Dip-based Deep Embedded Clustering with k-Estimation

Collin Leiber[1*], Lena G. M. Bauer[2*], Benjamin Schelling[2], Christian Böhm[1], Claudia Plant[2]
[1]Ludwig-Maximilians-Univeristät München, Munich, Germany {leiber, boehm}@dbs.ifi.lmu.de
[2]Universität Wien, Vienna, Austria {lena.bauer, benjamin.schelling, claudia.plant}@univie.ac.at
*First authors with equal contribution

Introduction

The combination of clustering with Deep Learning has gained much attention in recent years. Unsupervised neural networks like autoencoders can autonomously learn the essential structures in a data set. This idea can be combined with clustering objectives to learn relevant features automatically. Unfortunately, these methods are often based on a k-means framework and have to know the number of clusters a-priori.
In this paper, we present the novel clustering algorithm DipDECK, which can estimate the number of clusters simultaneously to improving a Deep Learning-based clustering objective. Our algorithm works by heavily overestimating the number of clusters in the embedded space of an autoencoder and, based on Hartigan's Dip-test - a statistical test for unimodality -, analyses the resulting micro-clusters to determine which to merge.

Contributions:

1. We combine Deep Clustering with k-Estimation by introducing Hartigan's Dip-test to Deep Learning
2. By repeatedly merging micro-clusters, DipDECK is able to identify arbitrarily shaped clusters
3. DipDECK shows impressive results, both in estimating the number of clusters and in cluster quality

Main Idea

- Pretrain the autoencoder (AE) by minimising the reconstruction loss $\mathcal{L}_{rec}$ for each batch $\mathcal{B}$

$$\mathcal{L}_{rec} = \frac{1}{|\mathcal{B}|} \sum_{x \in \mathcal{B}} ||x - \text{dec}(\text{enc}(x))||_2^2,$$

- Run k-means in the embedded space with the overestimated number of clusters $k_{init} \rightarrow$ get closest points to the k-means centres, so that they can serve as inputs to the AE

- Apply the Dip-test to obtain the Dip-value and consequently the Dip-p-value for each pairwise combination of clusters in the embedded space. Therefore, project each point assigned to either one of the clusters onto the connection line of the corresponding centres (figure in the right column illustrates this idea)

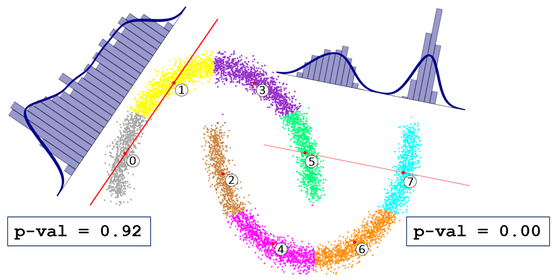
- For each batch encode the contained objects and all the cluster centres

- Update the cluster assignments in a k-means fashion

- Calculate the clustering objective $\mathcal{L}_{clu} =$

$$\frac{(1 + \text{std}(D_C))}{\text{mean}(D_C)} \frac{1}{|\mathcal{B}|} \sum_{x \in \mathcal{B}} \sum_{i=1}^{k} \hat{P}_{c_x,i} ||\text{enc}(x) - \text{enc}(\mu_i)||_2^2,$$

where c_x is the label of the cluster x is assigned to and D_C is the set of Euclidean distances between all embedded cluster centres



- mean(D_C) hinders the autoencoder of just reducing the embedding scale, std(D_C) prevents pushing individual clusters far away

- Final loss is: $\mathcal{L} = \mathcal{L}_{rec} + \mathcal{L}_{clu}$

- Update all labels, cluster centres, and the Dip-p-value matrix after each epoch

- Start merging if the maximum Dip-p-value is larger than the threshold T:

  – Reduce the number of clusters by one and merge the two clusters responsible for the Dip-p-value

  – Repeat until no Dip-p-value $\geq T$

$\Rightarrow$ Compared to methods like X-means or G-means (with and without AE for dimensionality reduction), DipDECK shows better results for the estimated number of clusters and the cluster quality. This shows the value of learning the embedding and estimating the number of clusters simultaneously.

Algorithm

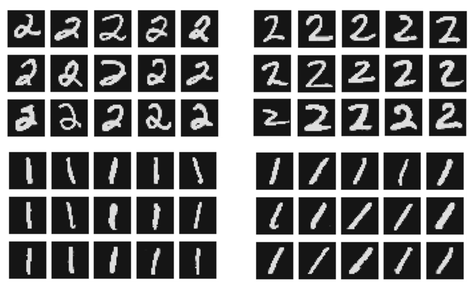
Input: data set X, starting number of clusters k_{init}, Dip-p-value threshold T, number of epochs n
Output: $labels$, k

k = k_{init}
AE = pretrained autoencoder
$(kmCentres, labels) =$ K-Means(AE.encode(X), k)
$centres =$ find closest points to $kmCentres$
$DipMatrix =$ calculate pairwise Dip-p-values of the clusters in the embedded space
$i = 0$
while $i < n$ **do**
  for $\mathcal{B}$ ***in*** X **do**
    if $i \neq 0$ **then**
      update $labels$ of $\mathcal{B}$
    calculate $\mathcal{L} = \mathcal{L}_{rec}$ + $\mathcal{L}_{clu}$ for $\mathcal{B}$
    optimise AE using $\mathcal{L}$
  update all $labels$, $centres$ and the $DipMatrix$
  i++
  // Start merging process
  while *max*($DipMatrix$) $\geq T$ **do**
    k--
    merge clusters with highest Dip-p-value $\rightarrow$ add the new centre to $centres$ and overwrite $labels$
    update the $DipMatrix$
    $i = 0$
return $labels$, k

Experiments

Method	USPS		MNIST		F-MNIST		Optdigist	
	k	NMI	k	NMI	k	NMI	k	NMI
Ground truth	10	-	10	-	10	-	10	-
DipDECK (ours)	**9.4**	**0.846**	**11.2**	**0.889**	**12.2**	**0.679**	**10.4**	**0.858**
X-means	35.0	0.607	35.0	0.551	35.0	0.512	35.0	0.709
G-means	35.0	0.608	35.0	0.550	35.0	0.511	35.0	0.715
PG-means	2.4	0.136	2.1	0.175	4.1	0.312	1.1	0.024
Dip-means	4.0	0.438	†	†	†	†	1.0	0.000
pDip-means	35.0	0.617	35.0	0.554	35.0	0.511	35.0	0.709
AE+X-means	2.0	0.293	18.1	0.620	24.4	0.570	12.8	0.804
AE+G-means	35.0	0.669	35.0	0.686	35.0	0.550	35.0	0.730
AE+PG-means	3.9	0.379	3.6	0.453	2.8	0.387	2.6	0.290
AE+Dip-means	6.8	0.617	†	†	†	†	1.0	0.000
AE+pDip-means	4.9	0.519	8.0	0.705	5.8	0.522	1.0	0.000

- Impressive results regarding estimated number of clusters and cluster quality (NMI, ARI)
- Variances show high stability of the results
- High robustness regarding k_{init} and T
- Sometimes a slight overestimation of the number of clusters $\rightarrow$ Can reveal interesting substructures in the data (see image on the right)



Example Run