EE-559 – Deep learning

2.5. Basic clustering and embeddings

François Fleuret
https://fleuret.org/ee559/

Sat Oct 6 00:30:20 CEST 2018

Deep learning models combine embeddings and dimension reduction operations.

They parametrize and re-parametrize multiple times the input signal under more invariant and interpretable forms.

Deep learning models combine embeddings and dimension reduction operations.

They parametrize and re-parametrize multiple times the input signal under more invariant and interpretable forms.

To get an intuition of how this is possible, we consider here two standard algorithms:

- K-means, and
- Principal Component Analysis (PCA).

Deep learning models combine embeddings and dimension reduction operations.

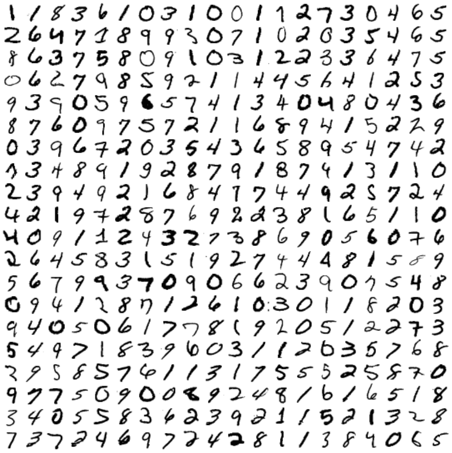
They parametrize and re-parametrize multiple times the input signal under more invariant and interpretable forms.

To get an intuition of how this is possible, we consider here two standard algorithms:

- K-means, and
- Principal Component Analysis (PCA).

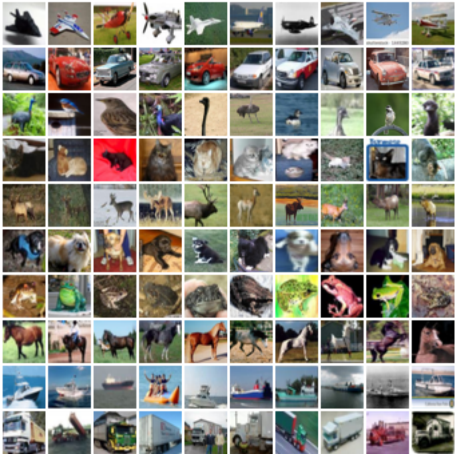
We will illustrate these methods on our two favorite data-sets.

MNIST data-set



28×28 grayscale images, 60k train samples, 10k test samples.

CIFAR10 data-set



32×32 color images, 50k train samples, 10k test samples.

(Krizhevsky, 2009, chap. 3)

Given

$$x_n \in \mathbb{R}^D, \ n = 1, \dots, N,$$

and a fixed number of clusters $K > 0$, K-means tries to find K “centroids” that span uniformly the training population.

Given

$$x_n \in \mathbb{R}^D, \ n = 1, \dots, N,$$

and a fixed number of clusters $K > 0$, K-means tries to find K "centroids" that span uniformly the training population.

Given a point, the index of its closest centroid is a good coding.

Formally, [Lloyd's algorithm for] K-means (approximately) solves

$$\underset{c_1,\dots,c_K \in \mathbb{R}^D}{\operatorname{argmin}} \sum_n \min_k \|x_n - c_k\|^2.$$

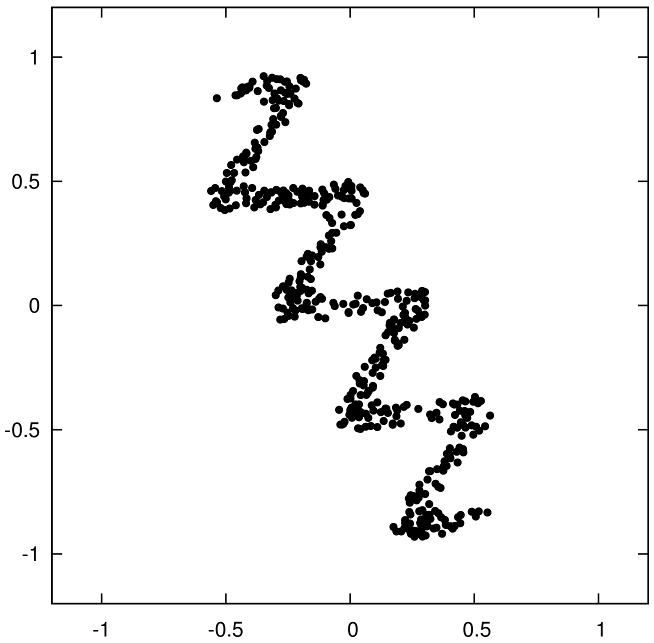
Formally, [Lloyd's algorithm for] K-means (approximately) solves

