Revisiting the K-means algorithm

with Kernels and Autoencoders

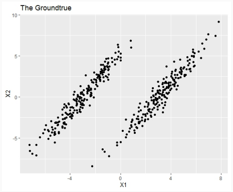
GRBIO retreat, 2024

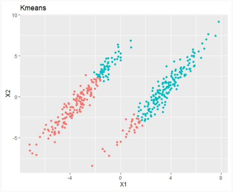
Table of Contents

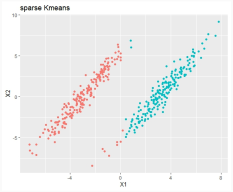
1. Sparse K-means

Motivating example

We wish to cluster the observations, and we suspect that the true underlying clusters differ only with respect to some of the features.







Witten, D. M., Tibshirani, R. (2010). A framework for feature selection in clustering. JASA, 105(490), 713-726. Implemented in **sparcl** R package.

Feature additive dissimilarity

- $\boldsymbol{X}$ denotes an $n \times p$ data matrix, with n observations and p features.
- $d(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$ denotes some measure of dissimilarity between observations $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i'}$, which are rows i and i' of the data matrix $\boldsymbol{X}$.
- Assume that d is **additive in the features**. That is,

$$d(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \sum_{j=1}^{p} d_{i,i',j}$$

  where $d_{i,i',j}$ indicates the dissimilarity between observations i and i' along feature j.
- The authors take d to be **squared** Euclidean distance, $d_{i,i',j} = (X_{ij} - X_{i'j})^2$. However, other dissimilarity measures are possible, such as the absolute difference, $d_{i,i',j} = |X_{ij} - X_{i'j}|$.

K-means clustering criterion

K-means clustering minimizes the within-cluster sum of squares. It seeks to partition the n observations into K clusters, such that:

$$\sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} \sum_{j=1}^{p} d_{i,i',j}$$

is minimal, where n_k is the number of observations in cluster k and C_k contains the indices of the observations in cluster k.
Note that if we define the between-cluster sum of squares as

$$\sum_{j=1}^{p} \Big(\frac{1}{n} \sum_{i=1}^{n} \sum_{i'=1}^{n} d_{i,i',j} - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} d_{i,i',j} \Big)$$

the minimizing the within-cluster is equivalent to maximizing the between-cluster.

Sparse K-means clustering criterion

Sparse K-means clustering criterion is as follows:

$$\max_{c_1,\ldots,c_k,\boldsymbol{w}} \Big(\sum_{j=1}^{p} w_j \big(\frac{1}{n} \sum_{i=1}^{n} \sum_{i'=1}^{n} d_{i,i',j} - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} d_{i,i',j} \big) \Big)$$

subject to

$$||\boldsymbol{w}||^2 \leq 1,\; ||\boldsymbol{w}||_1 \leq s,\; w_j \geq 0 \, \forall j$$

- The L_1 penalty results in sparsity for small values of the tuning parameter s.
- In general, without the L_2 penalty at most one element of $\boldsymbol{w}$ would be non-zero.

Algorithm for sparse K-means clustering

- Initialize $\boldsymbol{w}$ as $w_1 = \cdots = w_p = \frac{1}{\sqrt{p}}$.
- Iterate until convergence:

  a) Holding $\boldsymbol{w}$ fixed, optimize with respect $C_1, ..., C_K$:

  $$\min_{C_1,\dots,C_K} \Big(\sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} \sum_{j=1}^{p} w_j d_{i,i',j} \Big)$$

  by applying the **standard** K**-means** algorithm to the $n \times n$ dissimilarity matrix with (i, i') element $\sum_{j=1}^{p} w_j d_{i,i',j}$.

  b) Holding $C_1, ..., C_K$ fixed, optimize with respect $\boldsymbol{w}$:

  $$\max_{\mathbf{w}} \left\{ \boldsymbol{w}^\intercal \boldsymbol{a} \right\}$$

  subject to $||\boldsymbol{w}||^2 \leq 1, \quad ||\boldsymbol{w}||_1 \leq s, \quad w_j \geq 0 \, \forall j$, where $a_j = \frac{1}{n} \sum_{i=1}^{n} \sum_{i'=1}^{n} d_{i,i',j} - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} d_{i,i',j}$, by applying **soft-thresholding** method (Hastie et al. 2015).
- Clusters: $C_1, ..., C_K$, Feature weights: $w_1, ..., w_p$.

