Large-Scale Clustering through Functional Embedding

Frédéric Ratle* Jason Weston† Matthew L. Miller†

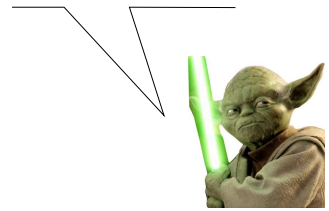
*IGAR - University of Lausanne
Switzerland

†NEC Labs America
Princeton NJ - USA

ECML PKDD 2008

A new way of performing data clustering.

- Dimensionality reduction with direct optimization over discrete labels.
- Joint optimization of embedding and clustering → improved results.
- Training by stochastic gradient descent → fast and scalable.
- Implementation within a neural network → no out-of-sample problem.



Clustering - the usual way

Popular clustering algorithms such as spectral clustering are based on a two-stage approach:

1. Find a “good” embedding
2. Perform k-means (or a similar variant)

Also:

- K-means in feature space (e.g. Dhillon *et al.* 2004)
- Margin-based clustering (e.g. Ben-Hur *et al.* 2001)

Embedding Algorithms

Many existing embedding algorithms optimize:

$$\min \sum_{i,j=1}^{U} L(f(x_i), f(x_j), W_{ij}), \quad f_i \in \mathbb{R}^d$$

MDS: minimize $(||f_i - f_j|| - W_{ij})^2$

ISOMAP: same, but W defined by shortest path on neighborhood graph.

Laplacian Eigenmaps: minimize $\sum_{ij} W_{ij} ||f_i - f_j||^2$

subject to "balancing constraint": $f^\top D f = I$ and $f^\top D 1 = 0$.

Spectral clustering $\rightarrow$ add k-means on top.

Siamese Networks: functional embedding

Equivalent to Lap. Eigenmaps but $f(x)$ is a NN.

DrLIM [Hadsell et al.,'06]:

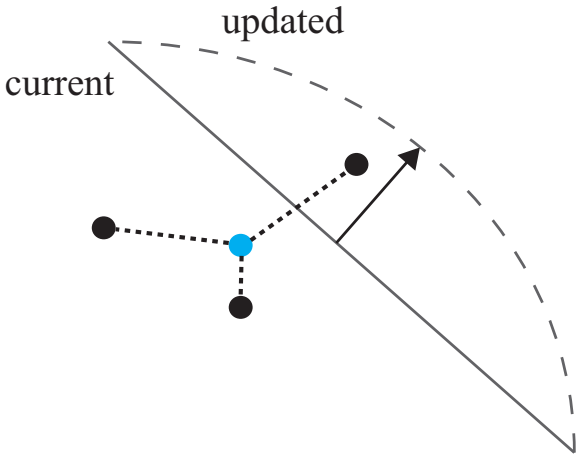
$$L(f_i, f_j, W_{ij}) = \begin{cases} ||f_i - f_j|| & \text{if } W_{ij} = 1, \\ \max(0, m - ||f_i - f_j||)^2 & \text{if } W_{ij} = 0. \end{cases}$$

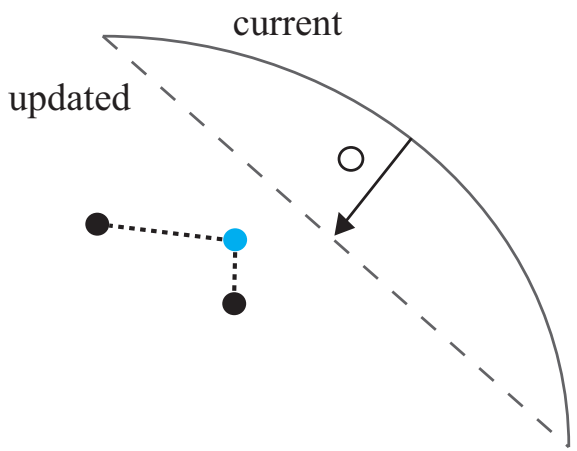
$\rightarrow$ *neighbors close, others have distance of at least m*

- Balancing handled by $W_{ij} = 0$ case $\rightarrow$ easy optimization
- $f(x)$ not just a lookup-table $\rightarrow$ control capacity, add prior knowledge, no out-of-sample problem

NCut Embedding

- Many approaches exist to learn manifolds with functional models.
- We wish to learn the clustering task directly.
- The main idea is to train a classifier $f(x)$ to:
  - Classify neighbors together.
  - Classify non-neighbors apart.





Functional Embedding for Clustering

We use a general objective of this type:

$$L(f_i, f_j, W_{ij}) = \sum_c \sum_{ij} H(f(x_i), c) \;\; Y_c(f(x_i), f(x_j), W_{ij})$$

where $H(\cdot)$ is a classification based loss function such as the hinge loss:

$$H\left(f(x), y\right) = \max(0, 1 - yf(x))$$

2-class clustering

$Y_c(f(x_i), f(x_j), W_{ij})$ encodes the weight to assign to point i being in cluster c.

It can be expressed as follows:

$$Y_c(f_i, f_j, W_{ij}) = \begin{cases} \eta^{(+)} & \text{if } \operatorname{sign}(f_i + f_j) = c \text{ and } W_{ij} = 1 \\ -\eta^{(-)} & \text{if } \operatorname{sign}(f_j) = c \text{ and } W_{ij} = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Optimization by stochastic gradient descent:

$$w^{t+1} \leftarrow w^t + \nabla L(f_i, f_j, 1)$$

NCut Embedding Algorithm.

Input: unlabeled data x_i^*, and matrix W

repeat

Pick a random pair of neighbors x_i^*, x_j^*.

