Université de Toulouse 3 - Paul Sabatier

Institut de Recherche en Informatique de Toulouse - IRIT

SAMOVA

Neural network vs. statistical methods

-

Comparison of methods for learning sound units using Dictionary Learning and Sparse Coding

---

Author
Thomas Rolland

Supervisor
Thomas Pellegrini
Carine Jauberthie

May 31, 2018

Abstract

Little document to summarize state-of-the-art sparse coding and dictionary learning for weekly supervised and unsupervised feature extraction in speech. Introduction to 1.2 is based on the Hugo Larochelle courses.

This document is a draft

Contents

1 Sparse coding

1.1 Introduction

Idea Sparse dictionary learning is a representation learning method which aims at finding a sparse representation of the input data (in form of a linear combination of basic elements (called Atoms). The idea of using a learned dictionary instead of a predefined one is based on wavelets. The sparse learned models has recently led to state-of-the-art result for denoising, classification,...

Unsupervised learning Only use the inputs $x^{(t)}$ ($X = [x_1, ..., x_n]$ in $\mathbb{R}^{m \times n}$) for learning. Automatically extract meaningful features of our data, leverage the availability of unlabeled data and add a data-dependent regularize to trainings.

Sparse coding is one of the methods used for unsupervised learning (like restricted Boltzmann machines and autoencoders).
The idea behind sparse coding is: For each x^t find a latent representation α^t such that:

- It is sparse: the vector α^t has many zeros (only few nonzero elements)
- We can reconstruct the original input $x^{(t)}$ as well as possible.

That mean, more formally:

$$\min_D \frac{1}{T} \sum_{t=1}^{T} \min_{\alpha^{(t)}} \frac{1}{2} \|x^{(t)} - D\ \alpha^{(t)}\|_2^2 + \lambda \|\alpha^{(t)}\|_1$$

- D is a matrix of weights, usually refer to that matrix as a dictionary matrix (containt atoms) with $D \in \mathbb{R}^{m \times k}$ (k the number of atoms)
- $\|x^{(t)} - D\ \alpha^{(t)}\|_2^2$ is the reconstruction error
- $D\ \alpha^{(t)}$ is the reconstruction of $\hat{x}^{(t)}$
- $\|\alpha^{(t)}\|_1$ is the sparsity penalty (more 0 in h we have, better it is)

This two objectives fight each other. But it still a optimization problem (cf min), and we'll try to optimize it for each training example $x^{(t)}$. This is why we have a sum over all the training examples.

We also constrain the columns of D to be of norm 1 (otherwise, D could grow big while $\alpha^{(t)}$ becomes small to satisfy the prior). And sometimes the columns are constrained to be no greather than 1.

However, $\alpha^{(t)}$ is now a complicated function of $x^{(t)}$:
Encoder is the minimization $\alpha(x^{(t)}) = arg\min_{\alpha^{(t)}} = \frac{1}{2} \|x^{(t)} - D\ \alpha^{(t)}\|_2^2 + \lambda \|\alpha^{(t)}\|_1$, so the optimization problem is more complicated than a simple non linear problem.
The idea to solve this minimization problem is [**?**]

```
while D not_converged :
    Fix D
    Minimize alpha      (1) // Sparse Coding step
    Fix alpha
    Minimize D            (2) // Update Dictionary step
```

But there are some improvement like K-SVD algorithm [**?**], which compute column by column a SVD computation over the relevant examples.

Dictionary We can also write $\hat{x}^{(t)} = D\ \alpha(x^{(t)}) \sum_{\substack{k s.t. \\ \alpha(x^{(t)})_k \neq 0}} D_{.,k}\, \alpha(x^{(t)})_k$

The images refer to $D_{.,k}$ (columns of D wich are not equals to 0) and the factor (1 or 0.8 in this case) refer to $\alpha(x^{(t)})_k$
We also refer to D as the dictionary:

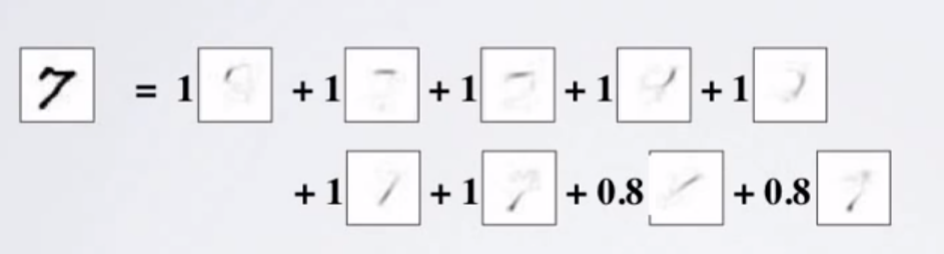


Figure 1: Example of reconstruction using sparse coding

- in certain applications, we know what dictionary matrix to use
- often however, we have to learn it

In general we have $k << n$. But we can use an overcomplete dictionary with $k > m$.

1.2 Inference of Sparse code

The original problem is a combinatorial problem (proven to be NP-hard). To solve this problem we use relaxation methods (then we can smooth the L_0 and use continuous optimization techniques) or greedy methods (then build the solution one non-zero element at time).

1.2.1 Compute α

Idea Here we develop step (1) of our algorithm.
Assume we are given a dictionary matrix D, how do we compute $h(x^{(t)})$. We have to optimize:

$$\text{Basic Pursuit: } l(x^{(t)}) = \tfrac{1}{2}\|x^t - D\alpha^{(t)}\|_2^2 + \lambda\|\alpha^{(t)}\|_1 w.r.t.\alpha^{(t)}$$

Here we used relaxation method to switch from norm l_0 to l_1 know as the Basic Pursuit (vs Matching Pursuit, a greedy method, if we keep l_0 norm and find one atom at a time).
We could use a gradient descent method to solve this minimization:

$$\Delta_{\alpha^{(t)}} l(x^{(t)}) = D^T(D\alpha^{(t)} - x^{(t)}) + \lambda sign(\alpha^{(t)})$$

The issue is l_1 norm is not differentiable at 0. The solution is : if $\alpha^{(t)}$ changes sign because of l_1 norm gradient then clamp to 0.That mean :
$\alpha_k^{(t)} = \alpha_k^{(t)} - \alpha(D_{.,k})^T(D\alpha^{(t)} - x^{(t)})$
if $\text{sign}(\alpha_k^{(t)}) \neq \text{sign}(\alpha_k^{(t)} - \alpha\lambda\ \text{sign}(\alpha_k^{(t)}))$ then: $\alpha_k^{(t)} = 0$
else $\alpha_k^{(t)} = \alpha_k^{(t)} - \alpha\lambda\ \text{sign}(\alpha_k^{(t)})$

ISTA (Iterative Shrinkage and Thresholding Algorithm) :

```
initialize h
while h not_converged:
    for each h_k in h:
        h_k = h_k -alpha * transpose(D[:,k]) * (D*h - x)
        h_k = shrink(h_k, alpha*lambda_coef)
return h
```

Here **shrink(a,b)** = [..., $\text{sign}(a_i)\ \max(|a_i| - b_i, 0)$, ...]

1.2.2 Compute D

There are three algorithms used for dictionary update.

Algorithm 1: A gradient descent method Our original problem is:

$$\min_D \frac{1}{T}\sum_{t=1}^{T} \min_{\alpha^{(t)}} \frac{1}{2}\|x^{(t)} - D\,\alpha^{(t)}\|_2^2 + \lambda\|\alpha^{(t)}\|_1$$

But here we assume $\alpha(x^{(t)})$ doesn't depend on D. So we must minimize:

$$\min_D \frac{1}{T}\sum_{t=1}^{T} \min_{\alpha^{(t)}} \frac{1}{2}\|x^{(t)} - D\,\alpha^{(t)}\|_2^2$$

```
while D not_converged:
    # Perform gradient update of D
    D = D - alpha * (1/T)* sum((x - D h)* tranpose(h))
    # Renormalize the columns of D
    for each column D[:,j]:
        D[:,j] = (D[:,j] / norm(D[:,j]))

return D
```

Algorithm 2: Block-coordinate descent We must minimize:

$$\min_D \frac{1}{T}\sum_{t=1}^{T} \min_{\alpha^{(t)}} \frac{1}{2}\|x^{(t)} - D\,\alpha^{(t)}\|_2^2$$

The idea is to solve for each column $D_{.,j}$ in cycle (that mean to optimize in one direction at time). For that we must set the gradient for $D_{.,j}$ to zero.
We have:

$$0 = \frac{1}{T}\sum_{t=1}^{T}(x^{(t)} - Dh(x^{(t)}))\ \alpha_j^{(t)}$$

We separe $D_{.,j}$ from the rest of D:

$$0 = \frac{1}{T}\sum_{t=1}^{T}(x^{(t)} - (\sum_{i\neq j} D_{.,i}\ h(x^{(t)})_i\ ) - (D_{.,j}\ h(x^{(t)})_j)\ )\ \alpha_j^{(t)}$$

