Table of contents

1. Introduction

What is sparsity ?

Sparse coding is a representation learning method which aims at finding a sparse representation of the input data.

The most common ways to represent a signal with sparsity

- Fourier transform
- Wavelet transform
- Dictionary learning

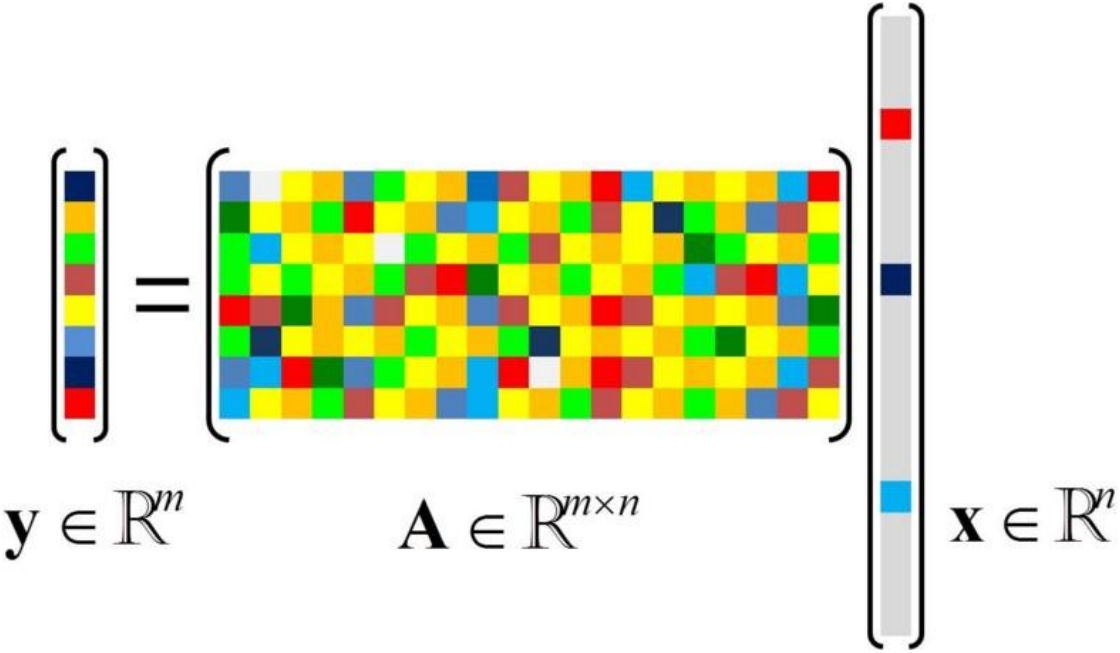
What is sparsity ?

Sparse coding is a representation learning method which aims at finding a sparse representation of the input data.

The most common ways to represent a signal with sparsity

- ~~Fourier transform~~
- ~~Wavelet transform~~
- Dictionary learning

Dictionary learning



What is the dictionary D ?

- D (dictionary) is a set of basis vectors or atoms that are used to represent the input signal

- Each column of the dictionary matrix D represents an atom or a building block

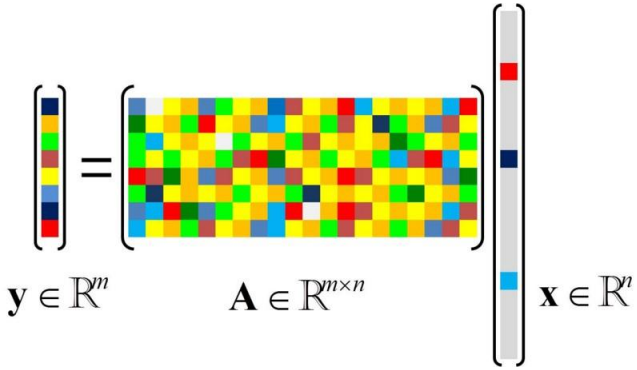
- We can use predefined dictionary D

What is the sparse vector ?

A sparse vector is a vector that has a relatively small number of non-zero elements

They can be used to represent data that is naturally sparse, such as text or images.

Dictionary learning



It is a method of sparse coding, which aims to represent data as a linear combination of a few basis elements from a learned dictionary.

What is the dictionary D ?

- The columns of the dictionary matrix must to be uncorrelated to recover a unique sparse representation

- In practice, enforcing uncorrelated columns in the dictionary matrix is often achieved through a process called decorrelation or whitening

The goal of dictionary learning

Find an optimal dictionary that can best represent the given data in a sparse and efficient manner.