2. A kernelization of Sparse K-means

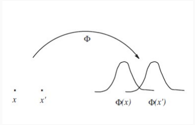
Positive semidefinite kernel function

- A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is positive semidefinite if for all integers $n \geq 1$ and elements $\{\boldsymbol{x}_i\}_{i=1}^{n} \subset \mathcal{X}$, the $n \times n$ matrix with elements $\boldsymbol{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is positive semidefinite.
- When $\mathcal{X} \subseteq \mathbb{R}^p$,
  - the linear kernel is defined by $k(\boldsymbol{x}, \boldsymbol{z}) = \langle \boldsymbol{x}, \boldsymbol{z} \rangle$,
  - the polynomial kernel by $k(\boldsymbol{x}, \boldsymbol{z}) = (\langle \boldsymbol{x}, \boldsymbol{z} \rangle + c)^d$, $c \geq 0$,
  - the Gaussian kernel by $k(\boldsymbol{x}, \boldsymbol{z}) = \exp\big(-\frac{1}{c}||\boldsymbol{x} - \boldsymbol{z}||^2\big)$, $c > 0$.
- The input space $\mathcal{X}$ is just any set equipped with a real valued symmetric and positive semidefinite kernel k.
- A kernel provides a measure of similarity between the points of the input space.

Feature space

- A kernel allows us identify each $\boldsymbol{x} \in \mathcal{X}$ with a real function of $\boldsymbol{x}$ through the feauture map ϕ, which dependes on the kernel k, defined as

$$\phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}} = \{f : \mathcal{X} \to \mathbb{R}\}, \qquad \boldsymbol{x} \mapsto \phi(\boldsymbol{x}) = k(\cdot, \boldsymbol{x})$$



- That map is 1-1. Identifying $\mathcal{X}$ with $\phi(\mathcal{X})$, $\phi(\mathcal{X}) \subset \mathbb{R}^{\mathcal{X}}$.
- Vector space $\mathcal{F}_k = \mathrm{span}(\phi(\mathcal{X}))$.
- The elements f of $\mathcal{F}_k$ are real valued functions with domain $\mathcal{X}$ of the form $f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, \boldsymbol{x}_i)$ for a convenient $m \in \mathbb{N}$, $\boldsymbol{x}_1, ..., \boldsymbol{x}_m \in \mathcal{X}$ and $\alpha_1, ..., \alpha_m \in \mathbb{R}$.
- Kernel k allows to define a **inner product** $\langle \cdot, \cdot \rangle_k$ in $\mathcal{F}_k$.
- The inner product space $\mathcal{F}_k$ can be turn into a HS $\mathcal{H}_k$ (Reproducing property):

$$\langle k(\cdot, \boldsymbol{x}), f \rangle_k = f(\boldsymbol{x}), \quad \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_k = k(\boldsymbol{x}, \boldsymbol{x}')$$

Distance induced by the kernel

- The distance in the feature space, $d_{\mathcal{H}_k}$, induced by the kernel.

$$\begin{aligned} d^2_{\mathcal{H}_k}(\phi(\boldsymbol{x}), \phi(\boldsymbol{x}')) &= \langle \phi(\boldsymbol{x}) - \phi(\boldsymbol{x}'), \phi(\boldsymbol{x}) - \phi(\boldsymbol{x}') \rangle_k \\ &= \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}) \rangle_k + \langle \phi(\boldsymbol{x}'), \phi(\boldsymbol{x}') \rangle_k - 2\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_k \\ &= k(\boldsymbol{x}, \boldsymbol{x}) + k(\boldsymbol{x}', \boldsymbol{x}') - 2k(\boldsymbol{x}, \boldsymbol{x}') \end{aligned}$$

- The Kernel trick

Shawe-Taylor, J. and Cristianini, N. (2004). Kernel methods for pattern analysis. Cambridge university press.