Select the class $c_i = \text{sign}(f_i + f_j)$

if BalancingConstraint(c_i) **then**

Gradient step for $L(x_i^*, x_j^*, 1)$

end if

Pick a random pair x_i^*, x_k^*.

Gradient step for $L(x_i^*, x_k^*, 0)$

until stopping criterion

Balancing constraint - 2 class

Balancing constraints prevent the solution from getting trapped.

Many possible ways:

1. "Hard" constraint
   - Keep a list of the N last predictions in memory.
   - Ignore examples of class c_i if $seen(c_i) > \frac{N}{2} + \xi$
2. "Soft" constraint
   - Weigh the learning rate for each class.
   - $\eta = \frac{\eta_0}{seen(c_i)}$

Multiclass algorithm.

Two different flavours: MAX and ALL.

1. MAX approach

   Select class c_i, with $i = \text{argmax}(\max(f_i), \max(f_j))$

2. ALL approach: one learning rate per class

$$Y_c(f_i, f_j, W_{ij}) = \begin{cases} \eta_c & \text{if } W_{ij} = 1 \\ 0 & \text{otherwise} \end{cases}$$

where

$$\eta_c \leftarrow \eta^{(+)} f_c(x_i)$$

We use balancing constraints similar to those for 2-class clustering.

Small-scale datasets.

data set	classes	dims	points
g50c	2	50	550
text	2	7511	1946
bcw	2	9	569
ellips	4	50	1064
glass	6	10	214
usps	10	256	2007

Table: Small-scale datasets used throughout the experiments.

2-class experiments.

Clustering error:

	bcw	g50c	text
k-means	3.89	4.64	7.26
spectral-rbf	3.94	5.56	**6.73**
spectral-knn	3.60	6.02	12.9
NCutEmb^h	3.63	4.59	7.03
NCutEmb^s	**3.15**	**4.41**	7.89

Out-of-sample error:

k-means	4.22	**6.06**	8.75
NCutEmb^h	**3.21**	**6.06**	7.68
NCutEmb^s	3.64	6.36	**7.38**

Multiclass experiments.

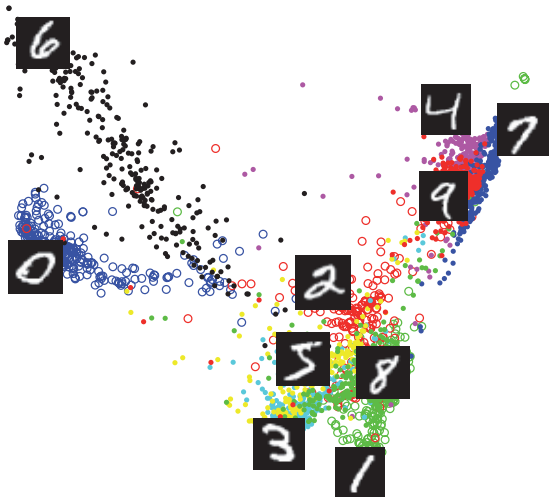
Clustering error:

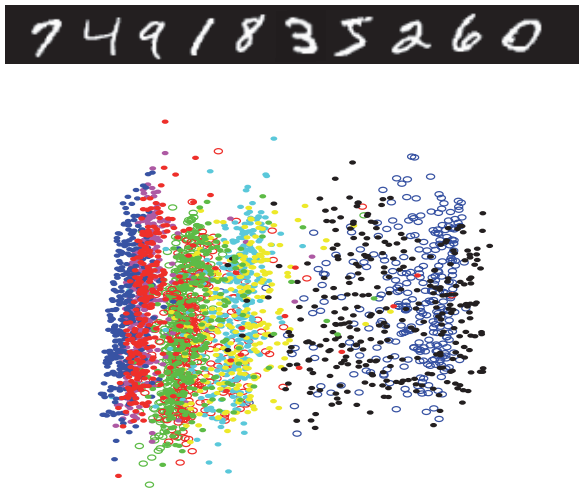
	ellips	glass	usps
k-means	20.29	25.71	30.34
spectral-rbf	10.16	39.30	32.93
spectral-knn	**2.51**	40.64	33.82
NCutEmb^{max}	4.76	**24.58**	19.36
NCutEmb^{all}	2.75	24.91	**19.05**

Out-of-sample error:

k-means	20.85	28.52	29.44
NCutEmb^{max}	5.11	25.16	20.80
NCutEmb^{all}	**2.88**	**24.96**	**17.31**

MNIST experiments





Clustering MNIST.

# clusters	method	train	test
50	k-means	18.46	17.70
	NCutEmb^{max}	**13.82**	**14.23**
	NCutEmb^{all}	18.67	18.37
20	k-means	29.00	28.03
	NCutEmb^{max}	20.12	23.43
	NCutEmb^{all}	**17.64**	**21.90**
10	k-means	40.98	39.89
	NCutEmb^{max}	**21.93**	**24.37**
	NCutEmb^{all}	24.10	24.90

Table: Clustering the MNIST database (60k train, 10k test). A one-hidden layer network has been used.

Training on Pairs?

- k-nn
  - OK for small datasets.
  - Very slow otherwise, but many methods to speed it up.
- Sequences
  - video: frames $t \& t+1 \rightarrow$ same label
  - audio: consecutive audio frames $\rightarrow$ same speaker
  - text: two words close in text $\rightarrow$ same topic
  - web: link information

Summary

- The joint optimization of clustering and embedding provides better results - or *at least* similar - to existing clustering methods.
- Functional embedding allows fast training and avoids out-of-sample problem.
- Neural nets provide a scalable and flexible framework to perform clustering.