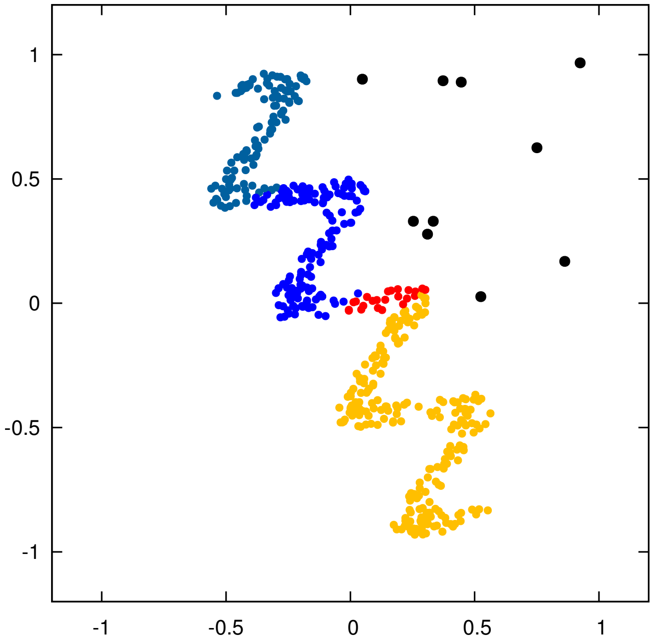
$$\underset{c_1,\dots,c_K \in \mathbb{R}^D}{\operatorname{argmin}} \sum_n \min_k \|x_n - c_k\|^2.$$

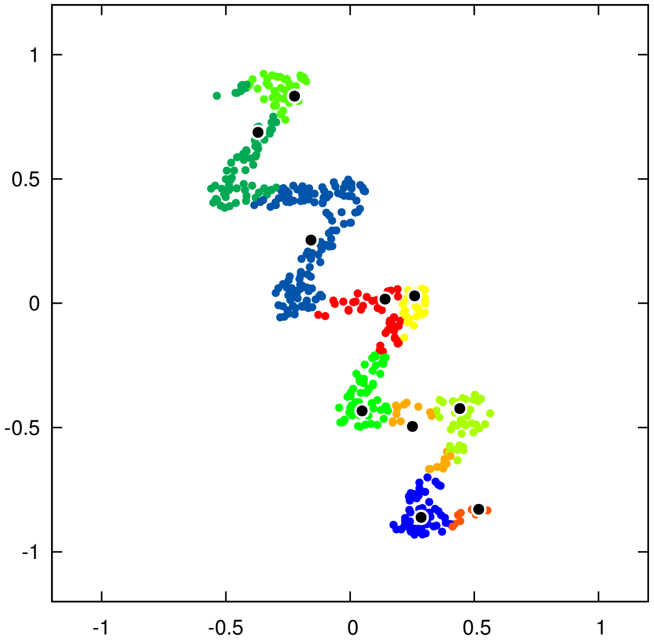
This is achieved with a random initialization of $c_1^0, \dots c_K^0$ followed by repeating until convergence:

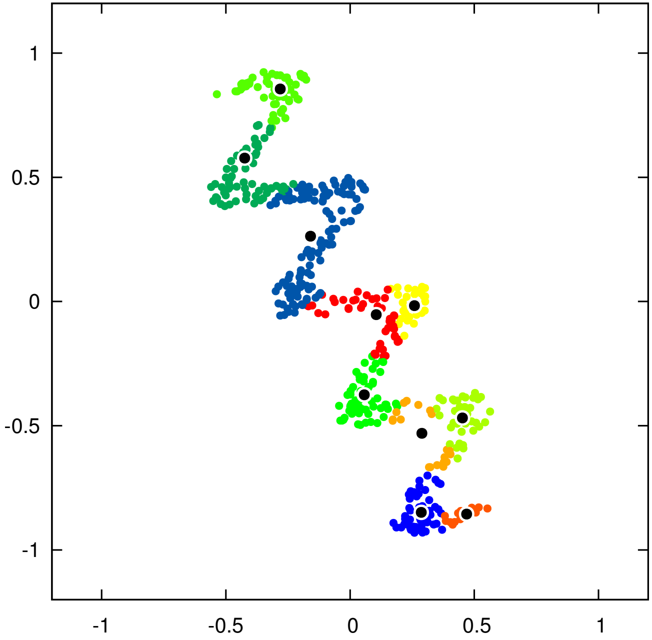
$$\begin{aligned} \forall n, k_n^t &= \underset{k}{\operatorname{argmin}} \|x_n - c_k^t\| & (1) \\ \forall k, c_k^{t+1} &= \frac{1}{|n : k_n^t = k|} \sum_{n : k_n^t = k} x_n & (2) \end{aligned}$$