Kernel k-means

The objective function is

$$\min_{C_1,\dots,C_K} \Big(\sum_{j=1}^{K} \sum_{\phi(\boldsymbol{x}_i)\in C_j} ||\phi(\boldsymbol{x}_i) - \boldsymbol{m}_j||_k^2\Big)$$

where $\boldsymbol{m}_j$ is the center of the cluster C_j. That is $\boldsymbol{m}_j = \frac{\sum_{\phi(\boldsymbol{x}_i)\in C_j} \phi(\boldsymbol{x}_i)}{|C_j|}$. Further, $||\phi(\boldsymbol{x}_i) - \boldsymbol{m}_j||_k^2$ can be calculated from the elements of kernel matrix $\boldsymbol{K}$.

$$\begin{aligned}
||\phi(\boldsymbol{x}_i) - \boldsymbol{m}_j||_k^2 &= \langle \phi(\boldsymbol{x}_i) - \boldsymbol{m}_j, \phi(\boldsymbol{x}_i) - \boldsymbol{m}_j \rangle_k \\
&= \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_i) \rangle_k - 2\langle \phi(\boldsymbol{x}_i), \boldsymbol{m}_j \rangle_k + \langle \boldsymbol{m}_j, \boldsymbol{m}_j \rangle_k \\
&= \langle \phi(\boldsymbol{x}_i), \phi(x\boldsymbol{x}_i) \rangle_k - 2\langle \phi(\boldsymbol{x}_i), \frac{\sum_{s\in C_j} \phi(\boldsymbol{x}_s)}{|C_j|} \rangle_k + \langle \frac{\sum_{s\in C_j} \phi(\boldsymbol{x}_s)}{|C_j|}, \frac{\sum_{l\in C_j} \phi(\boldsymbol{x}_l)}{|C_j|} \rangle_k \\
&= k(\boldsymbol{x}_i, \boldsymbol{x}_i) - 2\frac{1}{|C_j|} \sum_{s\in C_j} k(\boldsymbol{x}_i, \boldsymbol{x}_s) + \frac{1}{|C_j|^2} \sum_{s\in C_j} \sum_{l\in C_j} k(\boldsymbol{x}_l, \boldsymbol{x}_s) \\
&= k(\boldsymbol{x}_i, \boldsymbol{x}_i) - 2F(\boldsymbol{x}_i, C_j) + G(C_j)
\end{aligned}$$

Kernel k-means algorithm

Input: Initial partition $C_1, ..., C_K$ and $\boldsymbol{m}_1, ..., \boldsymbol{m}_K$

1. For each cluster C_j find $G(C_j)$
2. Compute $F(\boldsymbol{x}_i, C_j)$ for each $\boldsymbol{x}_i$ and each cluster C_j
3. Find $||\phi(\boldsymbol{x}_i) - \boldsymbol{m}_j||_k^2$ and assign $\boldsymbol{x}_i$ to the nearest center.
4. Update $\boldsymbol{m}_j$, $j = 1, ..., K$.
5. Repeat step 1 through step 4 until convergence.

Output: Final partition $C_1, ..., C_K$ and $\boldsymbol{m}_1, ..., \boldsymbol{m}_K$

Multiple Kernel Learning

- MKL (Bach et al, 2004) combine M subkernels $\{k_m\}_{m=1}^{M}$ where each subkernel k_m uses only a distinct set of features in $\mathcal{X}$.
- The kernel k is represented as the weighted sum of those subkernels

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{m=1}^{M} w_m k_m(\boldsymbol{x}_m, \boldsymbol{x}'_m)$$

  where $\boldsymbol{x}_m$ denotes the m-th set of features of $\boldsymbol{x}$.
- The weight coeffcients of subkernels $w_1, ..., w_m$ must be tuned to optimize a certain problem-dependent objective function.
- Multiple Kernel K-means (Du, L et al. 2015).
- Feature-wise kernel (Yamada, M et al. 2014). The sets of features have only a single feature, then each subkernel corresponds to each feature

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j=1}^{p} w_j k_j(x_j, x'_j)$$

Feature-wise kernel additive distance

- With the single feature-wise kernel, the induced distance in the feature space is additive in the features

$$\begin{aligned}
d^2_{\mathcal{H}_k}(\boldsymbol{x}, \boldsymbol{x}') &= k(\boldsymbol{x}, \boldsymbol{x}) + k(\boldsymbol{x}', \boldsymbol{x}') - 2k(\boldsymbol{x}, \boldsymbol{x}') \\
&= \sum_{j=1}^{p} w_j k_j(x_j, x_j) + \sum_{j=1}^{p} w_j k_j(x'_j, x'_j) - 2\sum_{j=1}^{p} w_j k_j(x_j, x'_j) \\
&= \sum_{j=1}^{p} w_j \big(k_j(x_j, x_j) + k_j(x'_j, x'_j) - 2k_j(x_j, x'_j) \big) \\
&= \sum_{j=1}^{p} w_j d^2_{\mathcal{H}_{k_j}}(x_j, x'_j)
\end{aligned}$$