Our aim is to find the value of $D_{.,j}$, we must isolate $D_{.,j}$:

$$0 = \frac{1}{T}\sum_{t=1}^{T}(x^{(t)}\alpha_j^{(t)} - (\sum_{i\neq j} D_{.,i}\ \alpha_i^{(t)}\ \alpha_j^t) - (D_{.,j}\ \alpha_j^{(t)2}))$$

$$0 = (\sum_{t=1}^{T}(x^{(t)}\alpha_j^{(t)} - (\sum_{i\neq j} D_{.,i}\ \alpha_i^{(t)}\ \alpha_j^t)\ ) - (\sum_{t=1}^{T}(D_{.,j}\ \alpha_j^{(t)2})))$$

$$\sum_{t=1}^{T}(D_{.,j}\ \alpha_j^{(t)2}) = \sum_{t=1}^{T}(x^{(t)}\alpha_j^{(t)} - (\sum_{i\neq j} D_{.,i}\ \alpha_i^{(t)}\ \alpha_j^t)\ )$$

$$D_{.,j}\sum_{t=1}^{T}\alpha_j^{(t)2} = \sum_{t=1}^{T}(x^{(t)}\alpha_j^{(t)} - (\sum_{i\neq j} D_{.,i}\ \alpha_i^{(t)}\ \alpha_j^t)\ )$$

$$D_{.,j} = \frac{1}{\sum_{t=1}^{T}\alpha_j^{(t)2}}\sum_{t=1}^{T}(x^{(t)}\alpha_j^{(t)} - (\sum_{i\neq j} D_{.,i}\ \alpha_i^{(t)}\ \alpha_j^t)\ )$$

$$D_{.,j} = \underbrace{\frac{1}{\sum_{t=1}^{T}\alpha_j^{(t)2}}}_{A_{j,j}}\underbrace{\sum_{t=1}^{T}(x^{(t)}\alpha_j^{(t)})}_{B_{.,j}} - \sum_{i\neq j} D_{.,i}(\underbrace{\sum_{t=1}^{T}\alpha_i^{(t)}\alpha_j^t)}_{A_{i,j}}$$

$$D_{.,j} = \frac{1}{A_{j,j}}(B_{.,j} - DA_{.,j} + D_{.,j}A_{j,j})$$

```
while D not_converged:
    # For each column D[:,j] perform updates
    for each column D[:,j]:
        D[:,j] = (1/A[j, j])*(B[:, j] - D A[:, j] + D[:, j] A[j, j])
        # Normalization
        D[:,j] = D[:,j]/norm(D[:,j])

return D
```

Algorithm 3: Online learning algorithm For large datasets we want to update D after visiting each $x^{(t)}$. The solution is for each $x^{(t)}$ [**?**] :

- Perform inference of $h(x^{(t)})$ after visiting each $x^{(t)}$
- Update running averages of the quantities required to update D:
  - B = $\beta B + (1-\beta)x^{(t)}\alpha(x^{(t)})^T$
  - A = $\beta A + (1-\beta)h(x^{(t)})\alpha(x^{(t)})^T$
- Use current value of D as " warm start" to block-coordinate descent (warm start $\iff$ With the previous value of D)

(We have to specifie β like a learning rate α (NB: this α isn't sparse matrix, it's a learning rate coefficient) in the gradient descent)

```
Initialize D # Not to 0 ! (To respect the constraint we define before)
while D not_converged:
    for each x:
        Infer code h
        #Update dictionary
        A = A +  h * transpose(h)
        B = B + x * transpose(h)
        #Batch upgrade
        #A = beta * A + ( 1 - beta ) * h * transpose(h)
        #B = beta * B + ( 1 - beta ) * x * transpose(h)
        while D not_converged:
            for each column D[:,j]:
                D[:,j] = (1/A[j,j])*(B[:,j] - D A[:,j] + D[:,j] A[j,j])
                # Normalization
                D[:,j] = D[:,j]/norm(D[:,j])
```

Optimizing the Algorithm In practice, it's possible to improve the convergence speed of this algorithm by using a Mini-batch extension: By drawing $\eta > 1$ signals at each iteration instead of a single one.

$$\begin{cases} A_t = \beta A_{t-1} + \sum_{i=1}^{\eta} \alpha_{t,i}\alpha_{t,i}^T \\ B_t = \beta B_{t-1} + \sum_{i=1}^{\eta} x\alpha_{t,i}^T \end{cases}$$

Then $\beta = \frac{\theta+1-\eta}{\theta+1}$, where $\theta = t\eta$ if $t < \eta$ and $\eta^2 + t - \eta$ if $t \geq \eta$

1.3 Application for MNIST dataset

The MNIST database of handwritten digits, available from Yann Lecun's website. MNIST has a training set of 60,000 examples and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centred in a fixed-size image. In my test, I'll use 55000 examples from the training set (using Tensorflow datasets). These are 28×28 images. One way to evaluate the quality of our results is to compare the original data vs the reconstructed ones.

1.3.1 Prototype

My first task is to realise a Sparse Coding prototype to compute Sparse Coding on this dataset, using Python. The aim here is to understand the underlying principles of this method, you can found this prototype in `Code directory` of this repository as `SparseCoding.py`.
These are some results of this prototype: For time-saving, I used only 100 digits as input.

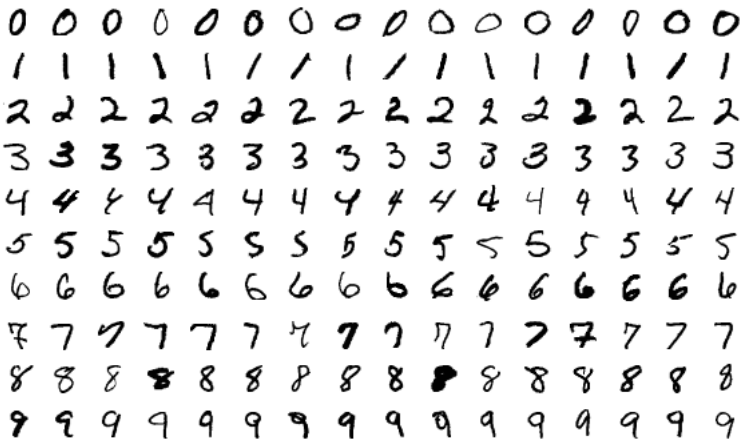


Figure 2: Example of MNIST's handwitten digits

1.3.2 SPAMS

SPAMS (SPArse Modeling Software) is an optimization toolbox for solving various sparse estimation problems.

- Dictionary learning and matrix factorization (NMF, sparse PCA, ...)
- Solving sparse decomposition problems with LARS, coordinate descent, OMP, SOMP, proximal methods
- Solving structured sparse decomposition problems (l1/l2, l1/linf, sparse group lasso, tree-structured regularization, structured sparsity with overlapping groups,...).

It is developed and maintained by Julien Mairal (Inria), and contains sparse estimation methods resulting from collaborations with various people: notably, Francis Bach, Jean Ponce, Guillermo Sapiro, Rodolphe Jenatton and Guillaume Obozinski.
You can find my code of Sparse Coding/Dictionary Learning on MNIST using SPAMS toolbox on my GitHub `test_spams.py`.

Test 1 In the first test, I used 256 atoms, 2 000 iterations and $\lambda = 0.015$ to learn the dictionary and the sparse coefficients. w

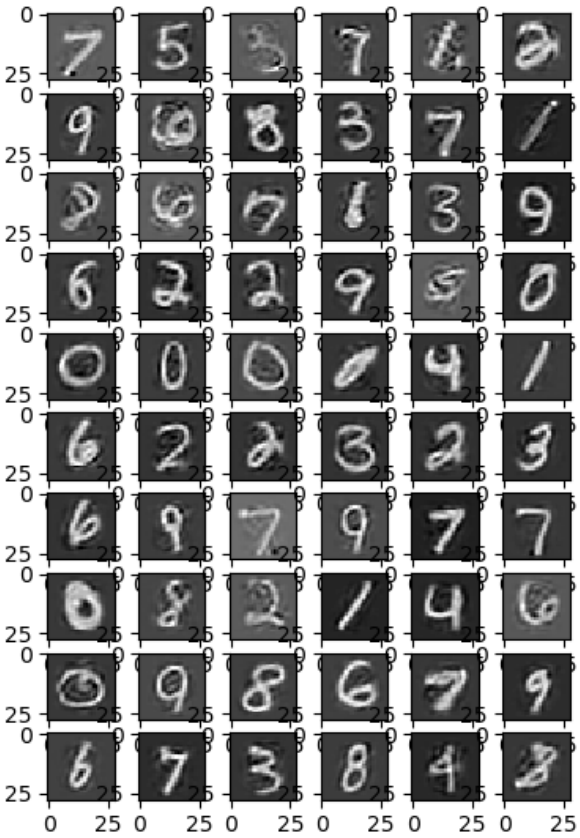
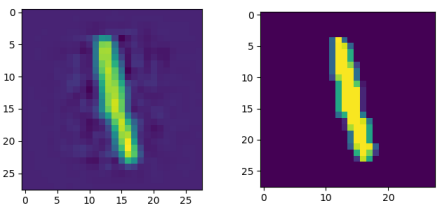
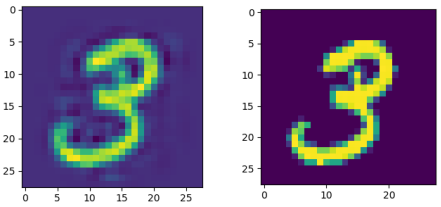


Figure 3: Few atoms of D



(a) Reconstructed 1 vs Original 1



(b) Reconstructed 3 vs Original 3

Test 2 In the second test I used 1024 atoms, 1 000 iterations and $\lambda = \frac{1.2}{\sqrt{m}}$ [**?**] *(In my case* ≈ 0.0042857*).*