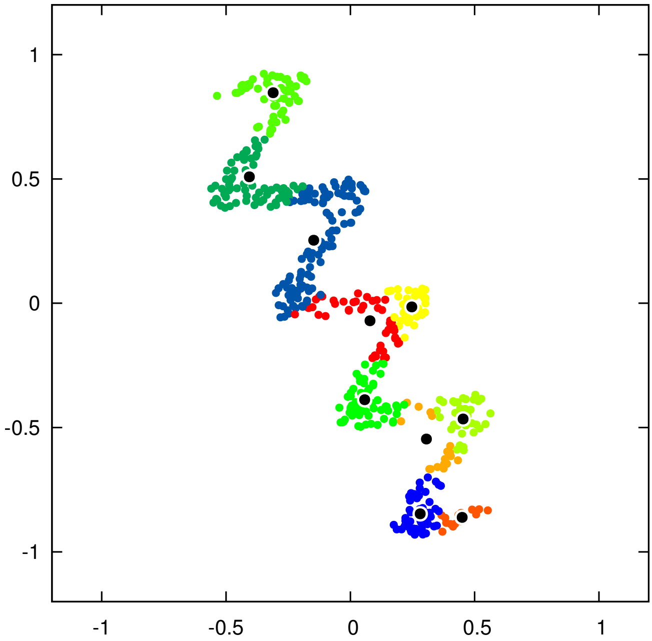
At every iteration, (1) each sample is associated to its closest centroid's cluster, and (2) each centroid is updated to the average of its cluster.