2. Supervised Dictionary Learning

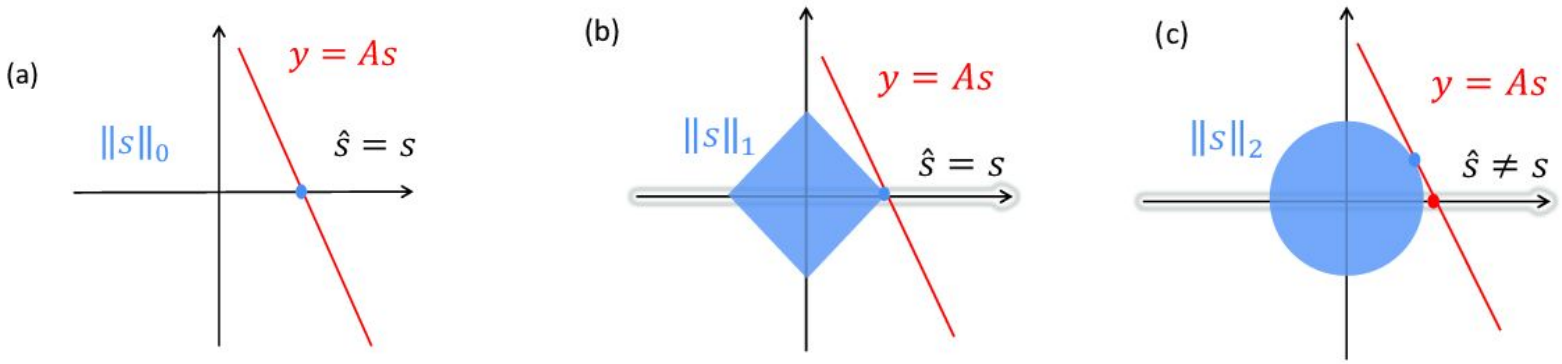
For a learned dictionary D the objective function becomes

Minimize $\frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\|\text{xi} - D\alpha_i\|_2^2 + \lambda\|\alpha_i\|_0$

$D \in C$, $\alpha \in R^{k \times n}$

This problem does not exhibit convexity !!!

How to measure sparsity ?



For a learned dictionary D the objective function becomes

Minimize $\frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\|xi - D\alpha_i\|_2^2 + \lambda\|\alpha_i\|_1$

$D \in C$, $\alpha \in R^{k \times n}$

This is a joint optimization problem with respect to the dictionary D and the coefficients α = [α1, . . . , αn] of the sparse decomposition,

which is not jointly convex, but convex with respect to each of the two variables D and α when the other one is fixed

ONLINE VS BATCH

3. The Online Algorithm

Intuition: What we need to compute?

- A dictionary ***D*** that encodes the patterns and the structure from the selected samples from the input data.
- The sparse coding coefficients α are calculated by solving the following minimization problem:

$$\boldsymbol{\alpha}_t \triangleq \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^k} \frac{1}{2}||\mathbf{x}_t - \mathbf{D}_{t-1}\boldsymbol{\alpha}||_2^2 + \lambda||\boldsymbol{\alpha}||_1 .$$

Algorithm 1: Computing *D* (Learned Dictionary)

1. Initialize the variables used.
   a. Reset the past information (Matrix A, and Matrix B).
   b. Initialize the dictionary to D_0.
   c. Prepare a probability distribution *p(x)*, and an algorithms to draw i.i.d samples from it.
2. For each time instance <u>till convergence</u> or till we <u>reach time instance *T*</u>
   a. Draw X_t from *p(x).*
   b. Compute the current α_t by solving the following optimization problem.

$$\boldsymbol{\alpha}_t \triangleq \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^k} \frac{1}{2}||\mathbf{x}_t - \mathbf{D}_{t-1}\boldsymbol{\alpha}||_2^2 + \lambda||\boldsymbol{\alpha}||_1.$$

   c. Update the values of A, B using the new computed value of α.

$$\mathbf{A}_t \leftarrow \mathbf{A}_{t-1} + \alpha_t \alpha_t^T.$$
$$\mathbf{B}_t \leftarrow \mathbf{B}_{t-1} + \mathbf{x}_t \alpha_t^T.$$

   d. Compute D_t = ***update_dictionary*** (D_{t-1} , *A, B*)
3. Return the learned dictionary.

Notes: Terminology

- Dictionary columns are known as *atoms*.
- Sparse coding coefficient columns α contains weights from the dictionary atoms to construct the original data samples X.
    - Contribution of each atom in the representation of the original signal.

Algorithm 2: Update *D* (Update Dictionary on Iteration *t)*

1. The arguments are: *Dictionary from previous iteration*, *A*, *B*.
2. For each dictionary column d_j

$$\mathbf{u}_j \leftarrow \frac{1}{\mathbf{A}_{jj}}(\mathbf{b}_j - \mathbf{D}\mathbf{a}_j) + \mathbf{d}_j.$$

$$\mathbf{d}_j \leftarrow \frac{1}{\max(||\mathbf{u}_j||_2, 1)}\mathbf{u}_j.$$

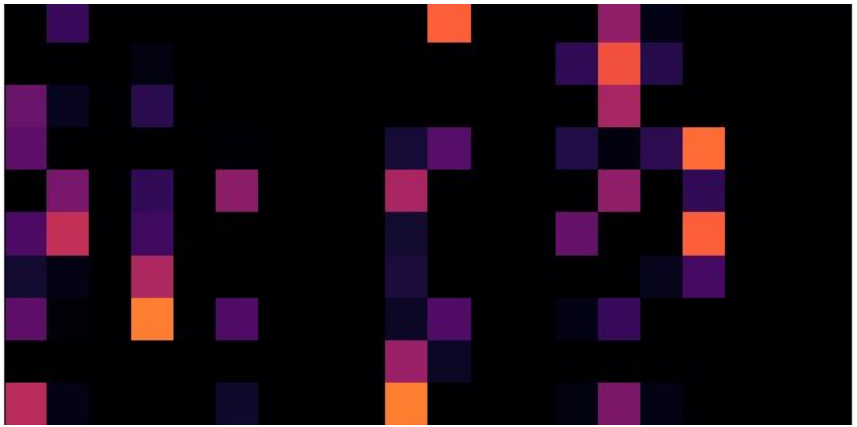
3. Return the updated Dictionary

This is equivalent to solving the following optimization problem *(with* α *as a constant)*

$$\arg\min_{\mathbf{D}\in\mathcal{C}} \frac{1}{t}\sum_{i=1}^{t}\frac{1}{2}||\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i||_2^2 + \lambda||\boldsymbol{\alpha}_i||_1,$$

