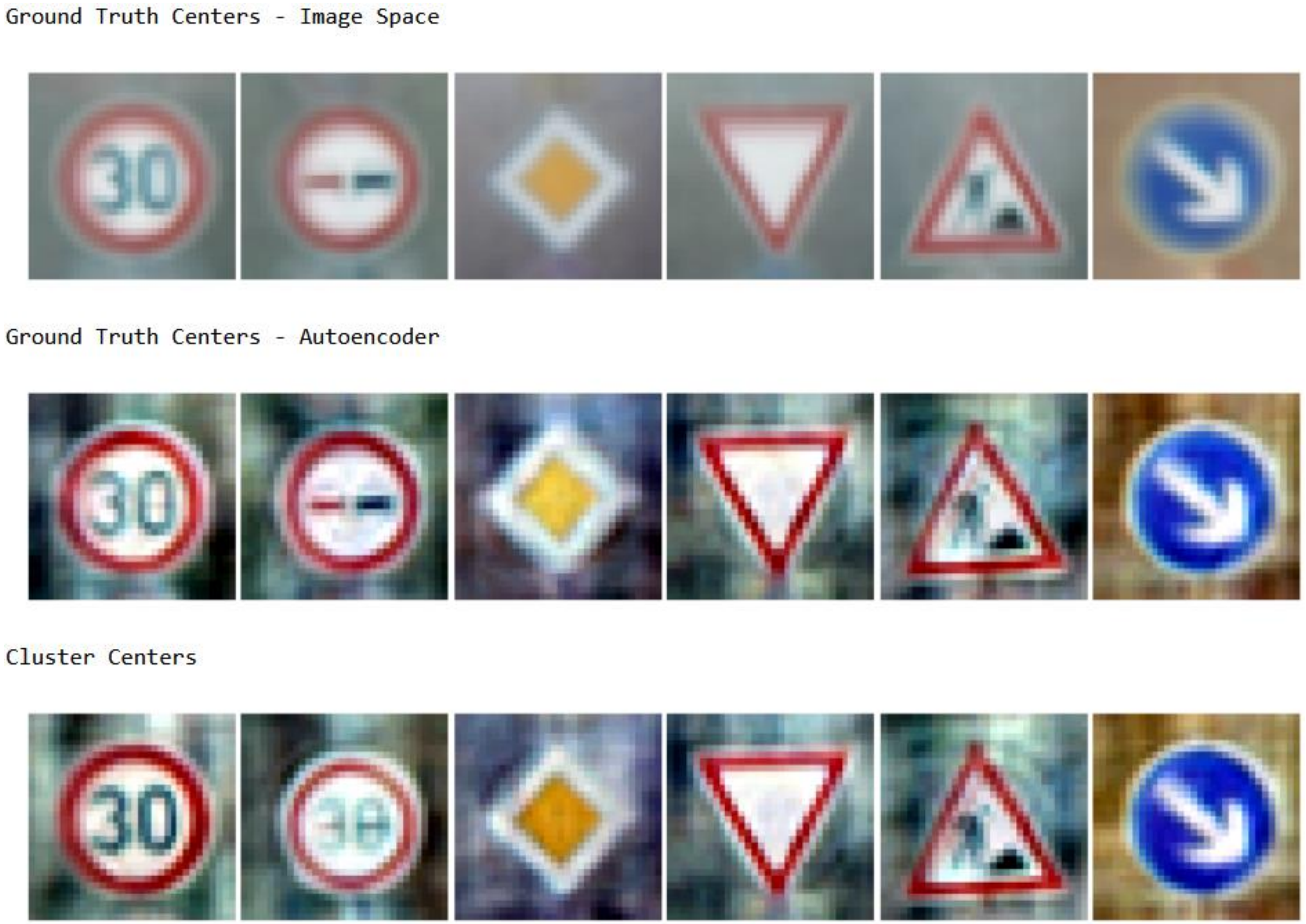
Compression loss: $l_C = KL(P||Q)$

Reconstruction loss: $l_R = ||\widehat{x_i} - x_i||_2^2$

- This alleviates to some degree the previous problem, but depends heavily on the hard to tune weighting hyperparameter λ
- Introduces a new problem called **Feature Drift [MMKK19]**
  - The reconstruction loss and the clustering loss have conflicting goals
  - Reconstruction Loss: Preserve the space as best as possible to reconstruct all features of the data
  - Compression Loss: Increase the separation of the clusters and only focus on the most discriminative features

Notebook Example

- Deep clustering with IDEC



Results

- Notebook summary

- What worked?

- What could be improved?

Results

- Notebook summary

- What worked?

- What could be improved? --> Augmentation

Motivation - Augmentation

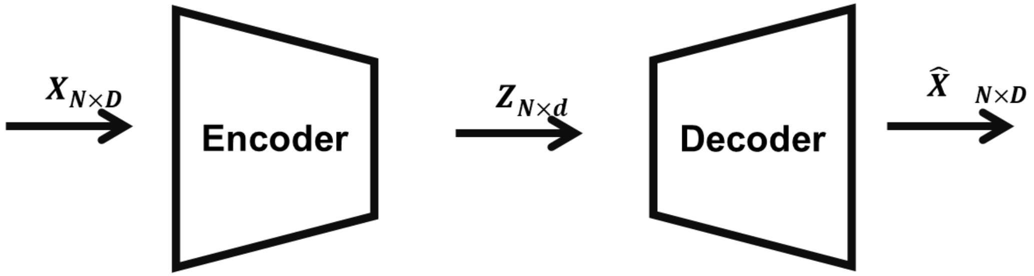
- Invariant representation learned by the autoencoder
  - Autoencoder learns to ignore certain patterns, i.e., rotations, noise, shifts,…
- Invariances inside a cluster
  - Cluster membership should not change due to spurious patterns i.e., slight rotations, lighting conditions, noise, shifts,…
- Include domain knowledge in the form of augmentation
  - E.g., we know that slight rotations of digits do not change the label assigned to them.
  - Strong rotations might flip the label, e.g., digits 6 and 9

Domain Knowledge and Invariances

- Cluster membership should not change due to spurious patterns i.e. slight rotations (Invariances inside clusters)
- $x^A \coloneqq aug(x)$ where $aug(\cdot)$ are different augmentations that we add to the original data point x.
- $z^A \coloneqq enc\big(aug(x)\big),\ \hat{x}^A \coloneqq dec(z^A)$