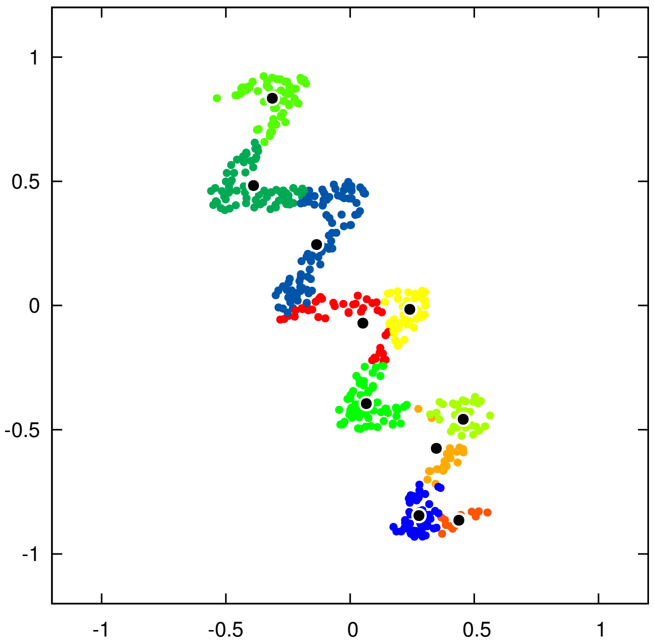

1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

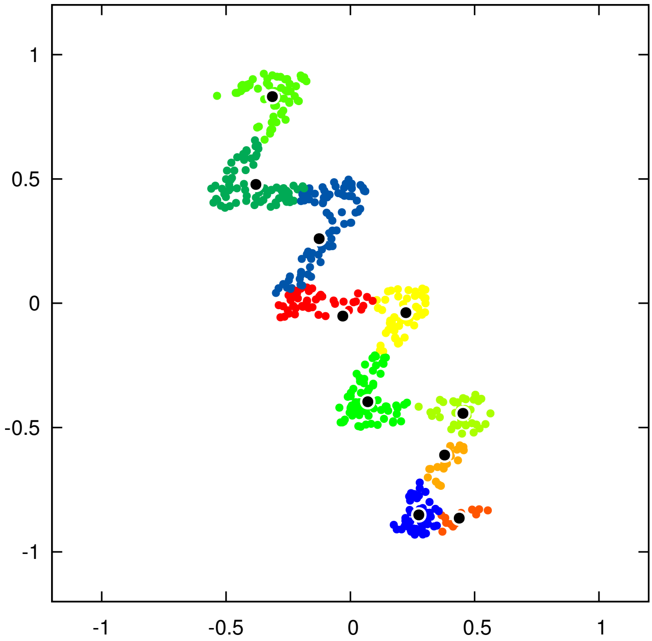

1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

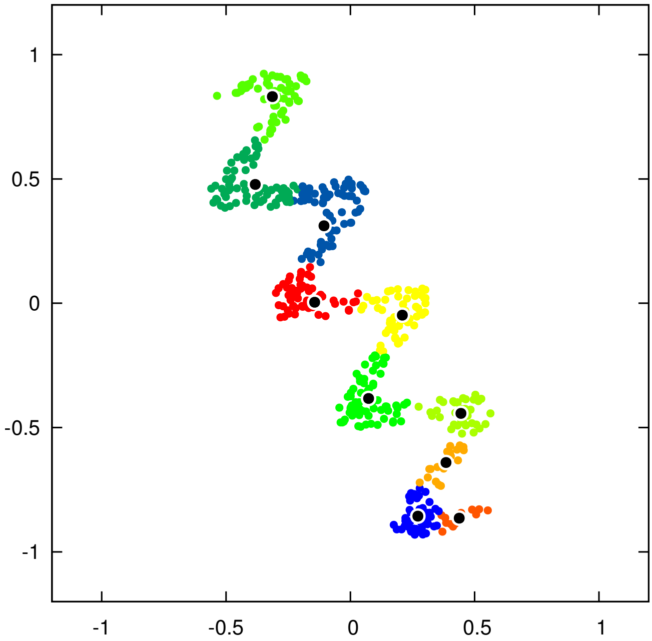

1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

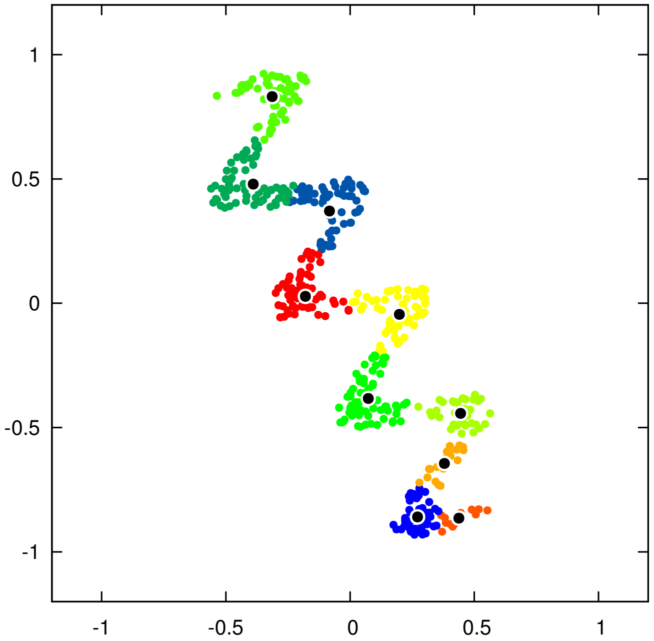

1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

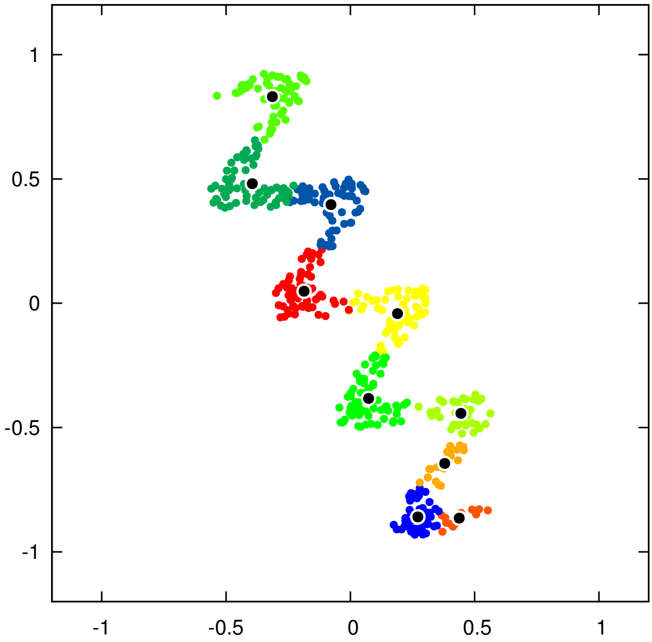

1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

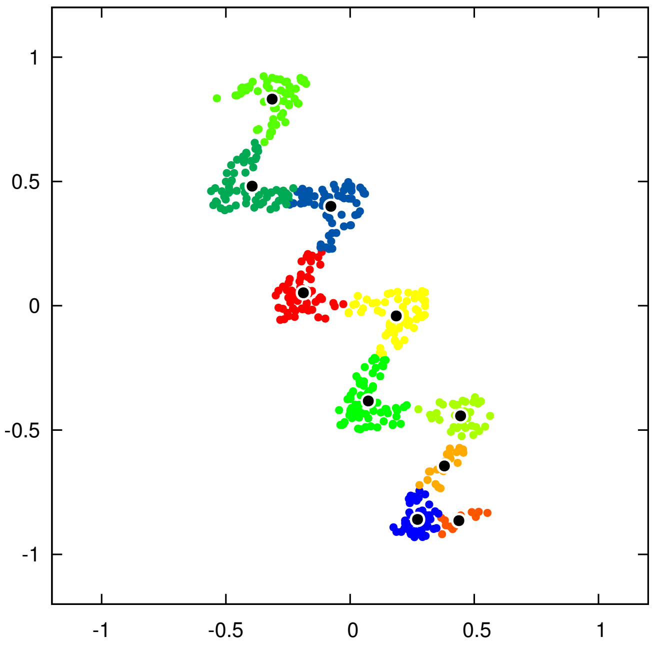

1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

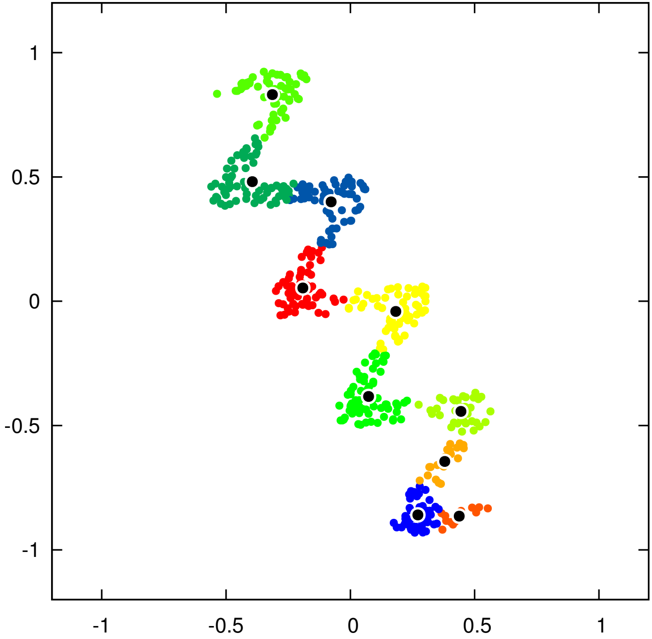

1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1


1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1


1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1


1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

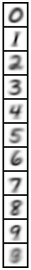

1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

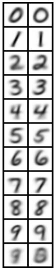

1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

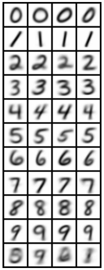
We can apply that algorithm to images from MNIST ($28 \times 28 \times 1$) or CIFAR ($32 \times 32 \times 3$) by considering them as vectors from $\mathbb{R}^{784}$ and $\mathbb{R}^{3072}$ respectively.