- Kernelization of the sparse k-means algorithm

Algorithm for sparse kernel K-means clustering

- Initialize $\boldsymbol{w}$ as $w_1 = \cdots = w_p = \frac{1}{\sqrt{p}}$.
- Iterate until convergence:
  a) Holding $\boldsymbol{w}$ fixed, optimize with respect $C_1, ..., C_K$:

$$\min_{C_1,\dots,C_K} \Big(\sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} \sum_{j=1}^{p} w_j d^2_{\mathcal{H}_{k_j}}(x_{i,j}, x_{i',j}) \Big)$$

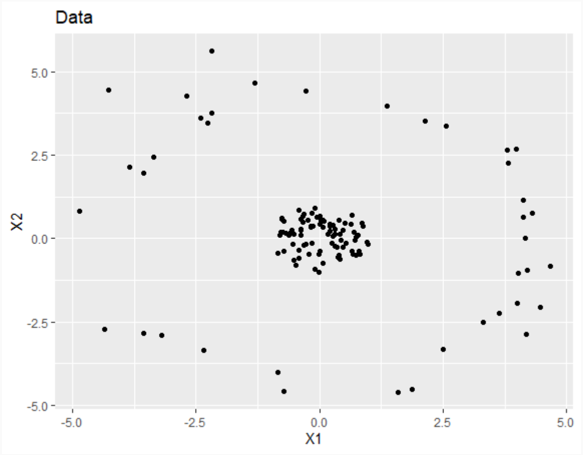
  by applying the standard kernel K-means algorithm to the $n \times n$ dissimilarity matrix with (i, i') element $\sum_{j=1}^{p} w_j d^2_{\mathcal{H}_{k_j}}(x_{i,j}, x_{i',j})$.

  b) Holding $C_1, ..., C_K$ fixed, optimize with respect $\boldsymbol{w}$:

$$\max_{\mathbf{w}} \left\{ \boldsymbol{w}^\intercal \boldsymbol{a} \right\}$$

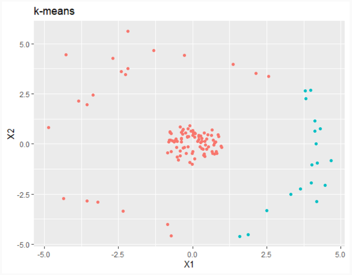
  subject to $||\boldsymbol{w}||^2 \leq 1, \quad ||\boldsymbol{w}||_1 \leq s, \quad w_j \geq 0 \, \forall j$, where
  $a_j = \frac{1}{n} \sum_{i=1}^{n} \sum_{i'=1}^{n} d^2_{\mathcal{H}_{k_j}}(x_{i,j}, x_{i',j}) - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} d^2_{\mathcal{H}_{k_j}}(x_{i,j}, x_{i',j})$,
  by applying **soft-thresholding**.
- Clusters: $C_1, ..., C_K$, Feature weights: $w_1, ..., w_p$.

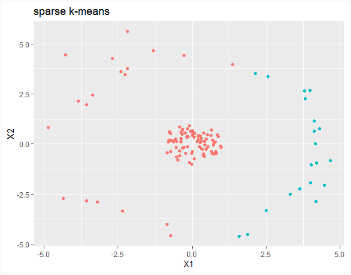
Doughnut data

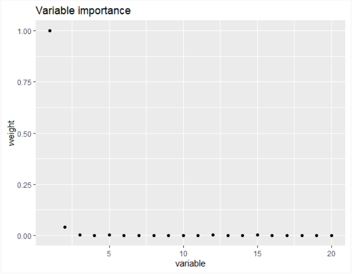


$X_3, ..., X_{20}$ are $U(-1, 1)$ i.i.d.