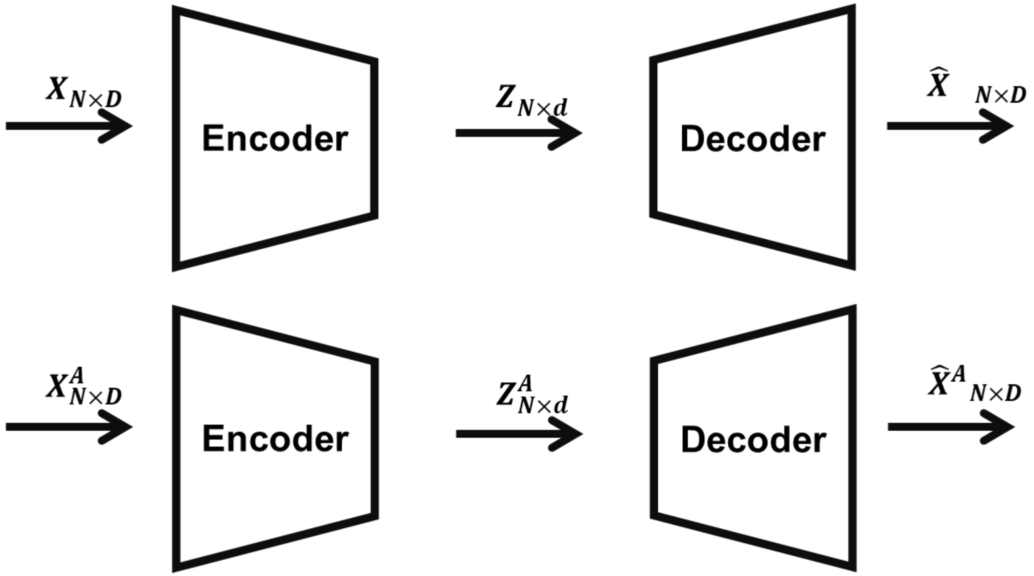
Domain Knowledge and Invariances

- Cluster membership should not change due to spurious patterns i.e. slight rotations (Invariances inside clusters)
- $x^A \coloneqq aug(x)$ where $aug(\cdot)$ are different augmentations that we add to the original data point x.
- $z^A \coloneqq enc\big(aug(x)\big),\ \hat{x}^A \coloneqq dec(z^A)$



Domain Knowledge and Invariances

- Cluster membership should not change due to spurious patterns i.e. slight rotations (Invariances inside clusters)
- $x^A \coloneqq aug(x)$ where $aug(\cdot)$ are different augmentations that we add to the original data point x.
- $z^A \coloneqq enc\big(aug(x)\big),\ \hat{x}^A \coloneqq dec(z^A)$



Domain Knowledge and Invariances

- Cluster membership should not change due to spurious patterns i.e. slight rotations (Invariances inside clusters)
- $x^A := aug(x)$ where $aug(\cdot)$ are different augmentations that we add to the original data point x.
- $z^A := enc\big(aug(x)\big),\ \hat{x}^A := dec(z^A)$
- New loss function $l = l_C^A + l_R^A$

$$l_C^A = ||z_i - \mu_i||_2^2 + ||z_i^A - \mu_i||_2^2$$
$$l_R^A = \ ||\widehat{x_i} - x_i||_2^2 + ||\hat{x}_i^A - x^A||_2^2$$

Domain Knowledge and Invariances

- Cluster membership should not change due to spurious patterns i.e. slight rotations (Invariances inside clusters)
- $x^A \coloneqq aug(x)$ where $aug(\cdot)$ are different augmentations that we add to the original data point x.
- $z^A \coloneqq enc\big(aug(x)\big),\ \hat{x}^A \coloneqq dec(z^A)$
- New loss function $l = l_C^A + l_R^A$

$$l_C^A = ||z_i - \boxed{\mu_i}||_2^2 + ||z_i^A - \boxed{\mu_i}||_2^2$$
$$l_R^A = ||\hat{x}_i - x_i||_2^2 + ||\hat{x}_i^A - x^A||_2^2$$

- We use the cluster assignments and centroids learned from our “clean” examples
- Thus we force the augmented data points to be in the same cluster as their originals

Notebook Example

Augmented images

Original images

Outline

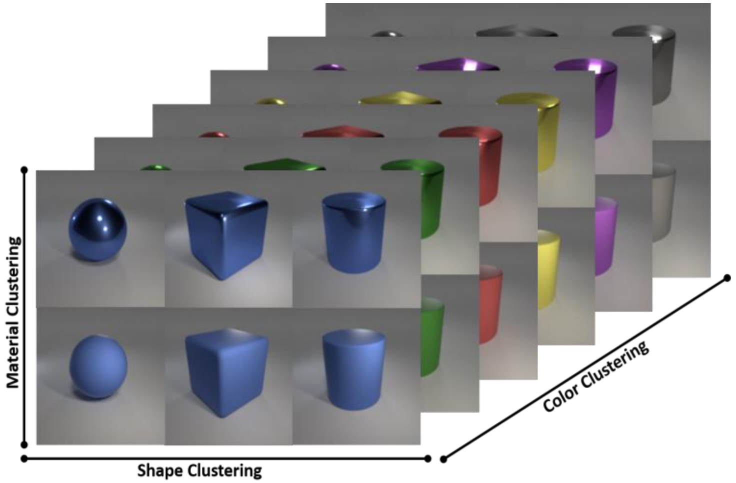
- Introduction to Clustering
- Introduction to Deep Clustering
- Application of Deep Clustering Algorithms
- Recent Approaches
- Outlook

Specialized Deep Clustering Algorithms

- Flat & Centroid based approaches
  - DEC [XGF16]
  - IDEC [GGLY17]
  - DCN [YFSH17]
  - ACe/DeC [MBMTBP21]
- Spectral Clustering
  - SpectralNet [SSLBNK18]
  - DualAE [YDZYL19]
- Mutual Information
  - IMSAT [HMTMS17]
  - IIC [JHV19]
- Density based
  - DDC [LCCC18]
- Probabilistic Methods
  - ClusterGAN [MALK19]
  - VADE [JYTTZ17]
- Other Approaches
  - Hierarchical Clustering
    - DeepECT [MPB19]
  - Non-Redundant Clustering:
    - ENRC [MMABP20]
  - Subspace Clustering
    - DSC [JZLSR17]
  - K-estimation
    - DipDECK [LBSBP21]

Deep Non-Redundant Clustering

- Embedded Non-Redundant Clustering algorithm (ENRC) [MMABP20]



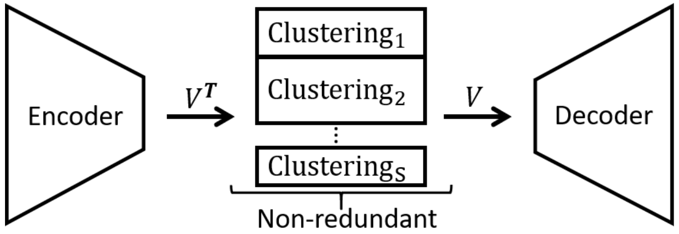
Non-redundant clusterings:

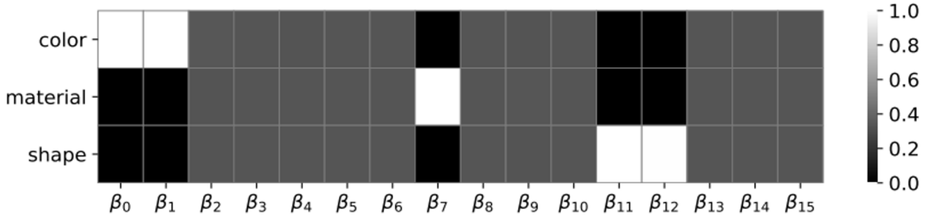
- Shapes : Cube, Cylinder, Sphere
- Colors: Red, Blue, Green, Yellow Purple, Grey
- Material: Rubber, Metal

How to find all three clusterings with unsupervised deep learning?