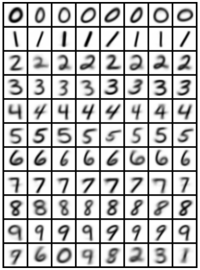
Centroids can similarly be visualized as images.

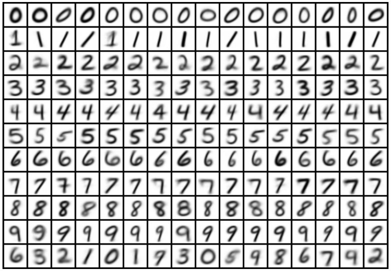
Clustering can be done per-class, or for all the classes mixed.

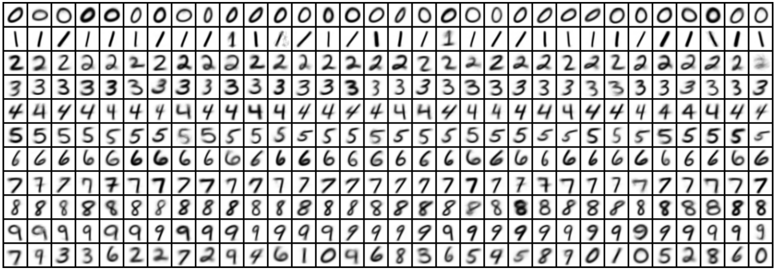

$K = 1$


$K = 2$

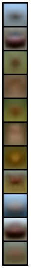

$K = 4$

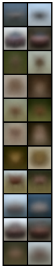

$K = 8$

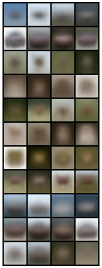

$K = 16$

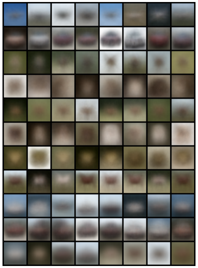


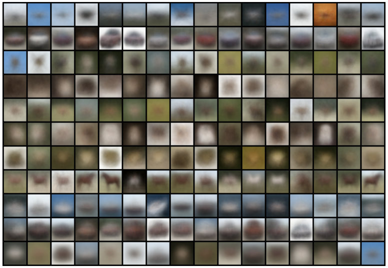
$K = 32$

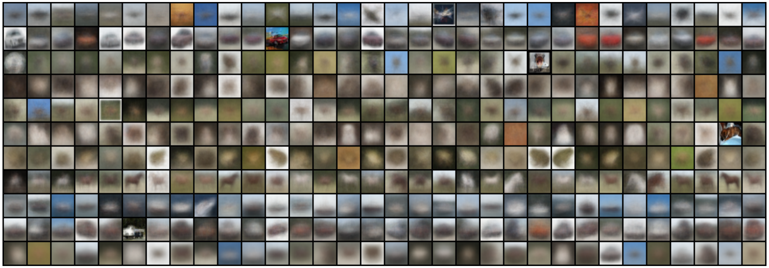

$K = 1$


$K = 2$


$K = 4$


$K = 8$


$K = 16$



$K = 32$

The Principal Component Analysis (PCA) aims also at extracting an information in a L^2 sense. Instead of clusters, it looks for an "affine subspace", *i.e.* a point and a basis, that spans the data.

Given data-points

$$x_n \in \mathbb{R}^D, \ n = 1, \dots, N$$

(A) compute the average and center the data

$$\begin{aligned} \bar{x} &= \frac{1}{N} \sum_n x_n \\ \forall n, \ x_n^{(0)} &= x_n - \bar{x} \end{aligned}$$

The Principal Component Analysis (PCA) aims also at extracting an information in a L^2 sense. Instead of clusters, it looks for an "affine subspace", *i.e.* a point and a basis, that spans the data.