Algorithm 2: Update *D* (Update Dictionary on Iteration *t)*

- **Exclude** the effect of other atoms to the construction of the current column of X, *only leave the effect of the atom we want to update.*
- Try to maximize update the current atom such that it maximizes the information of the signal X, *alone*.
- This will reduce the number of atoms needed, and hence more sparsity.



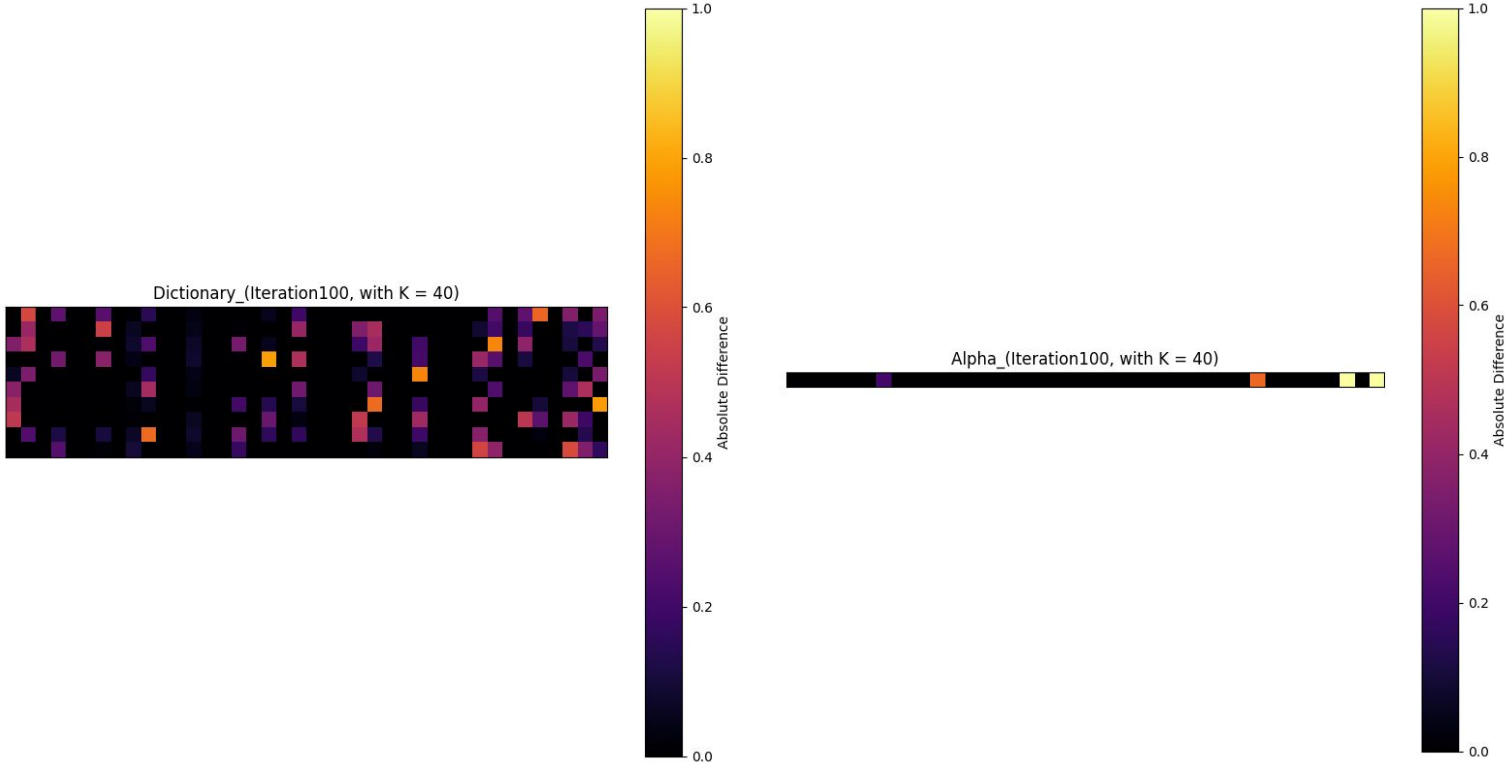
Updating D: Block-coordinate descent

- The algorithm for updating D is called block-coordinate descent with warm restarts.
- Block-coordinate descent**:** an optimization algorithm that works by optimizing one block of variables (or parameters) at a time while holding the other blocks fixed.
- It is **parameter-free** and does not require any learning rate tuning, which can be difficult in a constrained optimization setting.
- It sequentially updates each column of D (Using some *simple algebra)* while keeping the other constants.