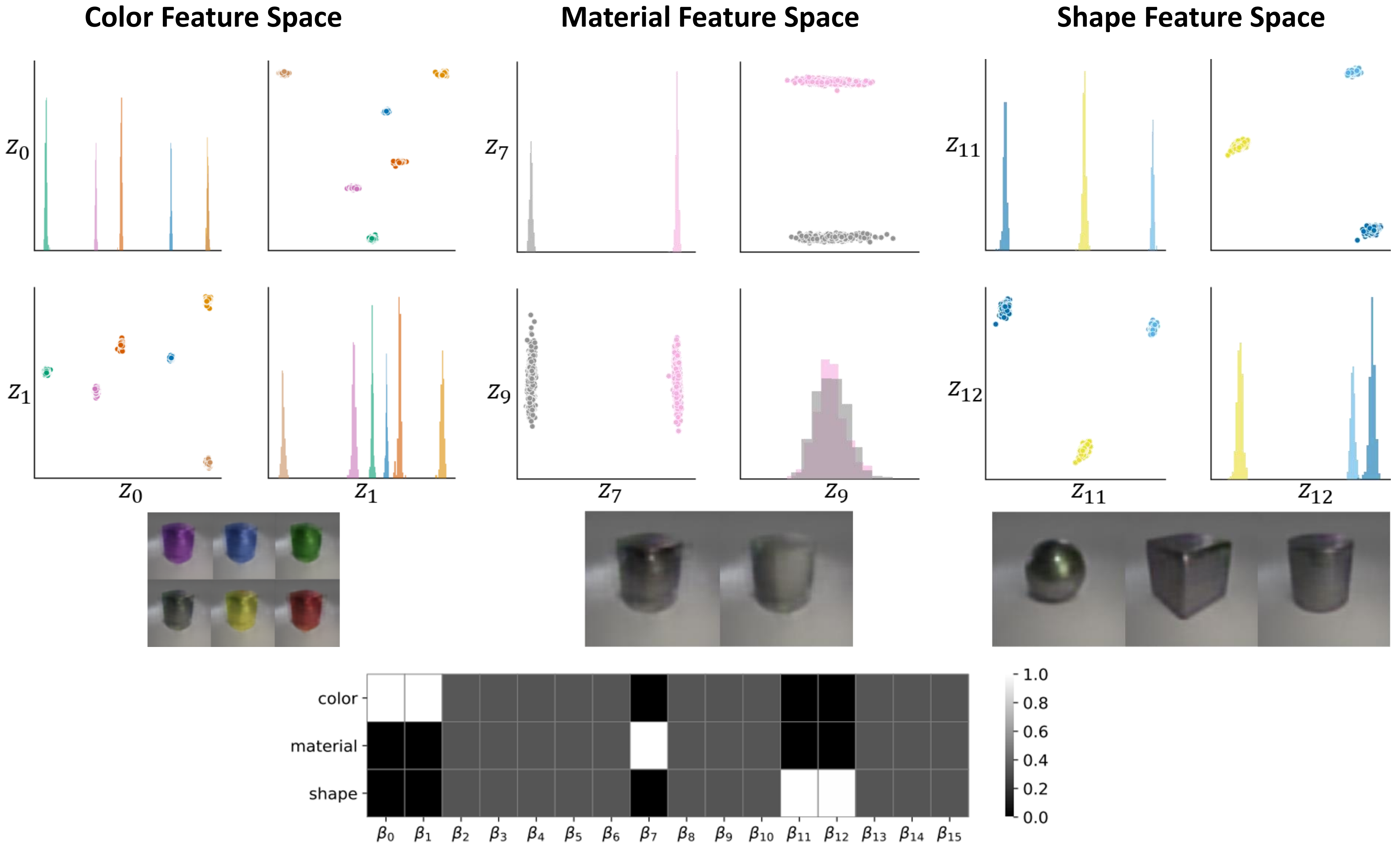
→ **Non-redundant clustering layer:**

Softly split the embedded space with learnable feature weights



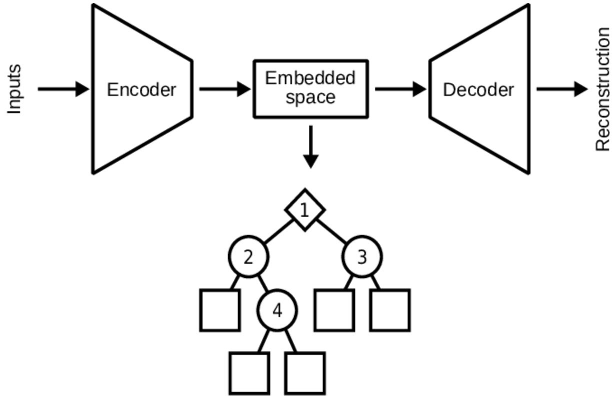


Deep Non-Redundant Clustering



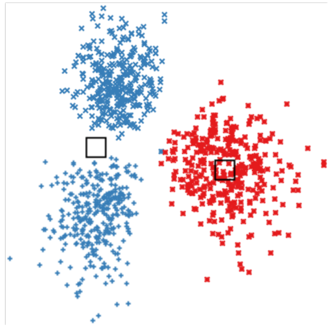
Deep Hierarchical Clustering

- Deep Embedded Cluster Tree (DeepECT) [MPB19]
  - Based on Bisecting Kmeans model
  - Recursively split embedded space in with $k = 2$
  - Uses projected cluster loss
    - Preserve structure along orthogonal dimensions spanned by the two centroids

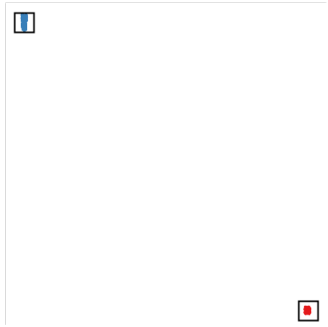


Example:

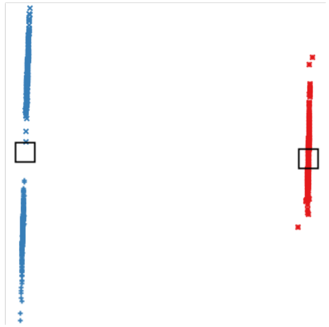
Original



Naïve Approach

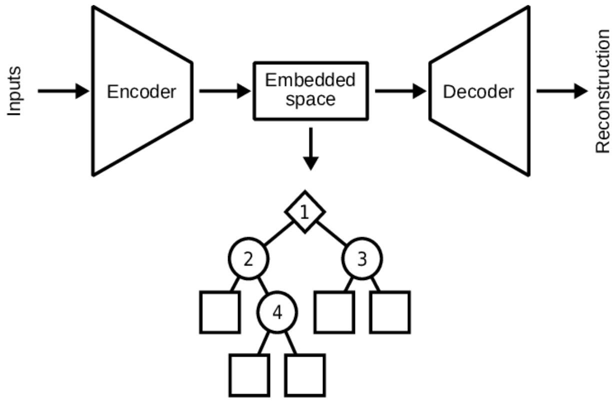


Projected Loss



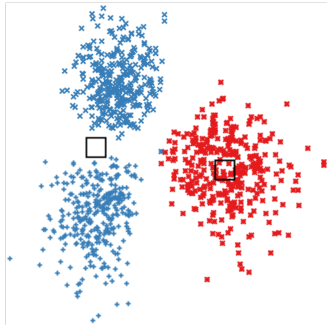
Deep Hierarchical Clustering

- Deep Embedded Cluster Tree (DeepECT) [MPB19]
  - Based on Bisecting Kmeans model
  - Recursively split embedded space in with $k = 2$
  - Uses projected cluster loss
    - Preserve structure along orthogonal dimensions spanned by the two centroids