Given data-points

$$x_n \in \mathbb{R}^D,\ n = 1, \dots, N$$

(A) compute the average and center the data

$$\begin{aligned} \bar{x} &= \frac{1}{N} \sum_n x_n \\ \forall n,\ x_n^{(0)} &= x_n - \bar{x} \end{aligned}$$

and then for $t = 1, \dots, D$,

(B) pick the direction and project the data

$$\begin{aligned} v_t &= \underset{\|v\|=1}{\operatorname{argmax}} \sum_n \left(v \cdot x_n^{(t-1)} \right)^2 \\ \forall n,\ x_n^{(t)} &= x_n^{(t-1)} - \left(v_t \cdot x_n^{(t-1)} \right) v_t . \end{aligned}$$

Although this is a simple way to envision PCA, standard implementations rely on an eigendecomposition. With

$$X = \begin{pmatrix} - & x_1 & - \\ & \vdots & \\ - & x_N & - \end{pmatrix}$$

we have

$$\sum_n (v \cdot x_n)^2$$

Although this is a simple way to envision PCA, standard implementations rely on an eigendecomposition. With

$$X = \begin{pmatrix} - & x_1 & - \\ & \vdots & \\ - & x_N & - \end{pmatrix}$$

we have

$$\sum_n (v \cdot x_n)^2 = \left\| \begin{pmatrix} v \cdot x_1 \\ \vdots \\ v \cdot x_N \end{pmatrix} \right\|_2^2$$

Although this is a simple way to envision PCA, standard implementations rely on an eigendecomposition. With

$$X = \begin{pmatrix} — & x_1 & — \\ & \vdots & \\ — & x_N & — \end{pmatrix}$$

we have

$$\begin{aligned} \sum_n (v \cdot x_n)^2 &= \left\| \begin{pmatrix} v \cdot x_1 \\ \vdots \\ v \cdot x_N \end{pmatrix} \right\|_2^2 \\ &= \left\| v X^T \right\|_2^2 \end{aligned}$$

Although this is a simple way to envision PCA, standard implementations rely on an eigendecomposition. With

$$X = \begin{pmatrix} — & x_1 & — \\ & \vdots & \\ — & x_N & — \end{pmatrix}$$

we have

$$\begin{aligned} \sum_n (v \cdot x_n)^2 &= \left\| \begin{pmatrix} v \cdot x_1 \\ \vdots \\ v \cdot x_N \end{pmatrix} \right\|_2^2 \\ &= \left\| v X^T \right\|_2^2 \\ &= (v X^T)(v X^T)^T \end{aligned}$$

Although this is a simple way to envision PCA, standard implementations rely on an eigendecomposition. With

$$X = \begin{pmatrix} — & x_1 & — \\ & \vdots & \\ — & x_N & — \end{pmatrix}$$

we have

$$\begin{aligned}
\sum_n (v \cdot x_n)^2 &= \left\| \begin{pmatrix} v \cdot x_1 \\ \vdots \\ v \cdot x_N \end{pmatrix} \right\|_2^2 \\
&= \left\| v X^T \right\|_2^2 \\
&= (vX^T)(vX^T)^T \\
&= v(X^T X)v^T.
\end{aligned}$$

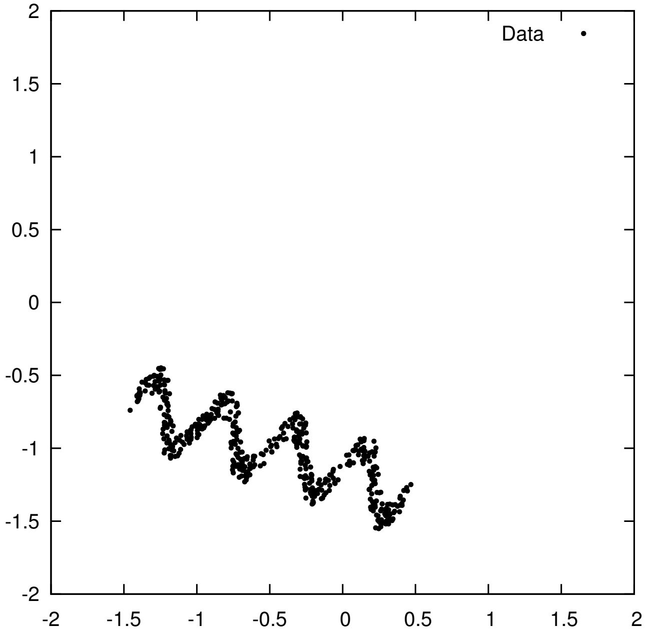
Although this is a simple way to envision PCA, standard implementations rely on an eigendecomposition. With