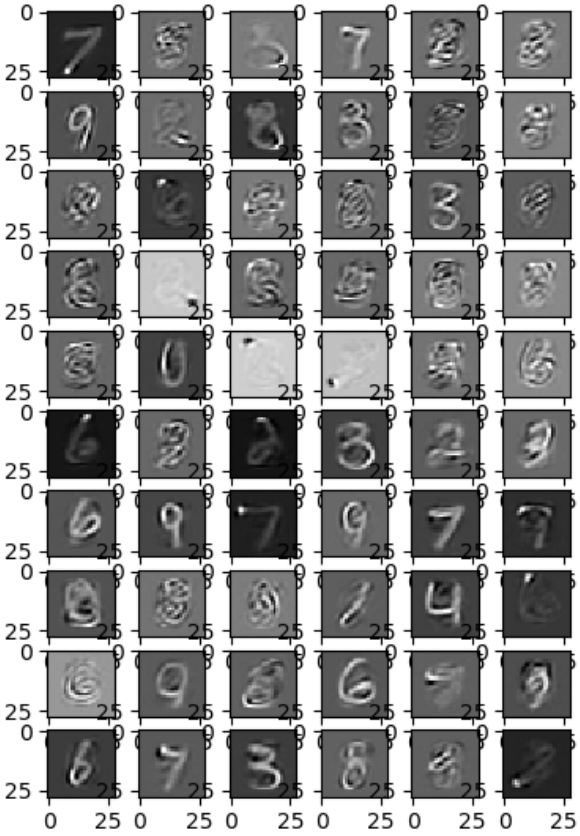
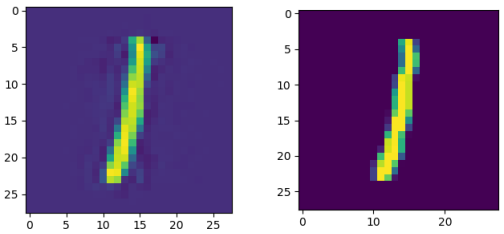
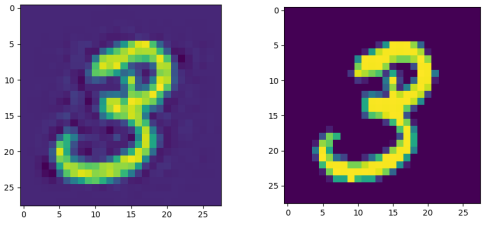


Figure 5: Few atoms of D



(a) Reconstructed 1 vs Original 1



(b) Reconstructed 3 vs Original 3

Test 3 In the third test I used 1024 atoms, 1 000 iterations and $\lambda = 5$.

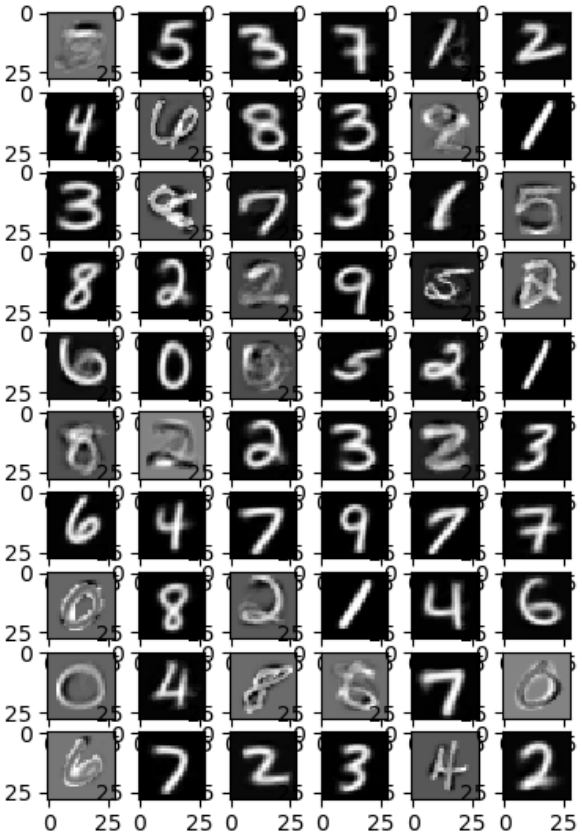
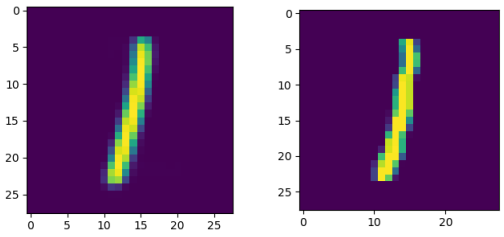
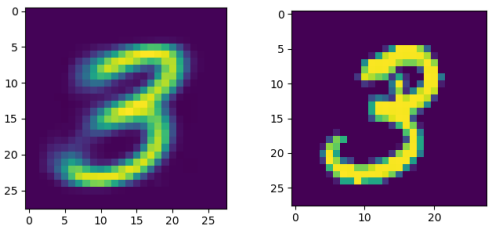


Figure 7: Few atoms of D



(a) Reconstructed 1 vs Original 1



(b) Reconstructed 3 vs Original 3

Test 4 In the fourth test I used 2048 atoms, 1 000 iterations and $\lambda = \frac{1.2}{\sqrt{m}}$

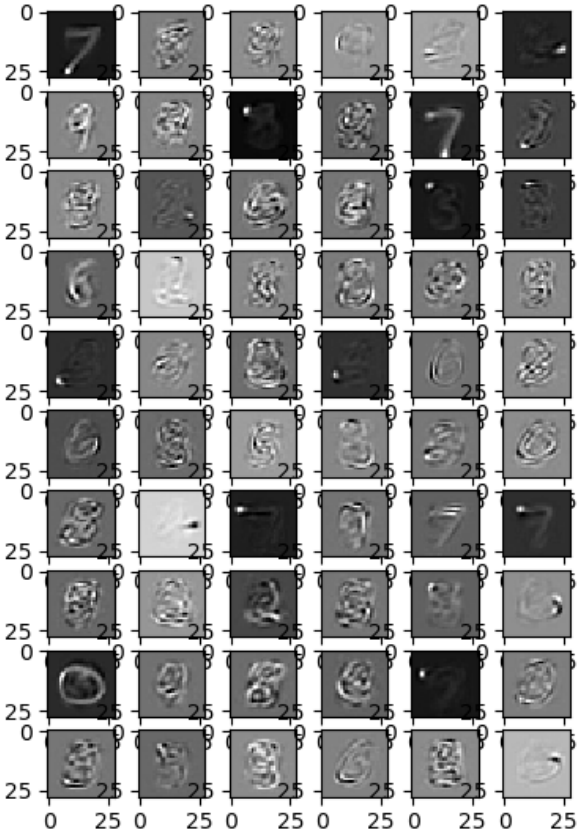
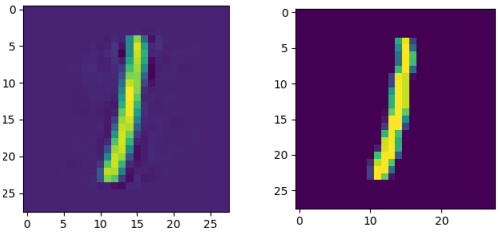
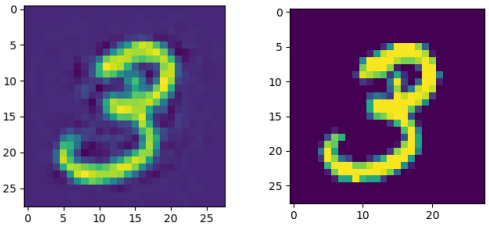


Figure 9: Few atoms of D



(a) Reconstructed 1 vs Original 1



(b) Reconstructed 3 vs Original 3

1.4 Application for Lenna

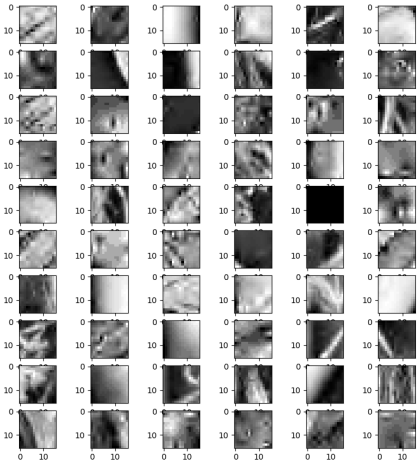


Figure 11: Some atoms of D when K = 1024

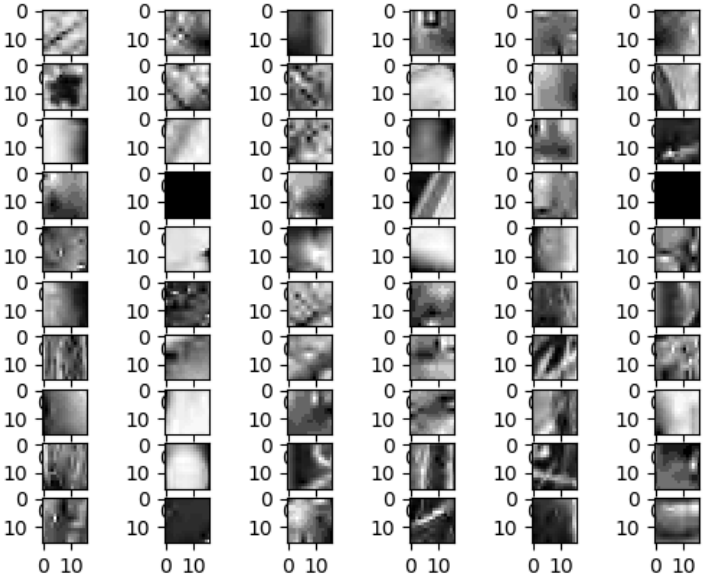


Figure 12: Some atoms of D when K = 2048

1.5 Dictionary learning and sparse coding for unsupervised clustering

Whereas the previous tests seem to have a good result, one question appears
What makes us confident about the fact that two close images (two handwritten 3 for example) have close coefficients representation h ?
Sprechmann and Sapiro [**?**] propose an algorithm to cluster datasets that are well represented in the sparse modelling framework with a set of K learned dictionaries. The main idea is, given a set of K dictionaries, find for each signal in the dictionary for which the "best" sparse decomposition is obtained, with :

$$\min_{D_i,C_i} \sum_{i=1}^{K} \sum_{x_j \in C_i} \mathcal{R}(x_j, D_i)$$

Here $D_i \in \mathbb{R}^{n \times k_i}$ is the k_i dictionary associated with the class C_i. $x_j \in \mathbb{R}^n$ are the input data and $\mathcal{R}$ a function that mesure how good the sparse decomposition is for the signal x_j under the dictionary D_i. Sprechmann and Sapiro propose to use the cost function in the Lasso-type problem as $\mathcal{R}$ the measure of performance, $\mathcal{R}(x, D) = \|x - D\alpha\|_2^2 + \lambda\|\alpha\|_1$. The class $\mathcal{C}$ for a given signal x is found by solving $\mathcal{C} = \arg\min_{j=1,..,K} \mathcal{R}(x, Dj)$.