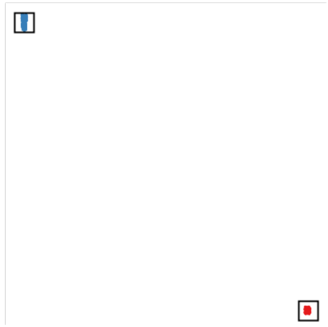


Example:

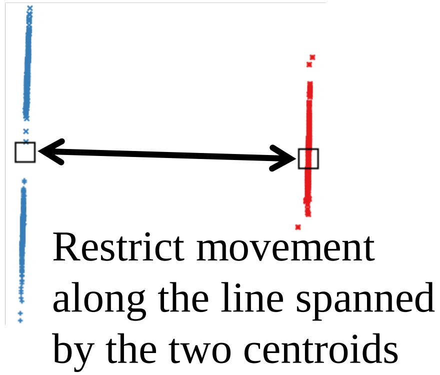
Original



Naïve Approach



Projected Loss



Deep Hierarchical Clustering

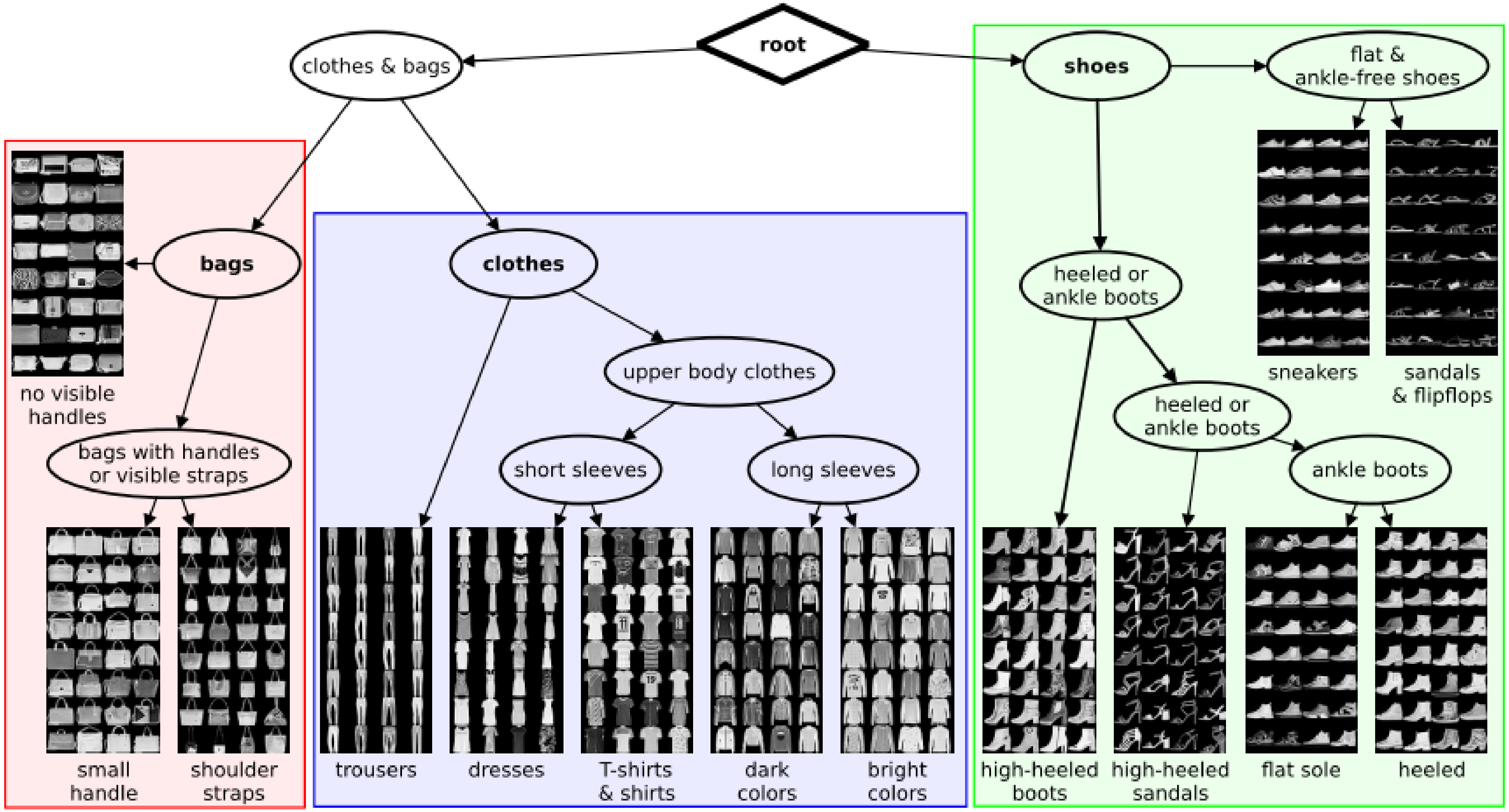
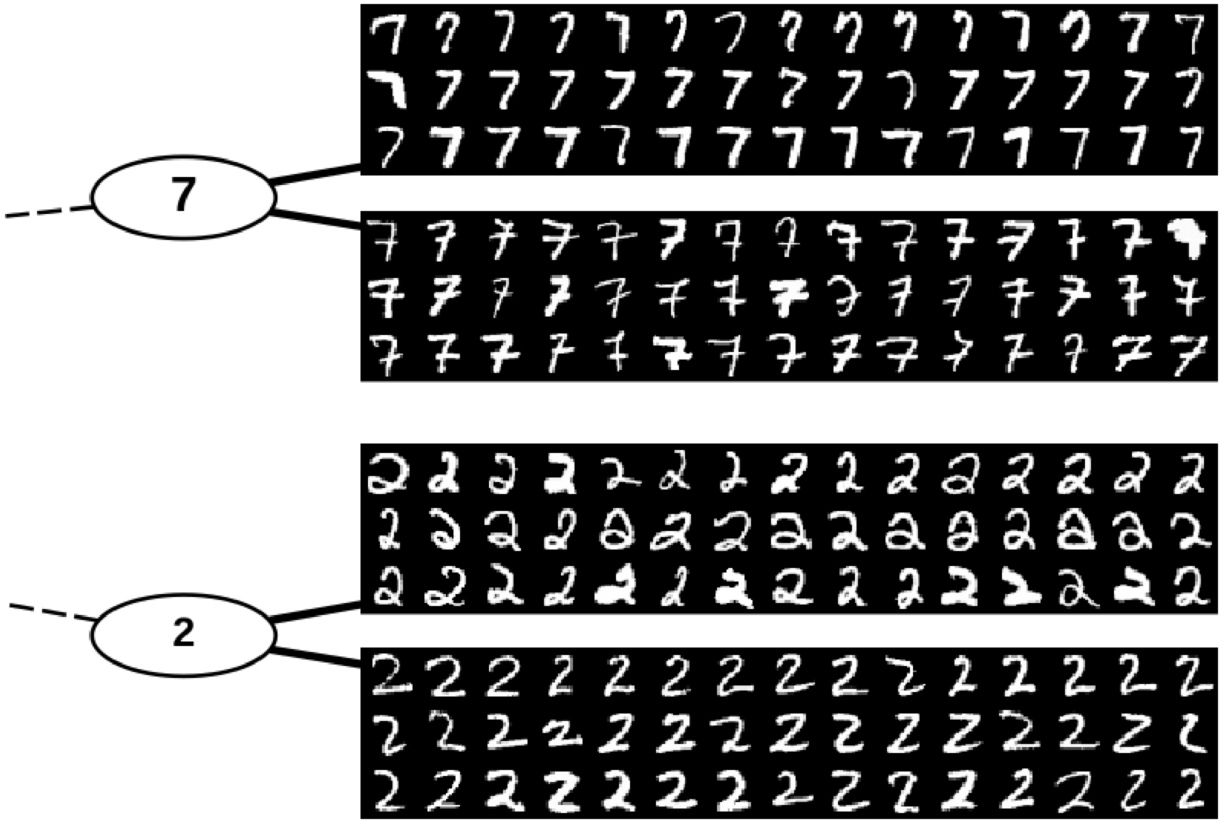