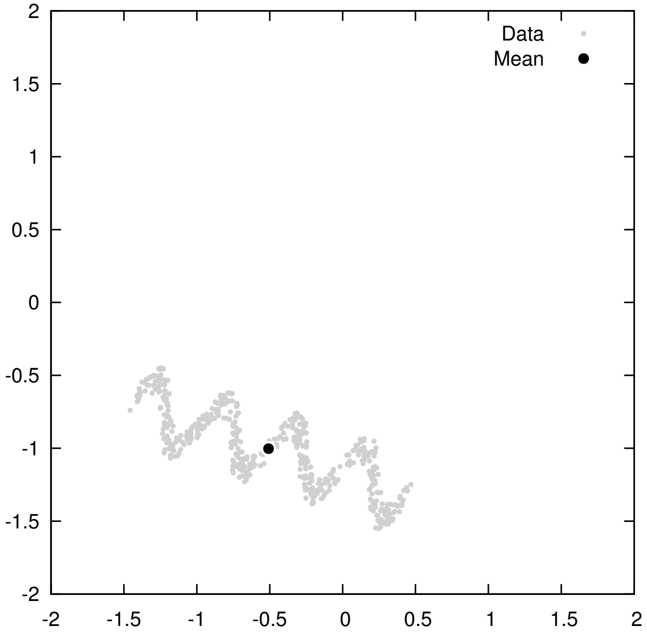
$$X = \begin{pmatrix} — & x_1 & — \\ & \vdots & \\ — & x_N & — \end{pmatrix}$$

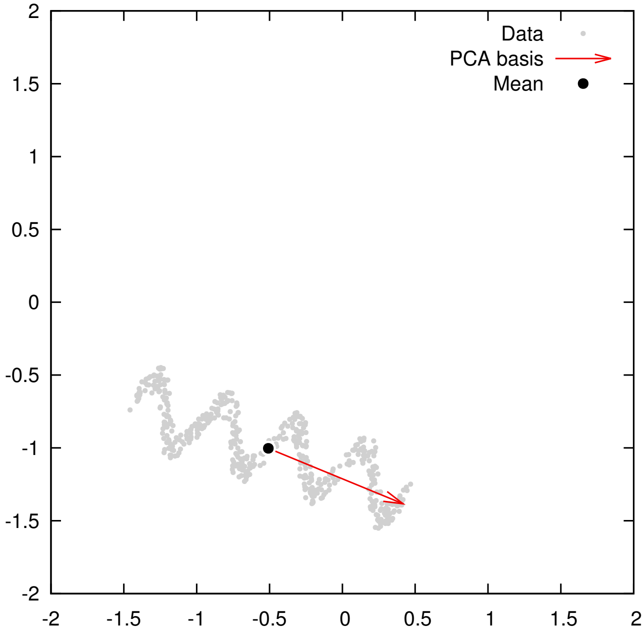
we have

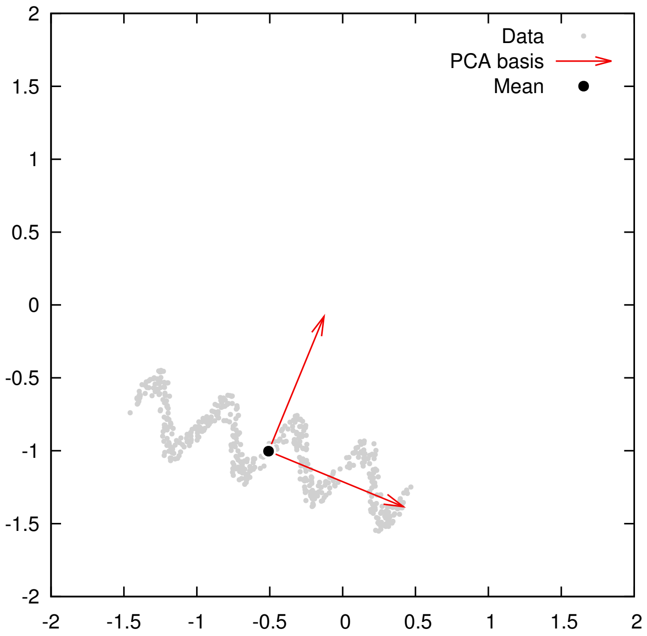
$$\begin{aligned} \sum_n (v \cdot x_n)^2 &= \left\| \begin{pmatrix} v \cdot x_1 \\ \vdots \\ v \cdot x_N \end{pmatrix} \right\|_2^2 \\ &= \left\| v X^T \right\|_2^2 \\ &= (vX^T)(vX^T)^T \\ &= v(X^T X) v^T. \end{aligned}$$

From this we can derive that $v_1, v_2, \dots, v_D$ are the eigenvectors of $X^T X$ ranked according to [the absolute values of] their eigenvalues.


Data
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Data
Mean
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