Other Ways To Update The Dictionary

- Newton method but this requires *inverting* a $k \times k$ matrix at each Newton iteration, which is impractical for an online algorithm.
- SGD method, but the problem is that we will need to tune the step size *(learning rate)* to control the magnitude of the updates made to the dictionary at each iteration.

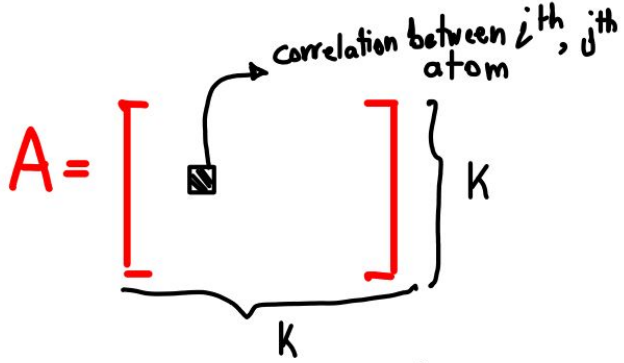
Notes: Sparsity

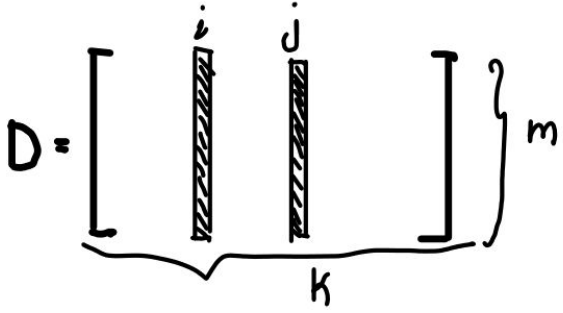


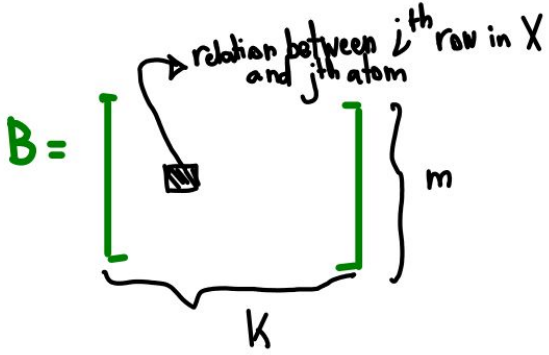
Notes: Meaning of A, B

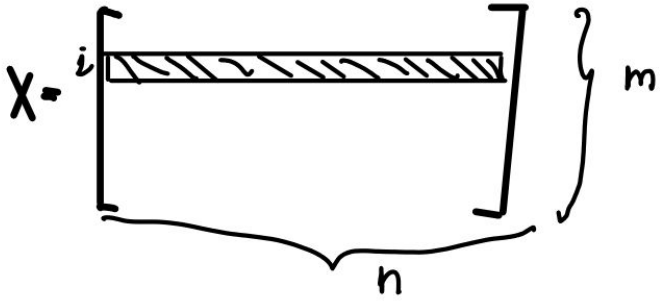
- They are auxiliary matrices. Mainly represent past information.
- The **matrix A** is used to impose a constraint on the *dictionary D* to ensure that it is *not too "similar" to previous dictionaries*.
    - Less correlation == better convergence.
- The **matrix B** is used to impose a constraint on D to ensure that it can *captures the details of the input data with good quality*.