Fig. 2. The diagram shows a cluster tree for the Fashion-MNIST dataset. Each leaf node shows randomly sampled objects assigned to it. The labels are interpretations by the authors. The colored areas highlight the three dominant sub-trees representing three types of objects found in the dataset: bags, clothes, and shoes.

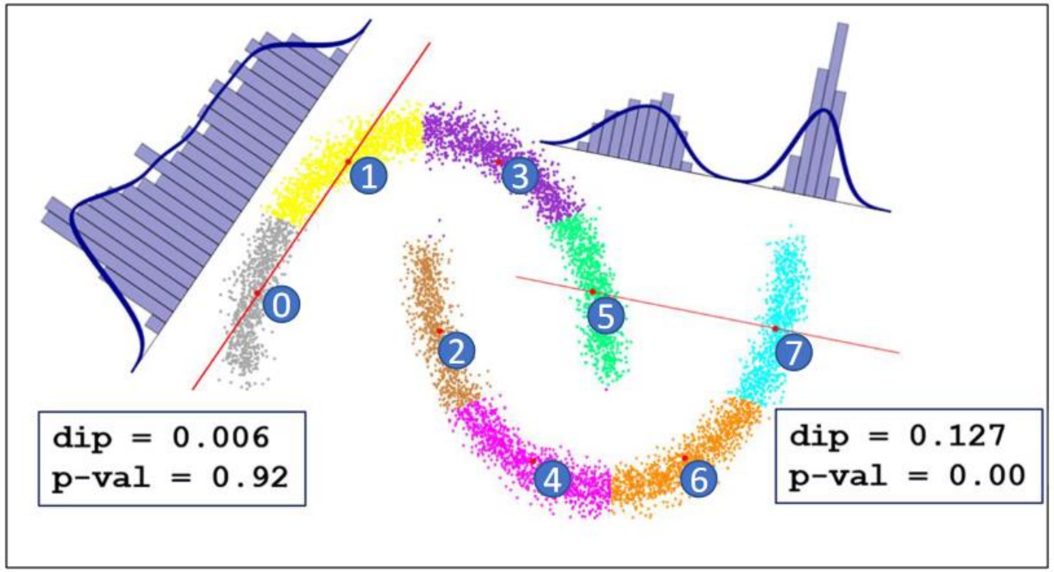
Deep Hierarchical Clustering

Finding populations and sub-populations and hierarchical structures e.g. different types of 7's and 2's

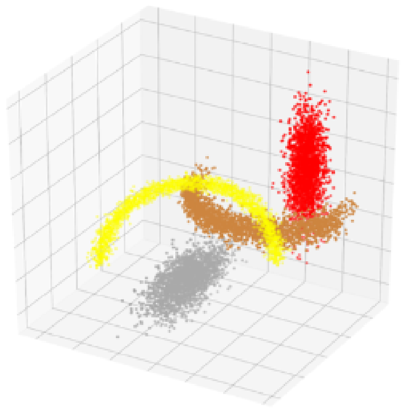


Deep Clustering with k-estimation

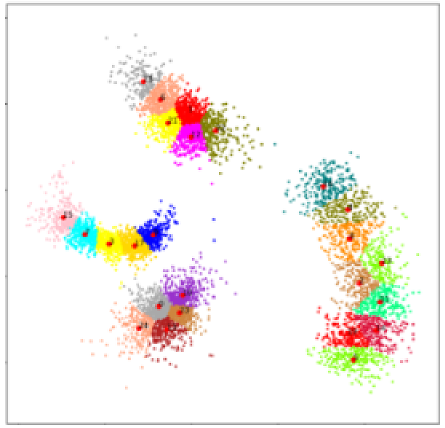
- Dip-based Deep Embedded Clustering with k-estimation (DipDECK) [LBSBP21]
- Problem: 'True' number of clusters is often unknown
- Idea: Overestimate the number of clusters and identify similar micro-clusters
  - Use Dip-test of unimodality to rate similarity
  - Micro-Clusters describing a common structure should be placed close to each other
    -> If similarity is high enough, they can be merged



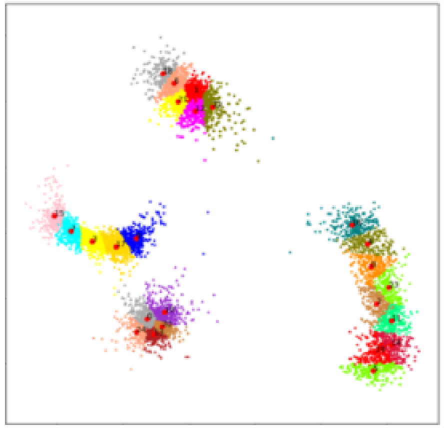
Deep Clustering with k-estimation



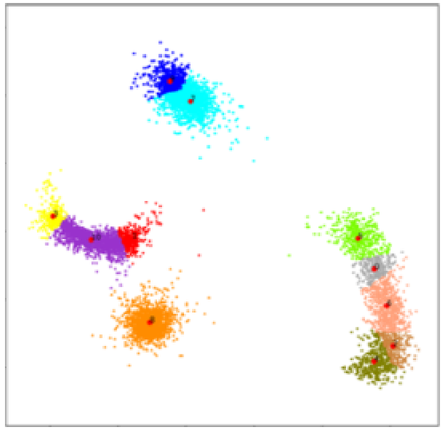
(a) Synthetic data set with 4 ground truth clusters.