1.5.1 Dictionary learning for clustering

Given a set of signals and number of classes, we want to find a set of K learned dictionaries that best represent x (the input data). [**?**] formulate thus as an energy minimization problem and use the measure previously proposed,

$$\min_{D_i,C_i} \sum_{i=1}^{K} \sum_{x_j \in C_i} \min_{\alpha_{ij}} \|x_j - D_i\alpha_{ij}\|_2^2 + \lambda\|\alpha_{ij}\|_1$$

The optimization is carried out by solving one problem at time:

- *Assignement step:* The dictionaries are fixed and each signals is assigned to the cluster for which the best representation is obtained.
- *Update step:* The new dictionaries are computed fixing the assignation found in the previous step.

One drawback of this algorithm is there is no guarantee of reach a global minimum. In this setting, repeated initialization are computationally expensive, thus we need a good initialization.

1.5.2 Initialization

The initialization can be given by a set of K dictionaries or as an initial partition of the data.
The main idea is to construct a similarity matrix and use it as the input for a spectral clustering algorithm. Let define $A = [\alpha_1, ..., \alpha_m]$ with α_j the sparse representation of each signal x_j. To obtain a good classification, we expect two signal to the same cluster to have decomposition that uses similar atoms. Thus we can compute two similarity matrix:

- *Clustering the signals :* Construct a similarity matrix $S_1 \in \mathbb{R}^{m \times m}$ which measure the similarity of two signals by comparing the corresponding sparse representation:
  $S_1 = |A|^T|A|$
- *Clustering the atoms :* Construct a similarity matrix $S_2 \in \mathbb{R}^{k_0 \times k_0}$ (with $D_0 \in \mathbb{R}^{n \times k_0}$) which represent the similarity of two atoms by comparing how many signals use them simultaneously and how they contribute in their sparse decomposition.
  $S_2 = |A||A|^T$

In this two case, the similarity matrixes are positive semidefinite and can be associated with a graph: $G_1 = \{X, S_1\}$ and $G_2 = \{D, S_2\}$ where the data (respectivly atoms) are the sets of vertexes with the corresponding S_i as edge weights matrixes. This graph is partitioned using standard spectral clustering algorithm to obtain the initialization.
However, when K is large (the number of class), the performance of initial clusterization decreases. To fix this problem [**?**] proposed to stat with the whole set as the only partition and at each itation we subdivise in two sets each of the current partitions, the procedure stops when the desired number of clusters is reached.

2 Discriminative Dictionary

There are serveral approach for discriminative Dictionary Learning enumerate by [**?**]:

- In presence of label:
  1. Learn one dictionary per class
  2. Prune large dictionaries
  3. Jointly learn dictionary and classifier
  4. Embed class label into the learning of sparse coefficients
  5. Learn a histogram of dictionary element over signal constituents
- For weakly supervised:
  Serveral max margin based, non-convolutive, syntgesis dictionary learning approaches.

However in our problem of extract new features for speech we must compare each α with other, thus we must use only one dictionary.

2.1 One dictionary per class

The idea is to extend proposed method in section 1.5 by adding a term $\mathcal{Q}(D_i, D_j)$ that promotes incoherence between the differents dictionaries, e.g. this term encourage dictionaries to be independent as possible :

$$\min_{D_i, C_i} \sum_{i=1}^{K} \sum_{x_j \in C_i} \mathcal{R}(x_j, D_i) + \eta \sum_{i \neq j} \mathcal{Q}(D_i, D_j)$$

For example we can take $\mathcal{Q}(D_i, D_j) = \|D_i^T D_j\|_F^2$ with F denotes Frobenius norm.

2.2 Supervised Dictionary learning SDL

2.2.1 Problem formulation

In [**?**] the signal may belong to any of p diffrent classes and they model the signal using a single shared D. They create a set of p decision functions $g_i(x, \alpha, \theta)$ (i = 1,...,p). Where:

$$g_i(x, \alpha, \theta) = \begin{cases} gi > 0 \text{ if } x \in \text{class i} \\ g_i \leq 0 \text{ otherwise} \end{cases}$$

The vector θ parametrizes the model and will be jointly learned with the dictionary D. There are two kinds of models in this paper:

- Linear in α: $g_i(x, \alpha, \theta) = w_i^T \alpha + b_i$ where $\theta = \{w_i \in \mathbb{R}^k, b_i \in \mathbb{R}\}_{i=1}^p$
- Bilinear in x and α: $g_i(x, \alpha, \theta) = x^T W_i \alpha + b_i$ where $\theta = \{W_i \in \mathbb{R}^{n \times k}, b_i \in \mathbb{R}\}_{i=1}^p$

They define a *softmax* discriminative cost function as :

$$\mathcal{C}_i(x_1, ..., x_p) = log(\sum_{i=1}^{p} e^{x_j - x_i})$$

Given x, a input signal, with D and θ fixed, the supervised sparse coding problem for the class p can be computing by :

$$S_i^*(x, D, \theta) = \min_{\alpha} S_i(\alpha, x, D, \theta)$$

where

$$S_i(\alpha, x, D, \theta) = \mathcal{C}_i(\{g_j(x, \alpha, \theta)\}_{j=1}^p) + \lambda_0 \|x - D\alpha\|_2^2 + \lambda_1 \|\alpha\|_1$$

Then, the classification problem can be compute by:

$$i^*(x, D, \theta) = \arg\min_{i=1,...,p} S_i^*(x, D, \theta)$$

2.2.2 Learning D and θ

The most direct method for learning D and θ is to minimize with respect to these (with T_i a sample of input signals corresponding to the class i):

$$\min_{D,\theta}(\sum_{i=1}^{p}\sum_{j\in T_i} S_i^*(x_j, D, \theta)) + \lambda_2\|\theta\|_2^2$$

With $\|\theta\|_2^2$ to prevent overfitting. They reffer to this model as SDL-G (Supervised Dictionary Learning - Generative) [**?**].
A more discriminative approach is not only make S_i^* small for signals with label i but also make the value of S_j^* (with $i \neq j$) greater than S_i^*. To do that they use the softmax cost function $\mathcal{C}_i$:

$$\min_{D,\theta}(\sum_{i=1}^{p}\sum_{j\in T_i} \mathcal{C}_i(\{S_l^*(x_j, D, \theta)\}_{l=1}^{p})) + \lambda_2\|\theta\|_2^2$$

But this more difficult to solve, thus they adopt a mixed formulation with SDL-G :

$$\min_{D,\theta}(\sum_{i=1}^{p}\sum_{j\in T_i} \mu\mathcal{C}_i(\{S_l^*(x_j, D, \theta)\}_{l=1}^{p}) + (1-\mu)S_i^*(x_j, D, \theta)) + \lambda_2\|\theta\|_2^2$$

They refer to this model as SDL-D (Supervised Dictionary Learning - Discriminative). With μ which control the trade-off between reconstruction and discrimination.

2.2.3 Optimization procedure

SDL-G When $\mu = 0$ or directly SDL-G have the same properties than classical dictionary learning techniques: Using block coordinate descent consist of iterating between *supervised sparse coding*, where D and θ are fixed and optimize α, and *supervised dictionary update*, where α is fixed but D and θ are updated.

SDL-D However the discriminative version of SDL (where $\mu \neq 0$) is not convex (even when D and θ or α are fixed). To reach a local minimum for this problem, they have chosen a continuation method: Starting from the generative case and ending with the discriminative one.

Input: p (number of classes); n (signal dimensions); $\{T_i\}_{i=1}^{p}$ (training signals); k (size of the dictionary); $\lambda_0, \lambda_1, \lambda_2$ (parameters); $0 \leq \mu_1 \leq \mu_2 \leq \ldots \leq \mu_m \leq 1$ (increasing sequence);
Output: $\mathbf{D} \in \mathbb{R}^{n\times k}$ (dictionary); $\boldsymbol{\theta}$ (parameters).
Initialization: Set $\mathbf{D}$ to a random Gaussian matrix. Set $\boldsymbol{\theta}$ to zero.
Loop: For $\mu = \mu_1, \ldots, \mu_m$,
Loop: Repeat until convergence (or a fixed number of iterations),
• *Supervised sparse coding:* Compute, for all $i = 1, \ldots, p$, all j in T_i, and all $l = 1, \ldots, p$,

$$\boldsymbol{\alpha}_{jl}^{\star} = \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^k} \mathcal{S}_l(\boldsymbol{\alpha}, \boldsymbol{x}_j, \mathbf{D}, \boldsymbol{\theta}). \quad (10)$$

• *Dictionary update:* Solve, under the constraint $||\mathbf{d}_l|| \leq 1$ for all $l = 1, \ldots, k$

$$\min_{\mathbf{D},\boldsymbol{\theta}} \Big(\sum_{i=1}^{p}\sum_{j\in T_i} \mu\mathcal{C}_i(\{\mathcal{S}_l(\boldsymbol{\alpha}_{jl}^{\star}, \boldsymbol{x}_j, \mathbf{D}, \boldsymbol{\theta})\}_{l=1}^{p}) + (1-\mu)S_i(\boldsymbol{\alpha}_{ji}^{\star}, \boldsymbol{x}_j, \mathbf{D}, \boldsymbol{\theta}) \Big) + \lambda_2||\boldsymbol{\theta}||_2^2. \quad (11)$$

Figure 13: SDL: Supervised dictionary learning Algorithm [**?**]

2.3 Supervised Dictionary learning LC-KSVD

[**?**] propose a supervised learning algorithm to learn a compact and discriminative dictionary for sparse coding using a new label consistency constraint ("discriminative sparse-code error") and combined reconstruction error and classification error to form a unified objective function which can be solved with the K-SVD algorithm.