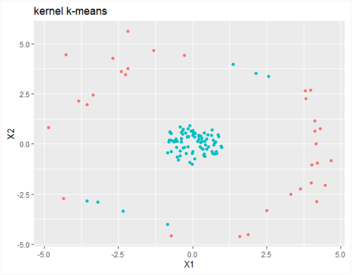
K-means and sparse K-means

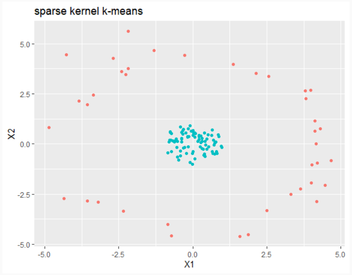


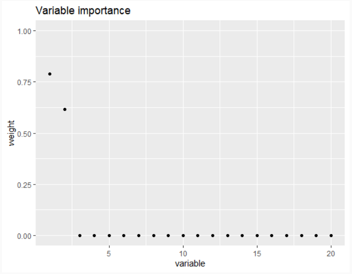




Kernel K-means and sparse Kernel K-means



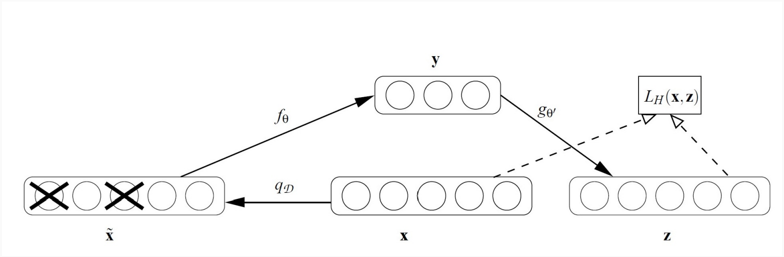




(poly, c=0,d=2)

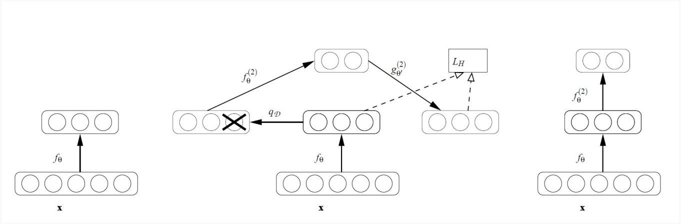
3. When Autoencoders meet Clustering

Denoising Autoencoders



Vincent, P. et *al*. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. JMLR, 2010.

Stacked Autoencoders (SAE)

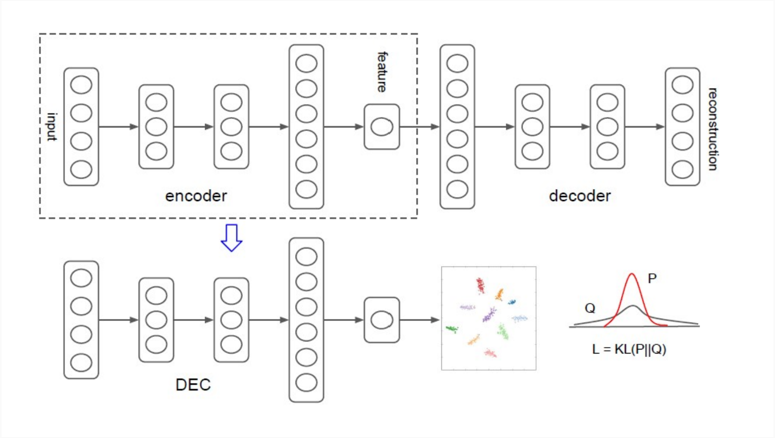


$$\boldsymbol{x} \xrightarrow{f_\theta^{(1)}} \boldsymbol{y} \xrightarrow{g_{\theta'}^{(1)}} \boldsymbol{x}$$

$$\boldsymbol{y} \xrightarrow{f_\theta^{(2)}} \boldsymbol{z} \xrightarrow{g_{\theta'}^{(2)}} \boldsymbol{y}$$

$$\boldsymbol{x} \xrightarrow{f_\theta^{(1)}} \boldsymbol{y} \xrightarrow{f_\theta^{(2)}} \boldsymbol{z} \xrightarrow{g_{\theta'}^{(2)}} \boldsymbol{y} \xrightarrow{g_{\theta'}^{(1)}} \boldsymbol{x}$$

Deep Embedding Clustering - Network structure



Xie, J., Girshick, R. and Farhadi, A. (2016, June). Unsupervised deep embedding for clustering analysis. In International conference on machine learning (pp. 478-487). PMLR.

Deep Embedding Clustering - Pipeline

DEC clusters data by simultaneously learning a set of k cluster centers $\{\boldsymbol{\mu}_j \in Z\}_{j=1}^{k}$ in the feature space Z and the parameters $\boldsymbol{\theta}$ of the encoder part of the SAE, $\boldsymbol{x} \xrightarrow{f_\theta^{(2)} \circ f_\theta^{(1)}} \boldsymbol{z}$, that maps data points into Z.