Notes: Meaning of A, B









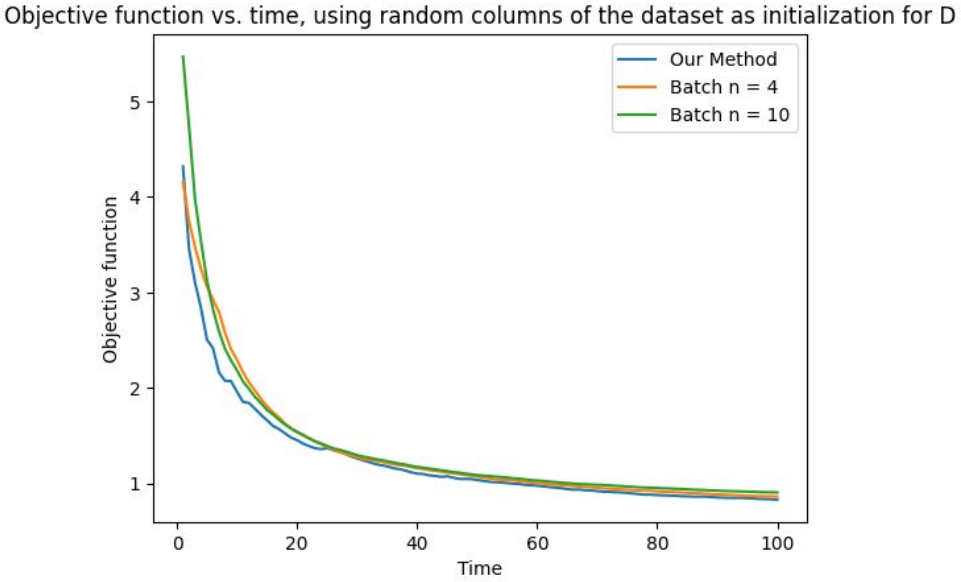
Notes: Dictionary Initial Values

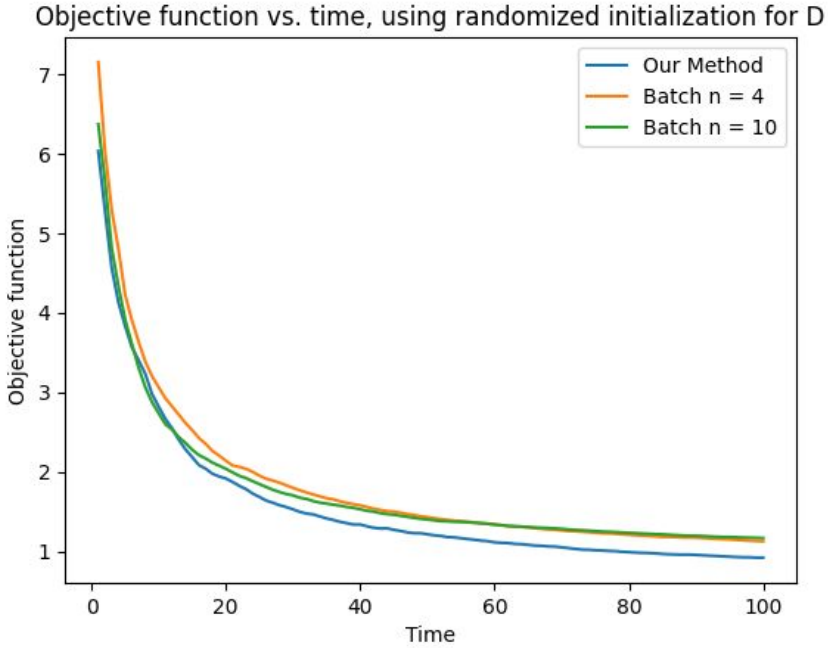
- There is no clear rule on how to initialize:
  - Initialize the dictionary by taking random columns from the dataset, then normalize it.
  - Use random generated values.
- Dictionary columns must be normalized, to prevent D columns from having large values, which in turn can lead to very small values of α

$$d_j = \frac{d_j}{\max(||d_j||_2, 1)}$$

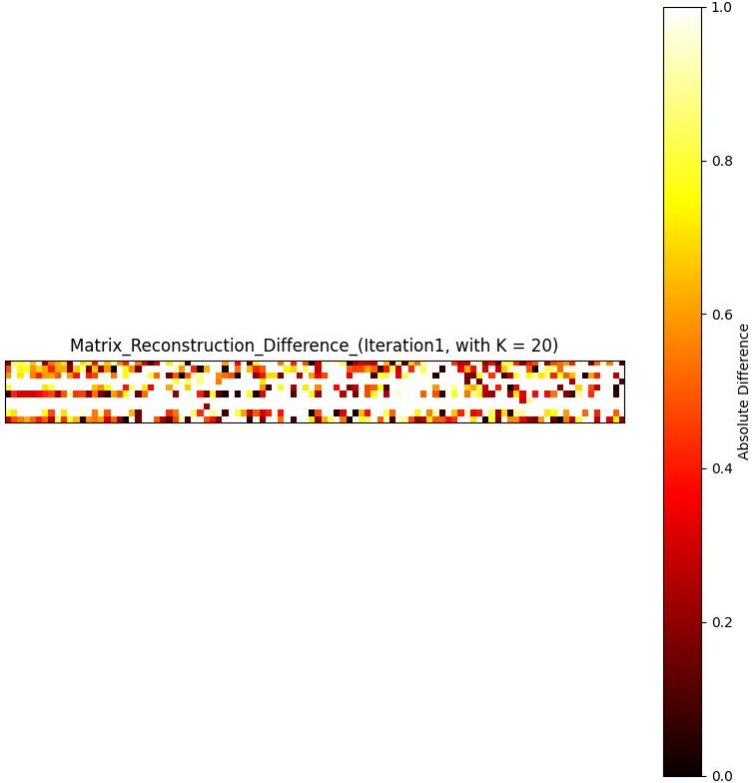
- Initialization values affects convergence.