2.3.1 Dictionary Learning for classification

A good classifier $f(x)$ can be obtained by determining its model parameters $W \in R^{m\times K}$ satisfying:

$$W = \arg\min_{W} \sum_{i} \mathcal{L}\{h_i, f(x_i, W)\} + \lambda_1\|W\|_F^2$$

where: m is the number of classes, $\mathcal{L}$ is the classification loss function (typically logistic loss function, suqare hinge loss, simple quadratic loss function), h_i the label of y_i and λ_1 a regularization parameters (which prevents overfitting).
The objective function for learning D and W jointly can be :

$$< D, W, \alpha >= \arg\min_{D,W,\alpha} \|X - D\alpha\|_2^2 + \sum_i \mathcal{L}\{h_i, f(\alpha_i, W)\} + \lambda_1 \|W\|_F^2 \text{ s.t. } \forall i, \|\alpha_i\|_0 \leq T$$

The sparse code α_i can be used as a feature descriptor of input signal x_i, then the risk minimization formulation can be written as:

$$< D, W >= \arg\min_{D,W} \sum_i \mathcal{L}\{h_i, f(\alpha^*(x_i, D), W)\} + \frac{\nu_1}{2} \|W\|_F^2$$

Here D is not explicitly defined in the function but implicitlu in the sparse coding step ($\alpha^*(x_i, D)$)

2.3.2 Label Consistent K-SVD (LC-KSVD)

[?] add a mabem consistency regularization term and a joint classification error and label consistency regularization term into the objective function :

$$< D, \alpha >= \arg\min_{D,\alpha} \|X - D\alpha\|_2^2 \text{ s.t. } \forall i, \|\alpha_i\|_0 \leq T$$

They refer to this two problem as LC-KSVD1 and LC-KSVD2 respectivly.

2.3.3 LC-KSVD1

The objective function for dictionary construction is defined as

$$< D, A, \alpha >= \arg\min_{D,A,\alpha} \|X - D\alpha\|_2^2 + \mu \|Q - A\alpha\|_2^2 \text{ s.t. } \forall i, \|\alpha_i\|_0 \leq T$$

Where μ constrols the contribution between reconstruction and label consistency regularization. $Q = [q_1, ..., q_N] \in \mathbb{R}^{K \times N}$ are the "discriminative" sparse codes of input signal X for classification.
Q is a discriminative sparse code corresponding to an input signal and the dictionary atoms, the nonzero values occur when $signal_i$ and $atom_i$ share the same label. To use K-SVD algorithm we can rewritte

Atomes \ Signaux	signal 1	signal 2	signal 3	signal 4	signal 5	signal 6
k1	0	1	0	1	0	0
k2	0	1	0	1	0	0
k3	1	0	1	0	0	1
k5	1	0	1	0	0	1
k6	0	0	0	0	1	0
k7	0	0	0	0	1	0
k8	0	0	0	0	0	0

Figure 14: Example of Q, each color correspond to a class (white color is the lack of classes)

our previous equation:

$$< D, A, \alpha >= \arg\min_{D,A,\alpha} \| \begin{pmatrix} X \\ \sqrt{\mu} Q \end{pmatrix} - \begin{pmatrix} D \\ \sqrt{\mu} A \end{pmatrix} \alpha \|_2^2$$

$$\text{s.t. } \forall i, \|\alpha_i\|_0 \leq T$$

We define $X_{new} = \begin{pmatrix} X \\ \sqrt{\mu} Q \end{pmatrix}$ and $D_{new} = \begin{pmatrix} D \\ \sqrt{\mu} A \end{pmatrix}$. The matrix D_{new} is L_2 normalized.We can solve our minimization problem using K-SVD algorithm on:

$$< D, A, \alpha >= \arg\min_{D,A,\alpha} \|X_{new} - D_{new}\alpha\|_2^2$$

$$\text{s.t. } \forall i, \|\alpha_i\|_0 \leq T$$

But we cannot use D and A directly after employing K-SVD algorithm because D and A are L_2 normalized in D_{new}. The desired $\hat{D} and \hat{A}$ can be compute as follow:

$$\hat{D} = (\frac{d_1}{\|d1\|_2} ... \frac{d_K}{\|d_K\|_2}) \text{ and } \hat{A} = (\frac{a_1}{\|d_1\|_2} ... \frac{a_K}{\|d_K\|_2})$$

```
Input: X, Q, lambda1, K
Output: D, A
================================================================
D_0 = K-SVD(X, lambda1)
A_0 = Q * tranpose(X) * inv(X*tranpose(X) + lamba2 * Id)
h_0 = omp(Dnew, Xnew, lambda1)

Initialize Xnew and Dnew
D = K-SVD(Dnew, Xnew, lambda1)

Extract D, A
return D, A
```

With $A_0 = QX^T(XX^T + \lambda_2 I)^{-1}$

In our case we don't want to train the classifier, we only want to find new features. This is why we only apply LC-KSVD1.
You can find my Python code of LC-KSVD1 on MNIST in my Github repository: `LC-KSVD.py`

2.4 Application on MNIST

For k = 1024 and $\mu = 5$:

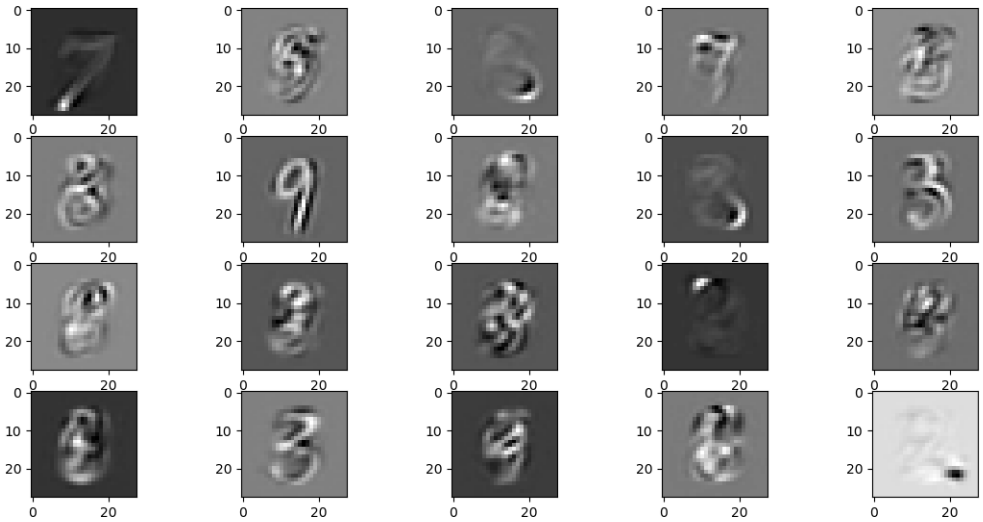


Figure 15: Examples of D's atoms

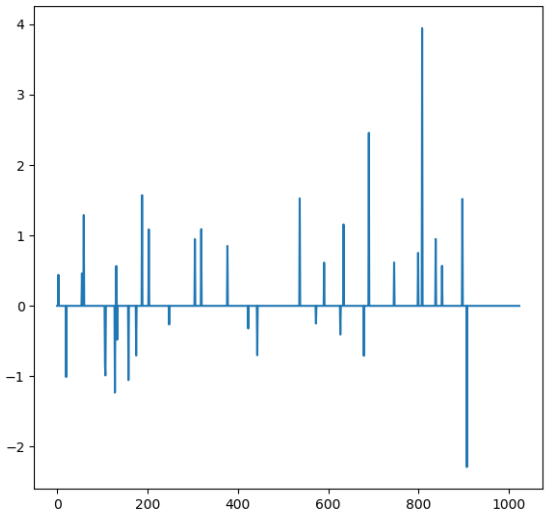


Figure 16: Sparse coefficients repartition

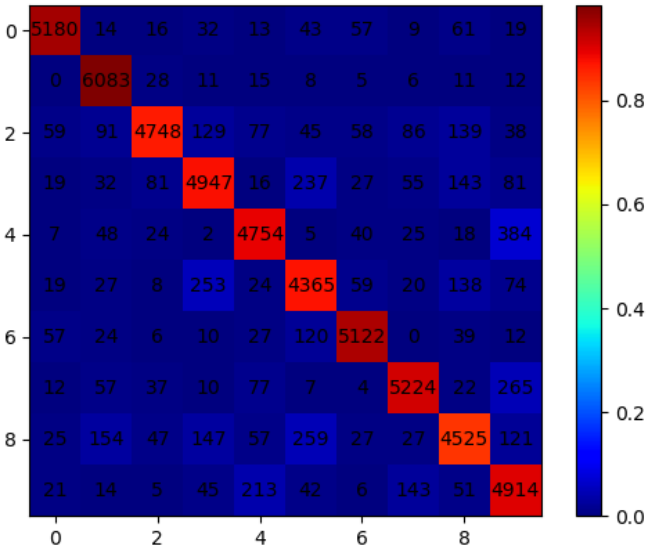
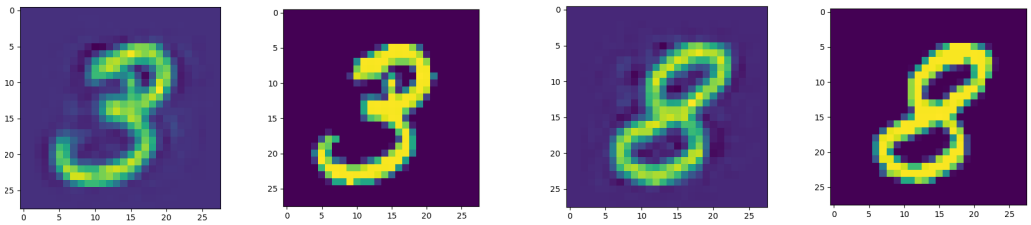