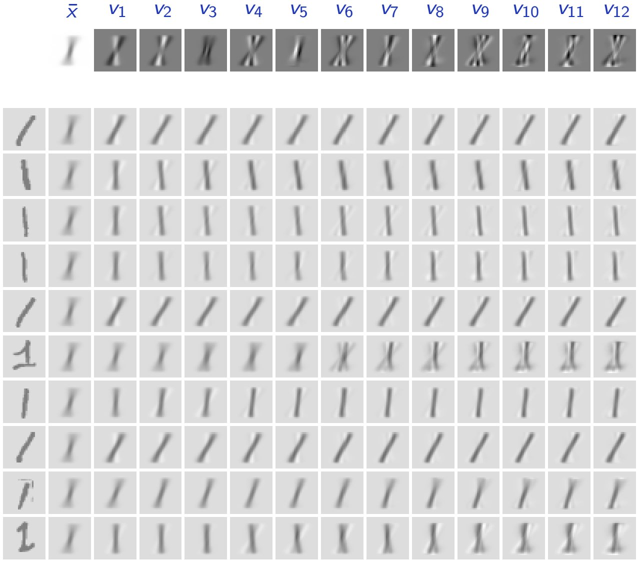

Data
PCA basis
Mean
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

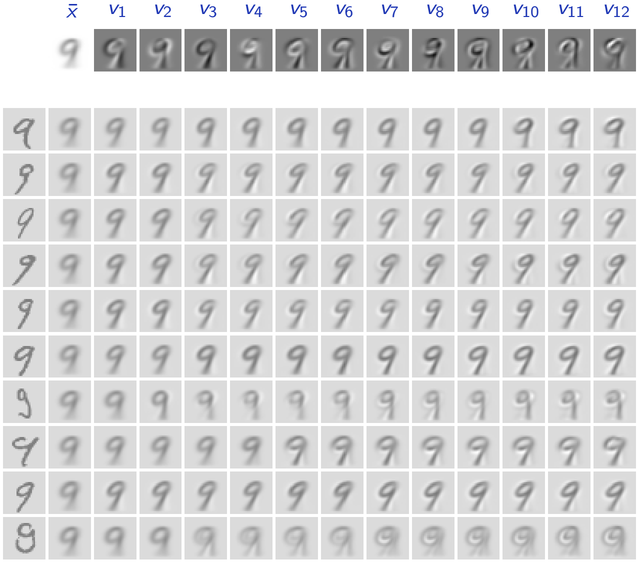

Data
PCA basis
Mean
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

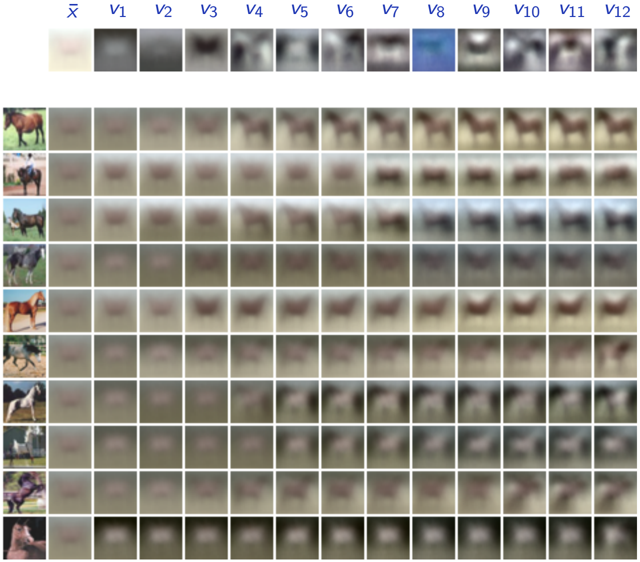
As for K-means, we can apply that algorithm to images from MNIST or CIFAR by considering them as vectors.

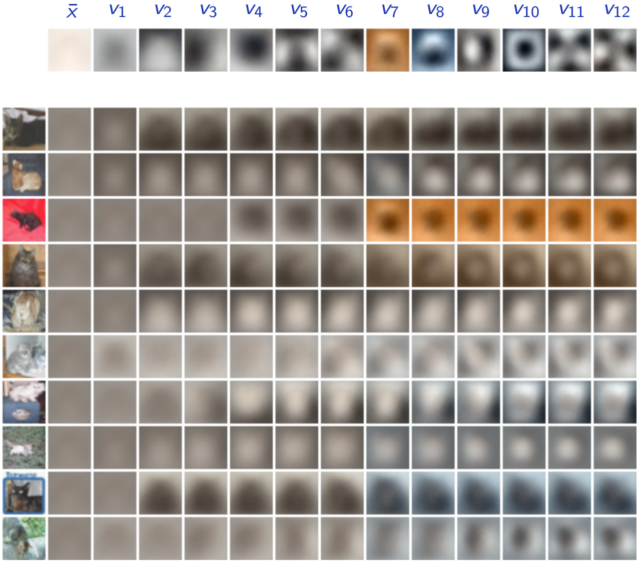
For any sample x and any T, we can compute a reconstruction using T vectors from the PCA basis, *i.e.*

$$\bar{x} + \sum_{t=1}^{T} (v_t \cdot x) v_t.$$


$\bar{x}$
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}


$\bar{x}$
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}


$\bar{x}$
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}


$\bar{x}$
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}

These results show that even crude embeddings capture something meaningful. Changes in pixel intensity as expected, but also deformations in the “indexing” space (*i.e.* the image plan).

However, translations and deformations damage the representation badly. “Composition” (*e.g.* object on background) is not handled at all.

These strengths and shortcomings provide an intuitive motivation for “deep neural networks”, and the rest of this course.

We would like

- to use many encoding “of these sorts” for small local structures with limited variability,
- have different “channels” for different components,
- process at multiple scales.

Computationally, we would like to deal with large signals and large training sets, so we need to avoid super-linear cost in one or the other.

The end

References

A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto, 2009.