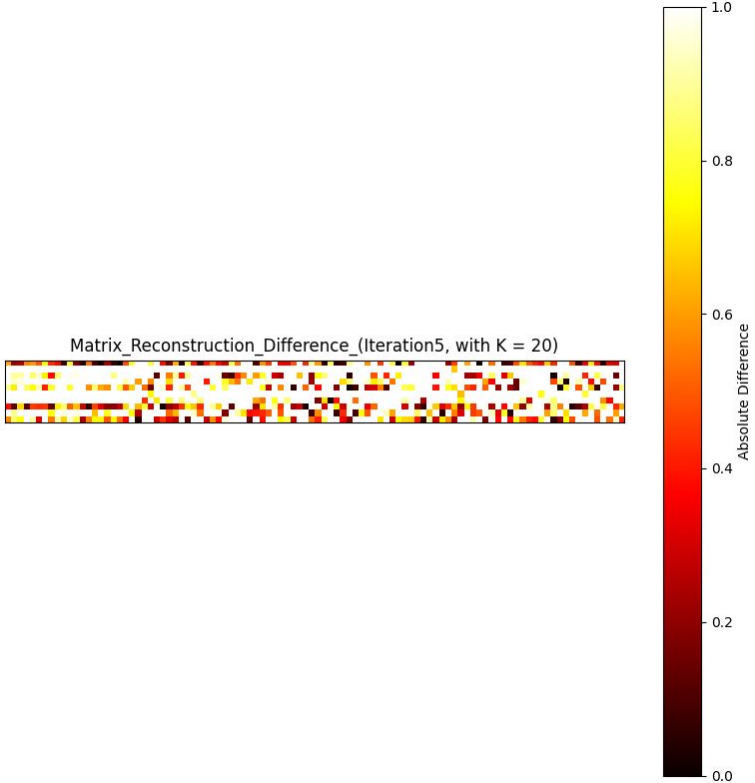
Notes: Dictionary Initial Values



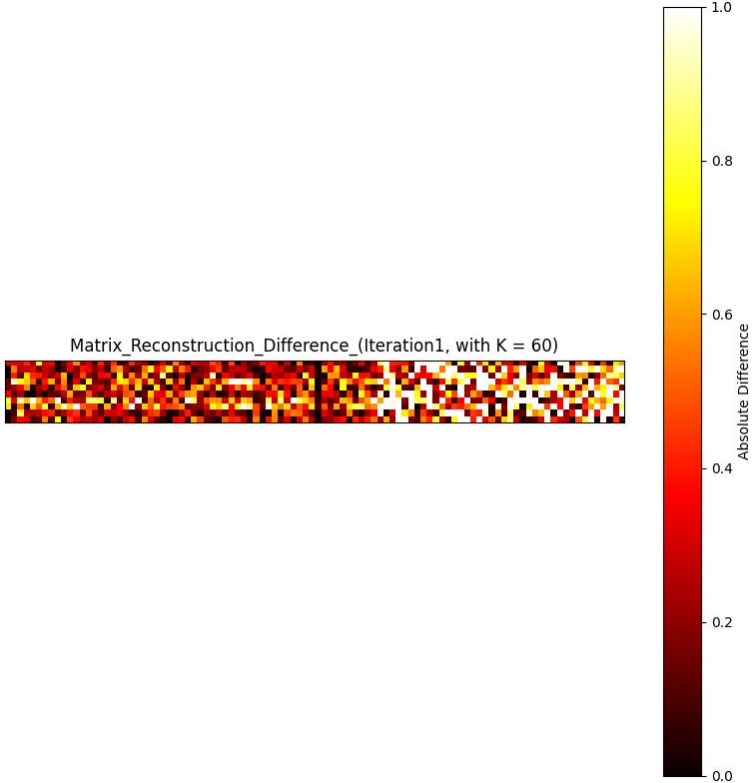


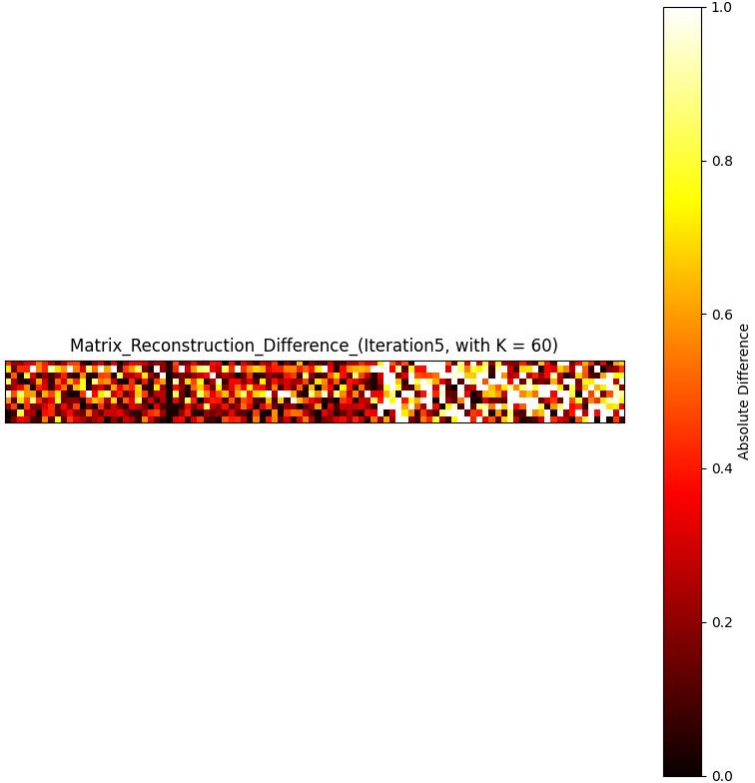
Example *(k = 20, m = 10, n = 100)*



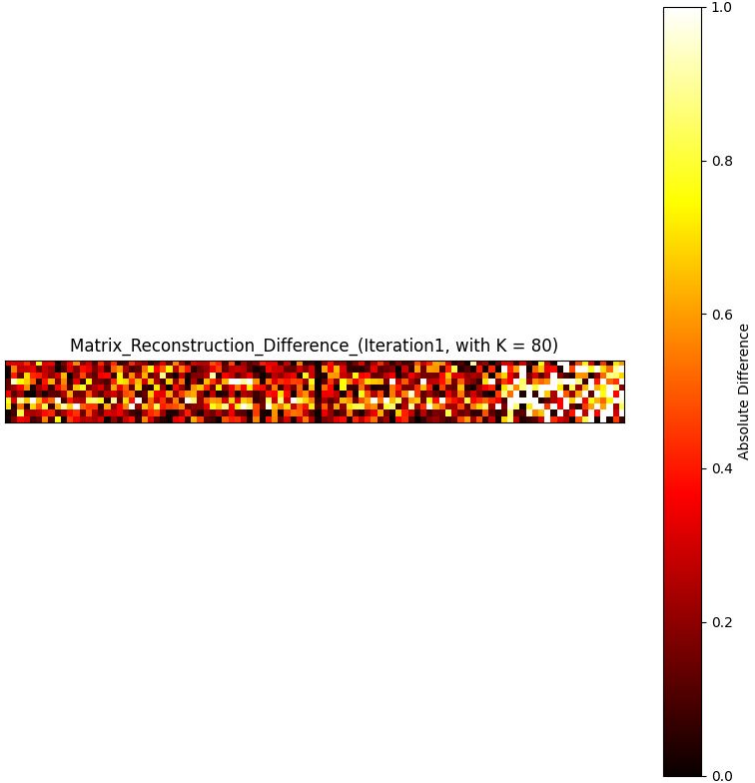


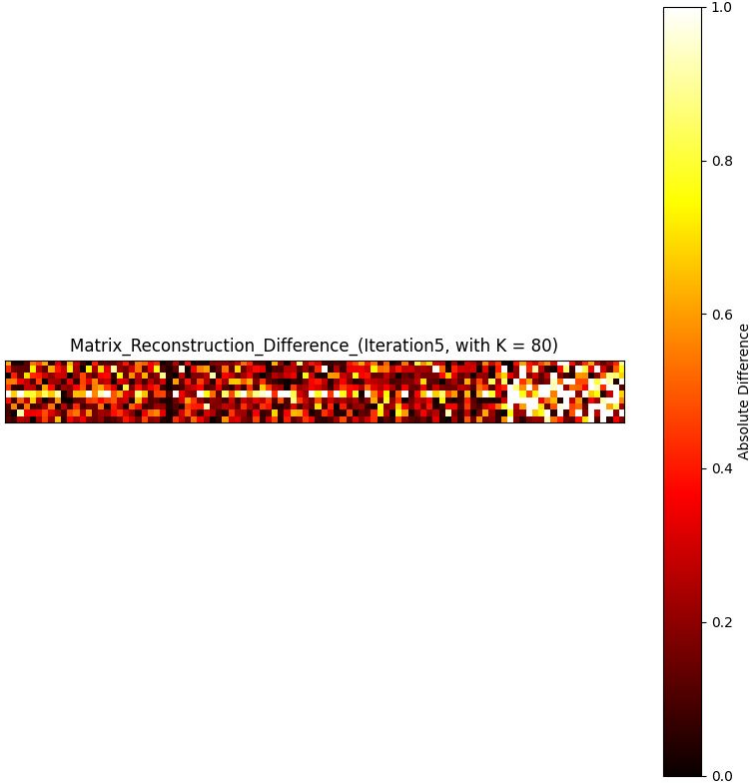
Example *(k = 60, m = 10, n = 100)*



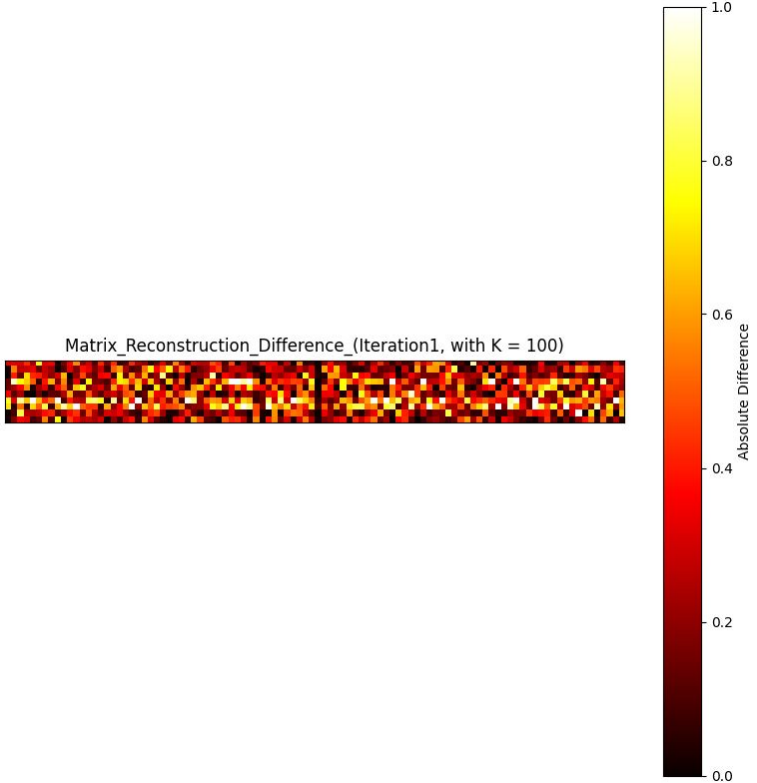


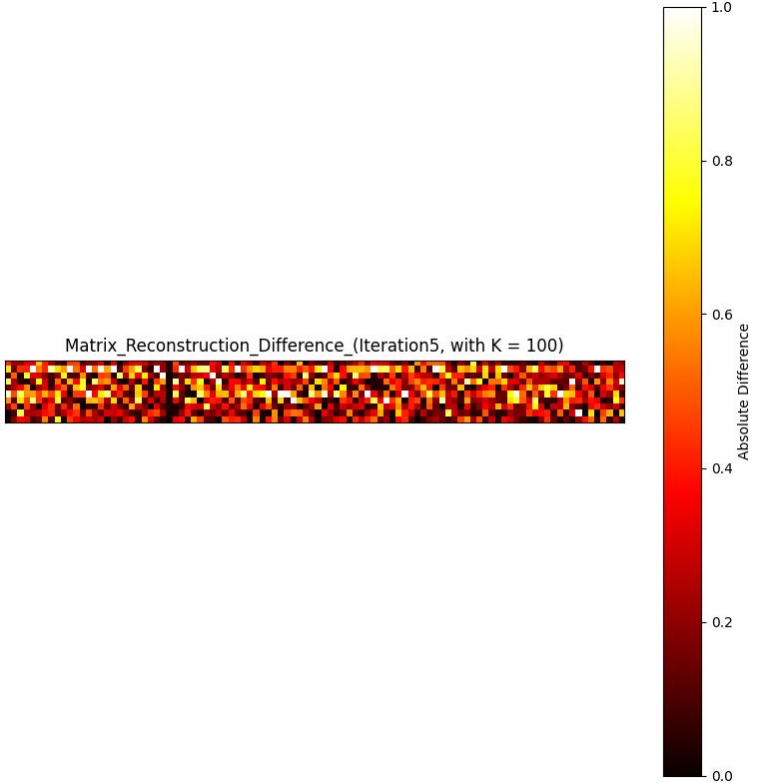
Example *(k = 80, m = 10, n = 100)*





Example *(k = 100, m = 10, n = 100)*





Notes: Dimensions

- **X** (Dataset): *m x n*
- **D** (Dictionary): *m x k*
- **A**: *k x k*
- **B**: *m x k*
- α (sparse coding coefficients): *k x n*

Four Optimizations

1. Handling Fixed-Size Datasets.
2. Mini-Batch Extension.
3. Scaling the “past” Data.
4. Purging the Dictionary from Unused Atoms.

1. Handling Fixed-Size Datasets.

- The size of the training set is often finite (may not be the case, when the data consists of a *video stream*).