Figure 17: Kmeans classification on Aα from the trainning dataset



(a) Reconstructed 3 vs Original 3

(b) Reconstructed 8 vs Original

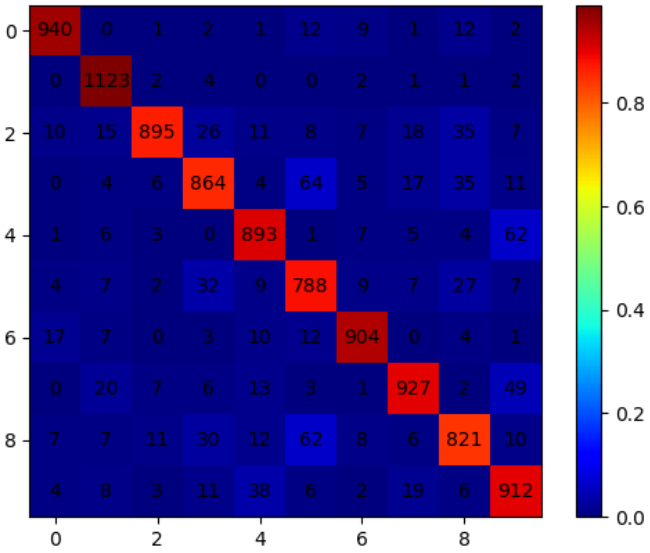


Figure 19: Kmeans classification on A α from the test dataset w

2.5 Application on "voyelle" dataset

We have for our experiments some audio signals that have been recorded in a studio. The speaker pronounces 10 vowels. There are 100 occurrences of each vowel (*aa, ee, eh, eu, ii, oe, oh, oo, uu, yy*). Each signal is of 1024 sample.
A cepstral parametrization has been extracted from these samples, and a PCA has been done to reduce the data's dimension to 2 and 12.

First, we focus our research on 2 dimension dataset. There are 10 classes which correspond to 10 vowels.

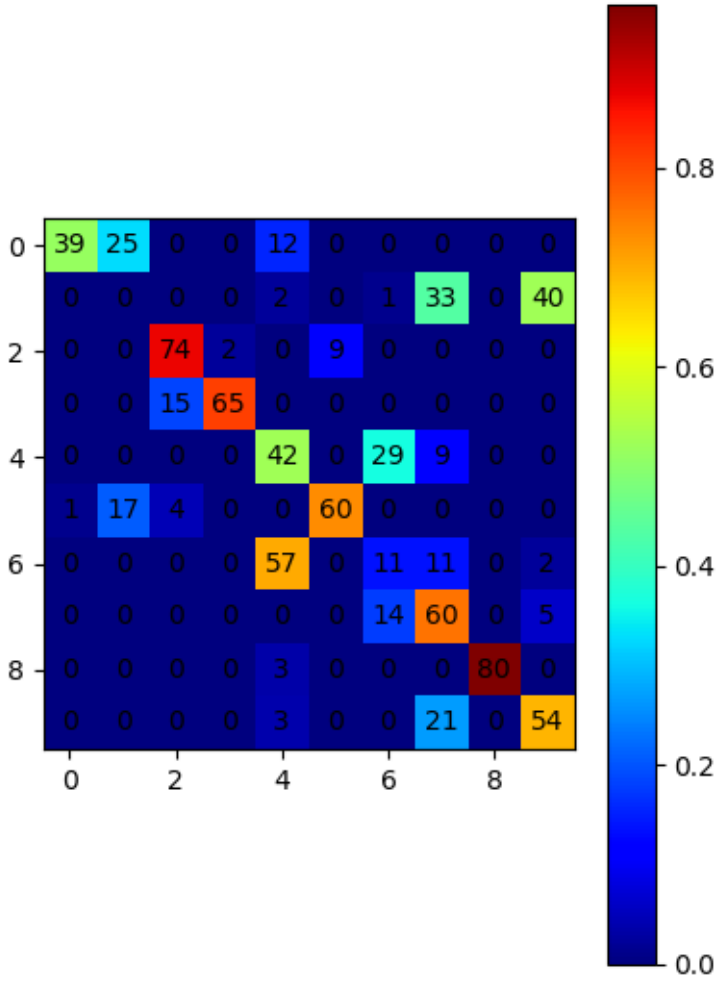


Figure 20: Kmean classification on Aα from the train dataset

Note: Here there is bad cluster identification (this is why we don't have a perfect diagonal matrix), however, the result is the same: In this case, our method is bad for classification because on the same row or column there is sometimes a bad repartition. It's due to the length of our input vector (2 dimensional vector), which is not enough to get the good result for classification using this method of Sparse representation of the signal.

Then we focus our approach on 12 dimension dataset:

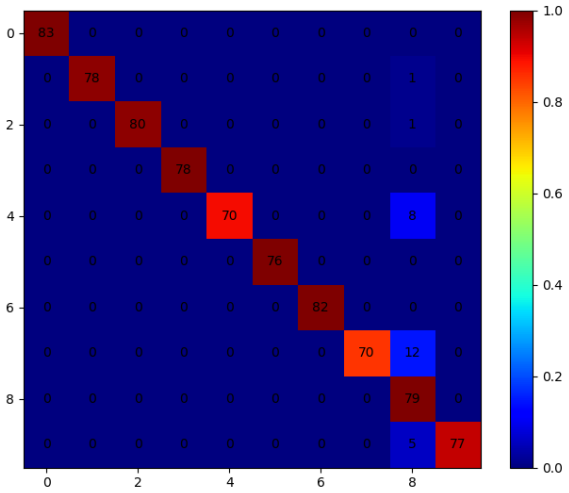


Figure 21: Kmean on data12 train set

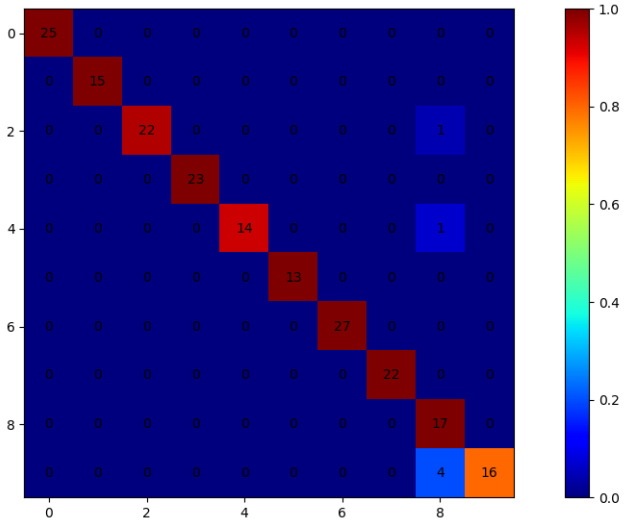


Figure 22: Kmean on data12 test set

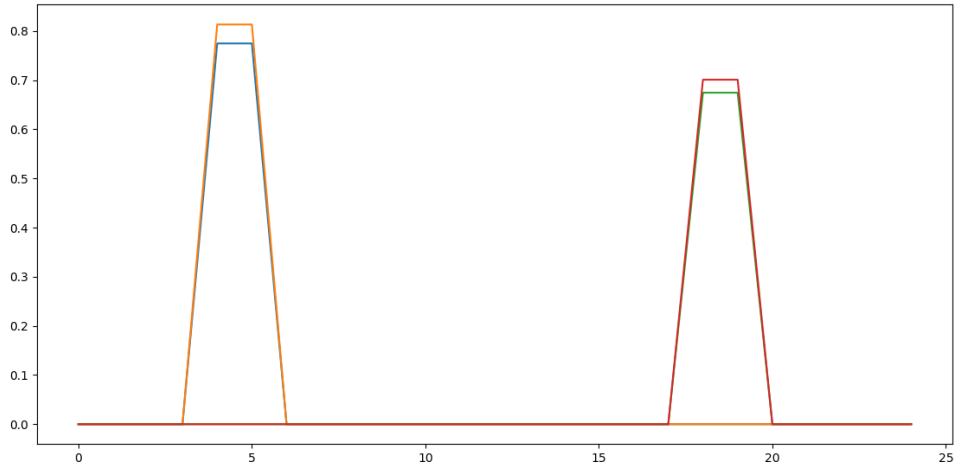


Figure 23: Four examples of Aα from two different classes (orange and blue : Class 1, red and green: Class 2)

2.6 Discussion

LC-KSVD perform well on our two datasets (MNIST and voyelle) when the input data are large enough (when input equal 2 we can see we get a bad result of clusterization). With this method we have two dictionaries:

1. **D** : A reconstructive dictionary
2. **A**: A discriminative dictionary

Using **A** we get a new representation of our input, a discriminative one, that we can put in the input of classifier to get a good classification.
However, this method works well on non-shift variant signals. When we have shift variant signal this method will fail to get a good discriminative representation i.e. If we have the same signal but delay with δ, LC-KSVD will fail to get a close representation of the two signals.

3 Convolutional Sparse Coding

Notations:

- Matrix: Uppercase $\mathbf{A}$
- Vector: Lower-case $\mathbf{a}$
- Scalar: Lower-case a
- 2D convolution operation : *
- $D \times D$ identity matrix: I_D
- Kronecker product : $\odot$
- $D \times D$ Fourier matrix : $\mathbf{F}$
- Hadamard product: $\otimes$
- inverse FFT : $\mathcal{F}^{-1}$
- Mapping function that preserve only $M \ll D$ active values relating to the small spaital structure of the estimated filter : $\mathcal{M}\{\}$

3.1 Idea

There are other algorithms like Efficient Shift-Invariant Dictionary learning which refers to the problem of discovering a set of latent basis vectors that capture informative *local patterns* at different locations of the input sequences and not in all input sequences [**?**], to do that, we will use Convolutional Sparse Coding method.