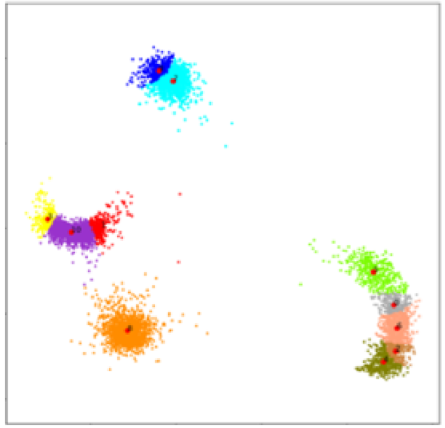
(b) Initial embedding



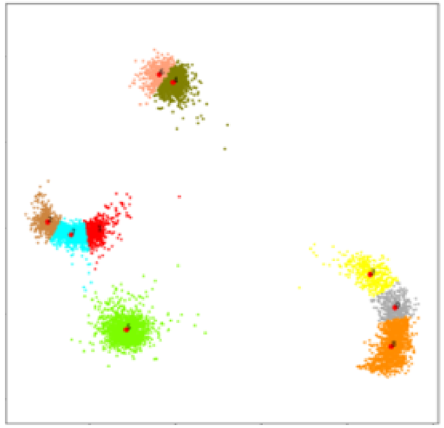
(c) Train until $p > T$



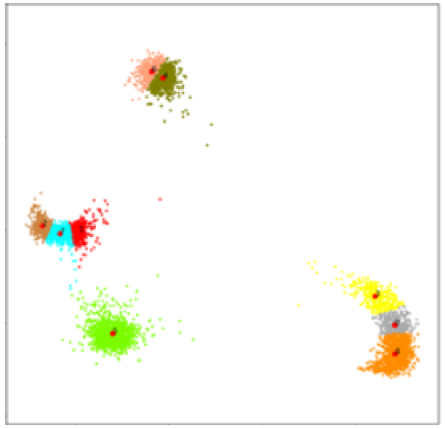
(d) After first (14) merges



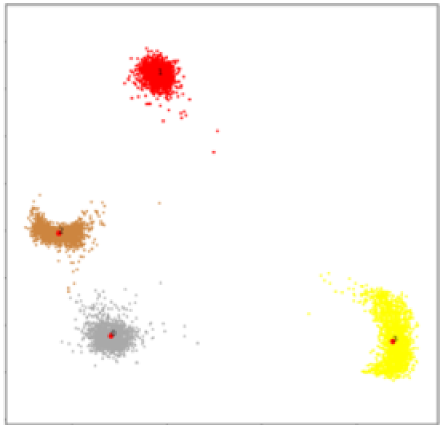
(e) Training until $p > T$



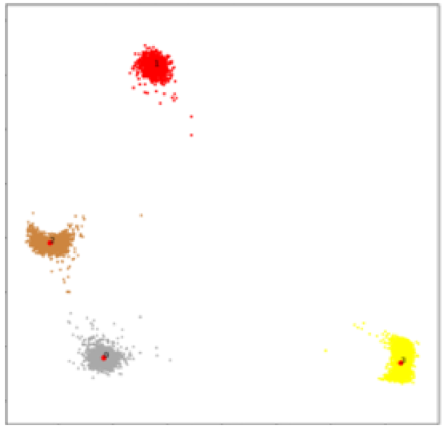
(f) After (2) more merges



(g) Training until $p > T$



(h) After (5) more merges



(i) Final result

Outline

- Introduction to Clustering
- Introduction to Deep Clustering
- Application of Deep Clustering Algorithms
- Recent Approaches
- Outlook

Discussion

- Pros:
  - Finds clusters which are non-linearly hidden in the original space
  - Can find higher “semantic” clusters e.g. digits, traffic signs, …
  - No need for feature engineering, “only” need to choose an architecture which fits the data type, e.g. convolutional neural nets for image data.
  - Fast inference for clustering unseen data from the same (unknown) distribution
  - Centroids and interpolations in the embedded space can be reconstructed and visualized in the original space.
  - Domain knowledge can be incorporated as data invariances
  - Scales to large amounts of data and dimensions

Discussion

- Cons:
  - Only useful for larger quantities of data
  - Works mostly on structured data, e.g., images, sound, text, …
  - Embedded space is hard to interpret (black box optimization)
  - Many hyperparameters (number of clusters, learning rate, batch size, architecture, …)
  - Highly dependent on a good initialization (local optima)
  - Sensitive to noise and outliers
  - Research until now is mostly empirical, no strong theoretical guarantees
  - High runtime in comparison to “classical” clustering methods
  - Need for specialized hardware (e.g., CUDA enabled GPUs, TPUs, …)

In Summary

- Representation learning for clustering (Deep Clustering) is an active research area (about 10 years of research)
- Many interesting algorithms have been proposed transferring “classical” clustering algorithms to the deep learning framework (similar to kernel approaches)
- Many problems of deep learning (e.g., high number of hyperparameters), which can be “easily” tackled in supervised learning are difficult to solve in deep unsupervised learning

Question for the Audience

- Aside from clustering, in which cases are clustered representations useful?

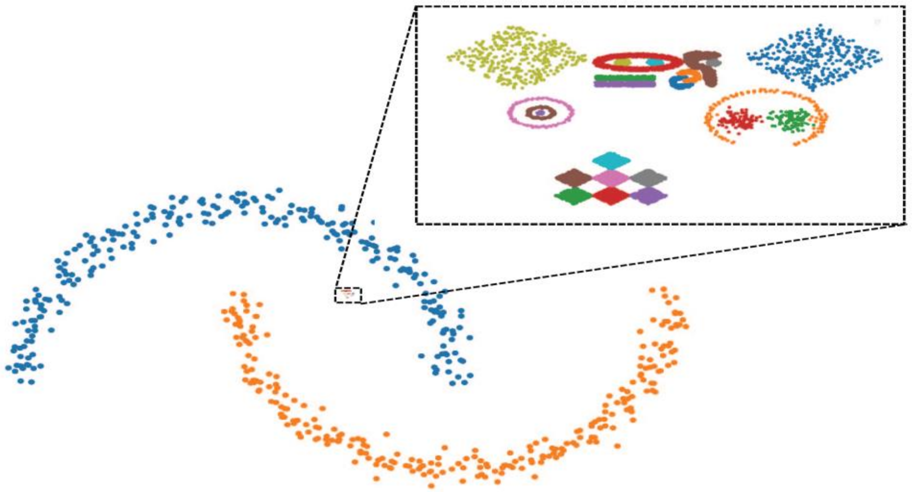
Question for the Audience

- Aside from clustering, in which cases are clustered representations useful?
  - Some thoughts:
    - In cases where abstraction is of interest, e.g., preserving only prototypical information
      - Simplified representation
      - Representations with less nuisance factors
    - In cases where we want to enforce cluster structure in the representation
      - Information retrieval
      - Task acquisition in meta-reinforcement learning [JHGELF19]
    - Other cases?
- In which might they be less useful?
  - Fine grained classification tasks
  - Generative tasks?
  - …