Two phases:

1. Parameter initialization ($\boldsymbol{\theta}$ and $\{\boldsymbol{\mu}_j\}_{j=1}^{k}$).
   1.1 Initialize DEC paremeters $\boldsymbol{\theta}$ with SAE training.
   1.2 Initialize $\{\boldsymbol{\mu}_j\}$ by k-means clustering on embedded data in feature space Z.
2. Parameter optimization (clustering). Iteration between two steps
   2.1 Computing the soft-assignment Q and the auxiliary target distribution P,
   2.2 Minimize with respect $\boldsymbol{\theta}$ and $\{\boldsymbol{\mu}_j\}_{j=1}^{k}$ the KL divergence between P and Q.

Soft Assignment

Use the Student's t-distribution as a kernel to measure the similarity between embedded point $\boldsymbol{z}_i$ and centroid $\boldsymbol{\mu}_j$

$$q_{ij} = \frac{(1 + ||\boldsymbol{z}_i - \boldsymbol{\mu}_j||^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'}(1 + ||\boldsymbol{z}_i - \boldsymbol{\mu}_{j'}||^2/\alpha)^{-\frac{\alpha+1}{2}}}$$

where $\boldsymbol{z}_i(\boldsymbol{\theta}) = (f_\theta^{(2)} \circ f_\theta^{(1)})(\boldsymbol{x}_i) \in Z$ corresponds to $\boldsymbol{x}_i \in X$ after the embedding, α are the degrees of freedom of t and q_{ij} can be seen as the "probability" of assigning sample i to cluster j (soft-assignment).

van der Maten et *al*. Visualizing data using t-SNE. JMLR, 2008

Kullback–Leibler divergence minimization

The model is trained by matching the soft-assignment q_i to the target distribution p_i. The objective is defined as KL divergence loss

$$L = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}}$$

The **target distribution** P (pursued properties: 1) more emphasis on data points assigned with high confidence, 2) normalize loss contribution of each centroid to prevent large clusters)

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_{j'} \left(q_{ij'}^2 / \sum_i q_{ij'} \right)}$$

Optimization

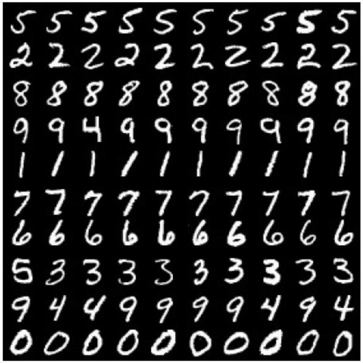
Jointly optimize the cluster centers $\{\boldsymbol{\mu}_j\}$ and encoder parameters $\boldsymbol{\theta}$

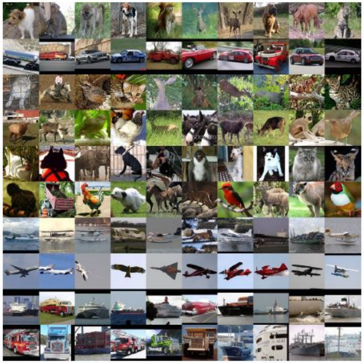
$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{z}_i} &= \frac{\alpha+1}{\alpha}\sum_j (1+\frac{||\boldsymbol{z}_i-\boldsymbol{\mu}_j||^2}{\alpha})^{-1}\cdot(p_{ij}-q_{ij})(\boldsymbol{z}_i-\boldsymbol{\mu}_j) \\
\frac{\partial L}{\partial \boldsymbol{\mu}_j} &= -\frac{\alpha+1}{\alpha}\sum_i (1+\frac{||\boldsymbol{z}_i-\boldsymbol{\mu}_j||^2}{\alpha})^{-1}\cdot(p_{ij}-q_{ij})(\boldsymbol{z}_i-\boldsymbol{\mu}_j)
\end{aligned}$$

The gradients $\frac{\partial L}{\partial \boldsymbol{z}_i}$ are passed down to the encoder and used in standard backpropagation to compute the encoder's parameters gradient

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \frac{\partial L}{\partial \boldsymbol{z}_i}\cdot\frac{\partial \boldsymbol{z}_i}{\partial \boldsymbol{\theta}}$$

DEC examples





Conclusions

- We have found a way to incorporate feature sparsity into the kernel K-means.
- The fusion of classical methods with new ones provides competitive algorithms for cluster analysis.

Moltes Gràcies!!!