3.2 Problem formulation

Instead of decomposing a signal as the $x = Dh$, Convolutional Sparse Coding (CSC) is the summation of convolutions between the feature map and the corresponding filter.

$$X = \sum_{i=1}^{m} d_i * Z_i$$

$$\arg\min_{d,z} \tfrac{1}{2}\|x - \sum_{k=1}^{K} d_k * z_k\|_2^2 + \beta\|z_k\|_1$$

This new approach assume the ensemble of input vectors X are independent of one another. But the complexity of convergence is dramatically increase. Bristow propose an Augmented Lagrangian and the used of Alternative Direction Method of Multipliers (ADMM) [**?**] to solve this problem. His approach is based on three key points:

- The use of ADMM
- The convolution subproblem can be solved efficiently (and explicitly) in the Fourier domain (instead of temporal domain)
- The use of quad-decomposition of the objective into four subproblems (that are convex)

In this approach, to solve a Convolutional Sparse Coding problem we introduce two auxilliary variables: $\mathbf{t}$ and $\mathbf{s}$ and posing the objective in the Fourier domain:

$$\begin{aligned} \arg\min_{d,s,z,t} \quad & \tfrac{1}{2D}\|\hat{x} - \sum_{k=1}^{K} \hat{d}_k \odot \hat{z}_k\|_2^2 + \beta \sum_{k=1}^{K} \|t_k\|_1 \\ \text{subject to} \quad & \|s_k\|_2^2 \leq 1 \text{ for k} = 1..\ \text{K} \\ & s_k = \Phi^T \hat{d}_k \text{ for k} = 1..\text{K} \\ & z_k = t_k \text{ for k} = 1..\text{K} \end{aligned}$$

With Φ a D $\times$ M submatrix of the Fourier matrix $F = [\Phi, \Phi_\perp]$. In this formulation $\hat{d}_k$ is a D dimensional vector whereas in the original formulation $d_k \in \mathbb{R}^M$ is of a significantly smaller dimensionality to $M \ll D$ corresponding to its smaller spatial support.

3.3 Solve the minimization problem

3.3.1 Augmented Lagrangian

We handle the introduction of new equality constraints through an augmented Lagrangian approach [**?**]:

$$\begin{aligned}\mathcal{L}(d,s,z,t,\lambda_s,\lambda_t) = & \\ & \tfrac{1}{2D}\|\hat{x} - \textstyle\sum_{k=1}^{K} \hat{d}_k \odot \hat{z}_k\|_2^2 + \beta\|t\|_1 \\ & +\lambda_s^T(s - [\Phi^T \otimes I_K]\hat{d}) + \lambda_t^T(z-t) \\ & +\tfrac{\mu_s}{2}\|s - [\Phi^T \otimes I_K]\hat{d}\|_2^2 \\ & +\tfrac{\mu_s}{2}\|z-t\|_2^2\end{aligned}$$

3.3.2 Quad-decomposion of the objective

We decompose our objective into four convex subproblems:

Subproblem z :

$$\begin{aligned}z^* &= \arg\min_z \mathcal{L}(z,d,s,t,\lambda_s,\lambda_t) \\ &= \mathcal{F}^{-1}\{\arg\min_z \tfrac{1}{2}\|\hat{x} - \hat{D}\hat{z}\| + \hat{\lambda}_t^T(\hat{z}-\hat{t}) + \tfrac{\mu_t}{2}\|\hat{z}-\hat{t}\|_2^2\} \\ &= \mathcal{F}^{-1}\{(\hat{D}^T\hat{D} + \mu_t I)^{-1}(\hat{D}^T\hat{x} + \mu\hat{t} - \hat{\lambda}_t)\}\end{aligned}$$

Where $\hat{D} = [diag(\hat{d}_1), ..., diag(\hat{d}_K)]$

Subproblem t :

$$\begin{aligned}t^* &= \arg\min_t \mathcal{L}(t,d,s,z,\lambda_s,\lambda_t) \\ &= \arg\min_t \tfrac{\mu_t}{2}\|z-t\|_2^2 + \lambda_t^T(z-t) + \beta\|t\|_1\end{aligned}$$

Unlike subproblem z, the solution to t cannot be efficiently computed in the Fourier domain (since L_1 norm is not rotaiton invariant). Solving t requires projecting $\hat{z}$ and $\hat{\lambda}_t$ back into the spatial domain. If this equation does not contain any rotations of the data, each element of t can be solved independently:

$$t^* = \arg\min_t \beta|t| + \lambda_t(z-t) + \tfrac{\mu}{2}(z-t)^2$$

Where the optimal value of each t can be found using shrinkage function:

$$t^* = sign(z + \tfrac{\lambda_t}{\mu_t}).max\{|z + \tfrac{\lambda_t}{\mu_t}| - t, 0\}$$

Subproblem d :

$$\begin{aligned}d^* &= \arg\min_s \mathcal{L}(d,s,z,t,\lambda_s,\lambda_t) \\ &= \mathcal{F}^{-1}\{\arg\min_{\hat{d}} \tfrac{1}{2}\|\hat{x} - \hat{Z}\hat{d}\|_2^2 + \hat{\lambda}_s^T(\hat{d}-\hat{s}) + \tfrac{\mu}{2}\|\hat{d}-\hat{s}\|_2^2\} \\ &= \mathcal{F}^{-1}\{(\hat{Z}^T\hat{Z} + \mu_s I)^{-1}(\hat{Z}^T\hat{x} + \mu_s\hat{s} - \hat{\lambda}_s)\}\end{aligned}$$

Subproblem s :

$$\begin{aligned}s^* &= \arg\min_s \mathcal{L}(d,s,z,t,\lambda_s,\lambda_t) \\ &= \arg\min_s \tfrac{\mu_s}{2}\|\hat{d} - [\Phi^T \otimes I_K]s\|_2^2 + \hat{\lambda}_s^T(\hat{d} - [\Phi^T \otimes I_K]s)\end{aligned}$$

Solving this equation as it is a quadratically constrained quadratic programming problem (QCQP). Due to the Kronecker product with the identity matrix I_K it can be broken down into K independent problems:

$$s_k^* = \arg\min_{s_k} \tfrac{\mu_s}{2}\|\hat{d}_k - \Phi^T s_k\|_2^2 + \hat{\lambda}_{sk}^T(\hat{d}_k - \Phi^T s_k)$$

Futher, since Φ is orthonormal projectiong the optimal solution to the unconstrained problem cab be found efficiently through:

$$s^* = \begin{cases} \|\tilde{s}\|_2^{-1}\tilde{s}_k \text{ , if } \|\tilde{s}\|_2^{-1} \geq 1 \\ \tilde{s}_k \text{ otherwise} \end{cases}$$

where,

$$\tilde{s}_k = (\mu_s \Phi\Phi^T)^{-1}(\Phi\hat{d}_k + \Phi\hat{\lambda}_{sk})$$

Finally the solution to this equation can be found using :
$\tilde{s}_k = \mathcal{M}\{\frac{1}{\mu_s}\sqrt{D}^{-1}(\mathcal{F}^{-1}\{\hat{d}_k\} + \mathcal{F}^{-1}\{\hat{\lambda}_{sk}\})\}$

3.3.3 Lagrange Multiplier Update

$$\lambda_t^{(i+1)} \leftarrow \lambda_t^{(i)} + \mu_t(z^{(i+1)-t^{(i+1)}})$$
$$\lambda_s^{(i+1)} \leftarrow \lambda_s^{(i+1)} + \mu_s(d^{(i+1)-s^{(i+1)}})$$

3.3.4 Penalty update

Convergence may be reach if $\mu^{(i)} \to \infty$:

$$\mu^{(i+1)} = \begin{cases} \tau\mu^{(i)} \text{ if } \mu^{(i)} < \mu_{max} \\ \mu^{(i)} \text{ otherwise} \end{cases}$$

3.4 SPORCO

SPORCO (SParse Optimization Research COde) is a Python package for solving optimization problem with sparsity problem. These conssit primarily of sparse coding and dictionary learning problems but there is also support for other problems suchs as Convolutional sparse coding [**?**, **?**]

You can find my code of Convolutional Sparse Coding/Dictionary learning on MNIST using SPORCO toolbox on my github `CSC.py`.

3.5 Application on random images

In this section we show result for applying unsupervised Convolutional Sparse Coding (denoted as CSC) presented before on 7 random images, 6 for our train and 1 for our test.