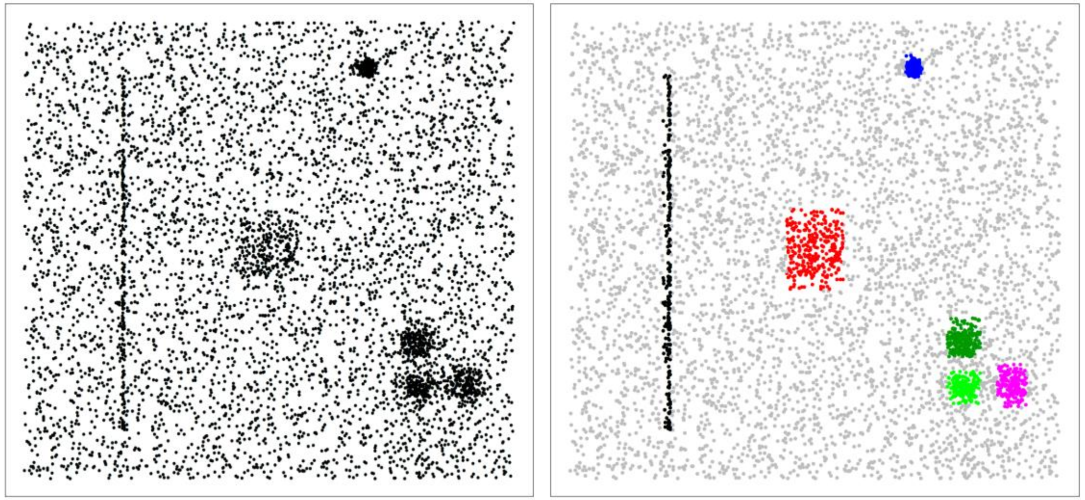
Open Problems in Deep Clustering

- Imbalanced clusters
- Adversarial Examples
- Fairness & Explainable AI
- Dependence on hyperparameters

Imbalanced Clusters, Noise, Outliers

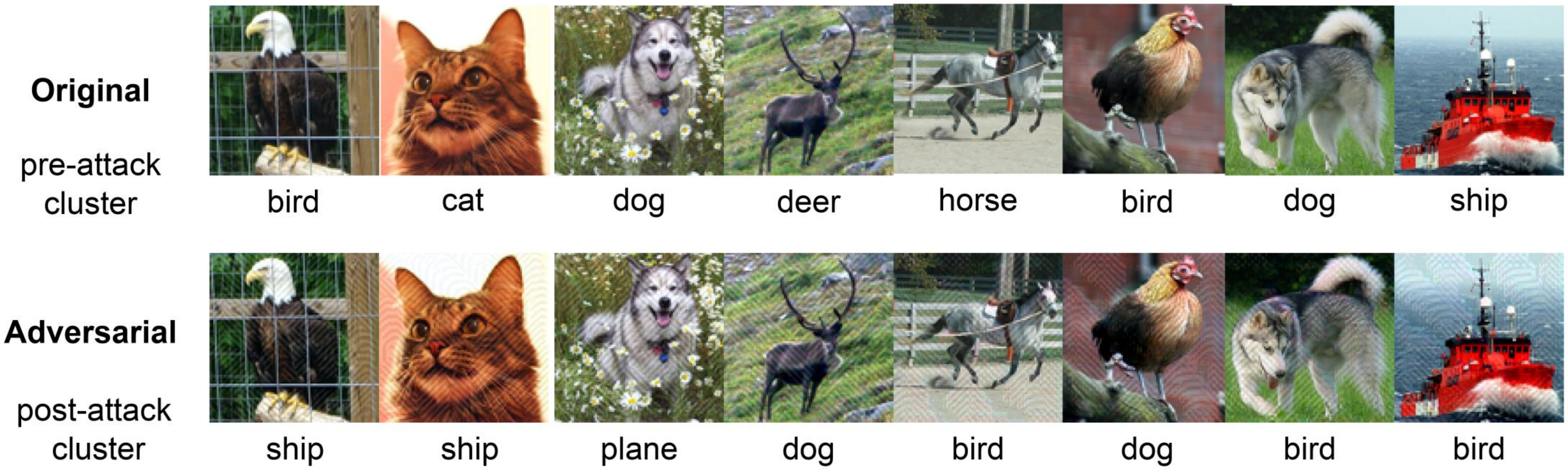


Imbalanced clusters of different scales [DMPB22].



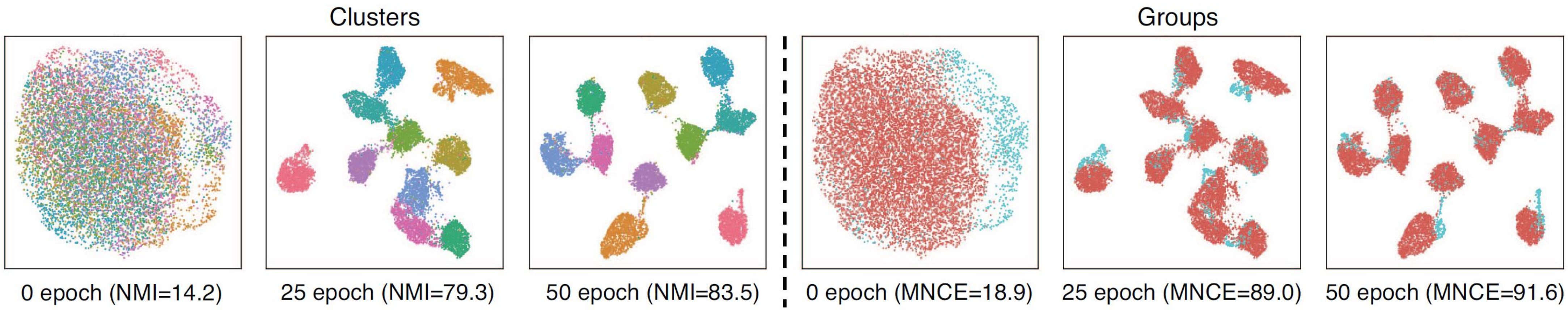
Massive amounts of noise points (80%) [MP16].

Adversarial Examples



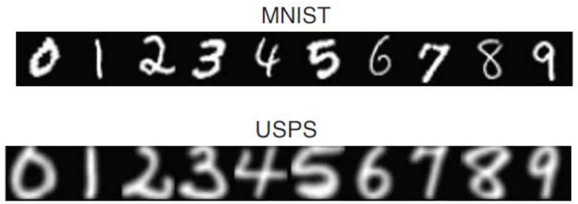
Slight modifications of the training images learned by a GAN can fool deep clustering methods [CSM22].

Fairness



Challenges:
- Single user-specified protected attribute,
- Weighting between fairness and quality.

Protected attribute:
Data source
[ZLHPLP23]



Considering the evolution of clustering methods

	High-dimensional data	Interpretability	Runtime	Parameterization
Traditional algorithms, e.g. K-means (1950 and older)	---	+++	+++	-
Subspace and spectral methods, e.g., NR-K-means [MYPB17] (starting in the 1990ies)	+	++	++	--
Deep clustering methods, e.g., ENRC [MMABP20] (popular since 2010)	+++	+	---	---

…hybrid methods might be the future.