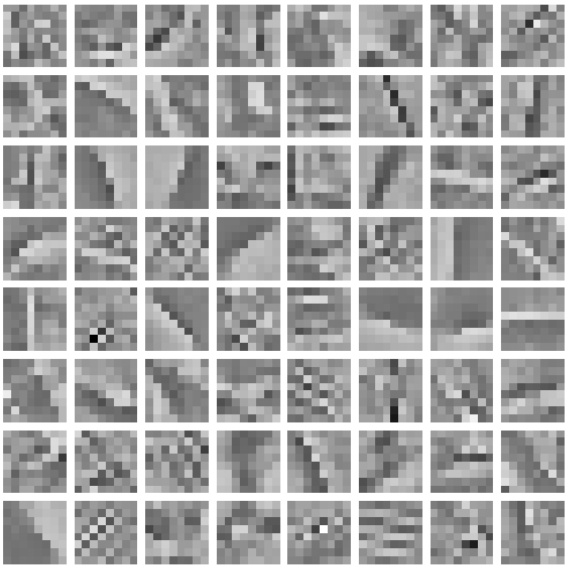


Figure 24: Dictionary learned with 6 random images

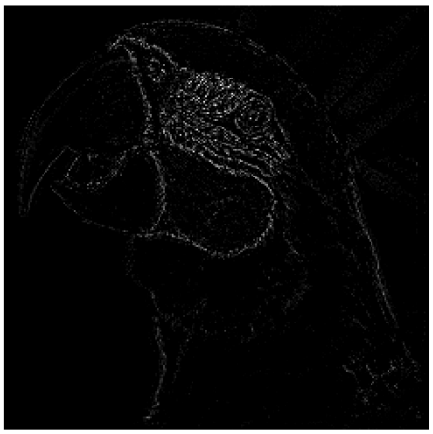


Figure 25: Activation

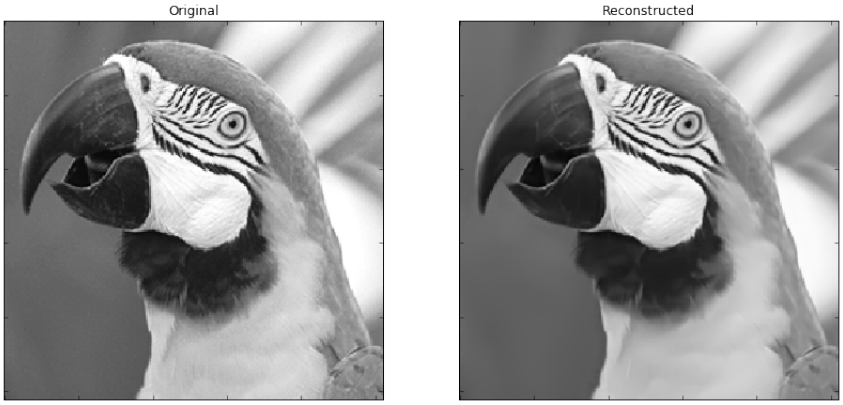


Figure 26: Original image vs Reconstructed one

3.6 Application on MNIST

In this section we show result for applying unsupervised Convolutional Sparse Coding (denoted as CSC) presented before on MNIST dataset.

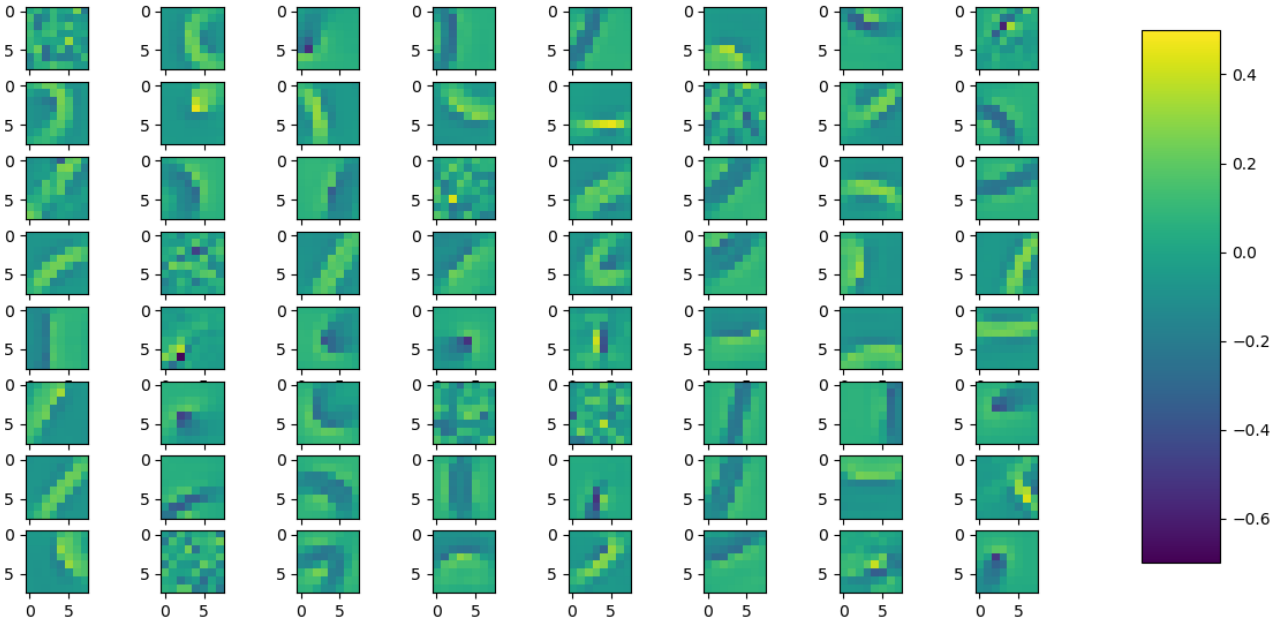


Figure 27: Filters from D

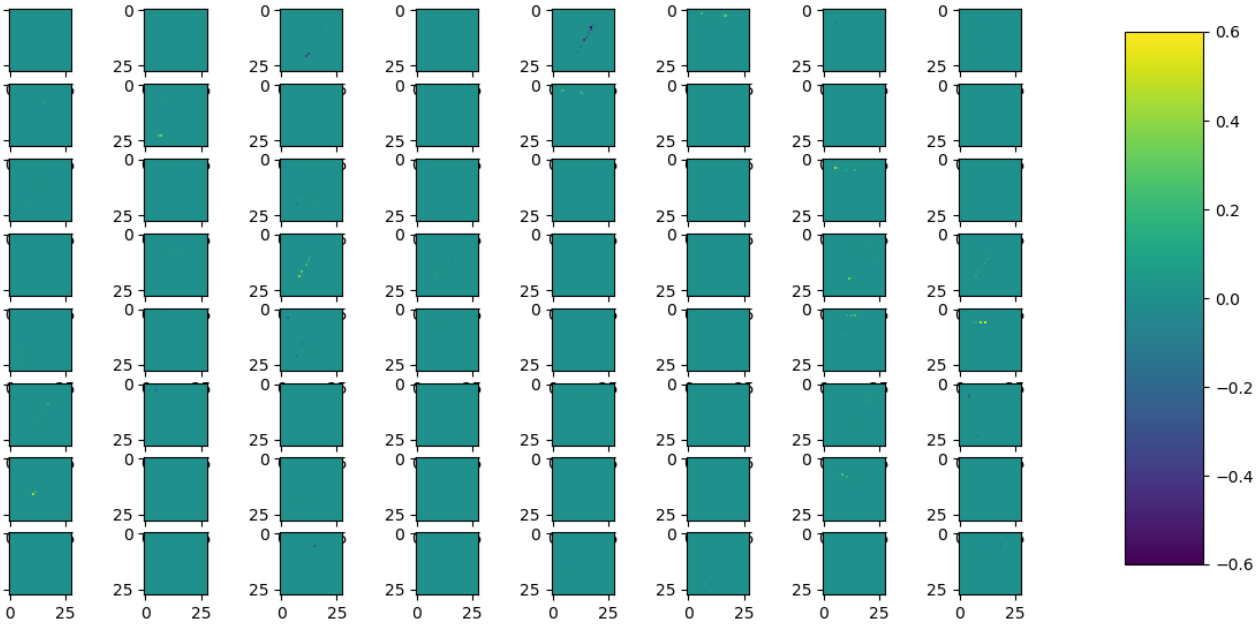


Figure 28: Activation map

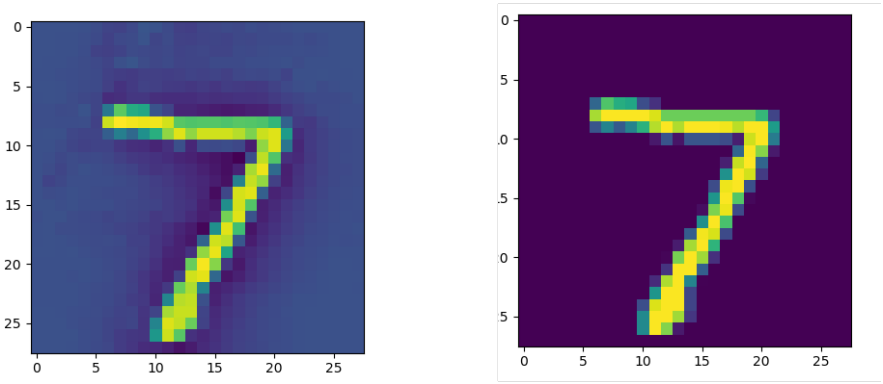


Figure 29: Reconstructed 7 vs Original 7

3.7 Discussion

Using this examples we validate that CSC allow us to reconstruct our input data using filters, convolutions and activation maps. But we still cannot use activation maps for clustering or classification due to the non-discriminative reconstruction. Indeed, two seven in our test dataset haven't the same decomposition (i.e. they don't have the same or close activation maps).

4 Supervised Convolutional Sparse Coding

TODO

5 Sparse Coding for speech recognition

In this part of the paper we will see novel feature exraction technique based on the principles of sparse coding [**?**]. Sparse codigin deals with the problem of how represent a given audio input as a linear combination of a minimum number of basis function. The weights of the linear combination are used as feature for speech recognition (acoustic modeling). Note the input dimensionality is typically **much** less than the number of atoms in the dictionary *i.e.* we use overcomplete dictionary.
We use Sparse Coding algorithm as describe before and we get the dictionary D and the matrix of sparse coefficients h.

Reflection path In [**?**] they used spectro-temporal speech domain wich is obtained by performing a short time Fourier transform (STFT) with an analysis window of length 25 ms and a frameshit of 10 ms on the input signal. Log critical band energies are subsequently obtained by projecting the magnitude square values of the STFT output on a set of frequency weights, which are equally spaced on the Bark frequence scale, and then applying a logarithm on the output projections.