	High-dimensional data	Interpretability	Runtime	Parameterization
Traditional clustering algorithms	---	+++	+++	-
Subspace and spectral methods	+	++	++	--
Deep clustering methods	+++	+	---	---
Hybrid methods	+++ expressiveness where needed?	++ interpretable where possible?	+ spend effort where needed?	-- partly automatic?

Contact

- Collin Leiber: leiber@dbs.ifi.lmu.de
- Lukas Miklautz: lukas.miklautz@univie.ac.at
- Claudia Plant: claudia.plant@univie.ac.at
- Christian Böhm: christian.boehm@univie.ac.at

Check out ClustPy at

References

[AGSC18] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, Daniel Cremers:
Clustering with Deep Learning: Taxonomy and New Methods. CoRR abs/1801.07648 (2018)

[APB13] Muzaffer Can Altinigneli, Claudia Plant, Christian Böhm:
Massively parallel expectation maximization using graphics processing units. KDD 2013: 838-846

[BCV13] Yoshua Bengio, Aaron C. Courville, Pascal Vincent:
Representation Learning: A Review and New Perspectives. IEEE Trans. Pattern Anal. Mach. Intell. 35(8): 1798-1828 (2013)

[BB94] Léon Bottou, Yoshua Bengio:
Convergence Properties of the K-Means Algorithms. NIPS 1994: 585-592

[CLX16] Shaosheng Cao, Wei Lu, Qiongkai Xu:
Deep Neural Networks for Learning Graph Representations. AAAI 2016: 1145-1152

[CSM22] Anshuman Chhabra, Ashwin Sekari, Prasant Mohapatra:
On the robustness of Deep Clustering Models. Adversarial Attacks and Defenses. NeurIPS2022

[DH04] Chris H. Q. Ding, Xiaofeng He:
K-means clustering via principal component analysis. ICML 2004

[DMPB22] Walid Durani, Dominik Mautz, Claudia Plant, Christian Böhm:
DBHD: Density-based clustering for highly varying density. ICDM 2022: 921-926

[FTG19] Maziar Moradi Fard, Thibaut Thonet, and Eric Gaussier
Deep k-means: Jointly clustering with k-means and learning representations. Pattern Recognition Letters

[GBC16] Ian Goodfellow and Yoshua Bengio and Aaron Courville:
Deep Learning. MIT Press, 2016

[GGLY17] Xifeng Guo, Long Gao, Xinwang Liu, Jianping Yin:
Improved Deep Embedded Clustering with Local Structure Preservation. IJCAI 2017: 1753-1759

[HPGAC18] Philip Häusser, Johannes Plapp, Vladimir Golkov, Elie Aljalbout, Daniel Cremers:
Associative Deep Clustering: Training a Classification Network with No Labels. GCPR 2018: 18-32

[JHGELF19] Allan Jabri, Kyle Hsu, Abhishek Gupta, Ben Eysenbach, Sergey Levine, Chelsea Finn:
Unsupervised Curricula for Visual Meta-Reinforcement Learning. NeurIPS 2019: 10519-10530

References

[JHV19] Xu Ji, João F. Henriques, Andrea Vedaldi:
Invariant Information Distillation for Unsupervised Image Segmentation and Clustering. ICCV 2019: forthcoming

[JZLSR17] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, Ian D. Reid:
Deep Subspace Clustering Networks. NIPS 2017: 24-33

[LBSBP21] Collin Leiber, Lena Bauer, Benjamin Schelling, Claudia Plant, Christian Böhm
Dip-based deep embedded clustering with k-estimation. KDD 2021: 903-913

[MSFK18] Naveen Sai Madiraju, Seid M. Sadat, Dimitry Fisher, Homa Karimabadi
Deep Temporal Clustering : Fully Unsupervised Learning of Time-Domain Features. CoRR abs/1802.01059 (2018): unpublished

[MZLC19] Qianli Ma, Jiawei Zheng, Sen Li, Gary W. Cottrell:
Learning Representations for Time Series Clustering. NeurIPS 2019: 3776-3786

[MPB19] Dominik Mautz, Claudia Plant and Christian Böhm:
Deep Embedded Cluster Tree. ICDM 2019: forthcoming

[MBMTBP21] Lukas Miklautz, Lena Bauer, Dominik Mautz, Sebastian Tschiatschek, Christian Böhm and Claudia Plant:
Details (Don't) Matter: Isolating Cluster Information in Deep Embedded Spaces. IJCAI 2021: 2826-2832

[MMABP20] Lukas Miklautz, Dominik Mautz, Muzaffer Can Altinigneli, Christian Böhm and Claudia Plant:
Deep Embedded Non-Redundant Clustering. AAAI 2020: 5174-5181

[MMKK19] Nairouz Mrabah, Naimul Mefraz Khan, Riadh Ksantini:
Deep Clustering with a Dynamic Autoencoder. CoRR abs/1901.07752 (2019): unpublished

[MYPB18] Dominik Mautz, Wei Ye, Claudia Plant, Christian Böhm:
Towards an Optimal Subspace for K-Means. KDD 2017: 365-373

[MP16] Samuel Maurus, Claudia Plant:
Skinny-dip: Clustering in a Sea of Noise. KDD 2016: 1055-1064

[MALK19] Sudipto Mukherjee, Himanshu Asnani, Eugene Lin, Sreeram Kannan:
ClusterGAN: Latent Space Clustering in Generative Adversarial Networks. AAAI 2019: 4610-4617

[PB10] Claudia Plant, Christian Böhm:
Parallel EM-Clustering: Fast Convergence by Asynchronous Model Updates. ICDM Workshops 2010: 178-185

References

[S10] D. Sculley:
Web-scale k-means clustering. WWW 2010: 1177-1178

[SSLBNK18] Uri Shaham, Kelly P. Stanton, Henry Li, Ronen Basri, Boaz Nadler, Yuval Kluger:
SpectralNet: Spectral Clustering using Deep Neural Networks. ICLR 2018

[TNSZ19] Panagiotis Tzirakis, Mihalis A. Nicolaou, Björn W. Schuller, Stefanos Zafeiriou:
Time-series Clustering with Jointly Learning Deep Representations, Clusters and Temporal Boundaries. FG 2019: 1-5

[VLLBM10] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol:
Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. J. Mach. Learn. Res. 11: 3371-3408 (2010)

[HMTMS17] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, Masashi Sugiyama:
Learning Discrete Representations via Information Maximizing Self-Augmented Training. ICML 2017: 1558-1567

[XGF16] Junyuan Xie, Ross B. Girshick, Ali Farhadi:
Unsupervised Deep Embedding for Clustering Analysis. ICML 2016: 478-487

[YDZYL19] Xu Yang, Cheng Deng, Feng Zheng, Junchi Yan, Wei Liu:
Deep Spectral Clustering Using Dual Autoencoder Network. CVPR 2019: 4066-4075

[YFSH17] Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, Mingyi Hong:
Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering. ICML 2017: 3861-3870

[JYTTZ17] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, Hanning Zhou:
Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering. IJCAI 2017: 1965-1972

[ZLHPLP23] Pengxin Zeng, Yunfan Li, Peng Hu, Dezhong Peng, Jiancheng Lv, Xi Peng
Deep Fair Clustering via Maximizing and Minimizing Mutual Information: Theory Algorithm and Metric. CVPR 2023: 23986-23995