- When it's finite, the same data points may be examined several times, and it is very common in online algorithms to simulate an i.i.d. sampling of p(x) by cycling over a randomly permuted training set. When the training set is small enough, it is possible to further speed up convergence.
- Suppose that at time t_0, a signal *x* is drawn. If the same signal *x* is drawn again at time t > t_0, one would like to remove the "old" information from A_t and B_t related to the first pick.

$$\mathbf{A}_t \leftarrow \mathbf{A}_{t-1} + \alpha_t \alpha_t^T - \alpha_{t_0} \alpha_{t_0}^T$$
$$B_t \leftarrow B_{t-1} + X_t \alpha_t^T - X_t \alpha_{t_0}^T$$

- α_{t_0} : represents past coefficients from this same selected column of *X*.

2. Mini-Batch Extension.

- We can improve the **convergence speed** of our algorithm by drawing $\eta \geq 1$ samples *(columns)* at each iteration from the dataset, instead of a single one.
- This is a classical heuristic in stochastic gradient descent algorithms.
- We will reflect those changes on A, B as follows:

$$\mathbf{A}_t \leftarrow \mathbf{A}_{t-1} + \frac{1}{\eta} \sum_{i=1}^{\eta} \alpha_{t,i} \alpha_{t,i}^T,$$

$$\mathbf{B}_t \leftarrow \mathbf{B}_{t-1} + \frac{1}{\eta} \sum_{i=1}^{\eta} \mathbf{x}_{t,i} \alpha_{t,i}^T.$$

3. Scaling the “past” Data

- At each iteration, the “new” information α_t is added to the matrices A_t and B_t. We are accumulating it to the old information.
- We want the newer coefficients α_t to have more weight than the old ones. This can be done by *rescaling the “old” information*.

$$\mathbf{A}_t \leftarrow \beta_t \mathbf{A}_{t-1} + \alpha_t \alpha_t^T$$
$$\mathbf{B}_t \leftarrow \beta_t \mathbf{B}_{t-1} + \mathbf{x}_t \alpha_t^T,$$

- Where the scaling factor: $\beta_t = (1 - \frac{1}{t})^\rho$, **ρ** is a new parameter >= *0*.
- Results in better convergence for large data sets only *(n ≥ 100 000)*.

4. Purging the Dictionary from Unused Atoms.

- Every dictionary learning technique sometimes encounters situations where some of the dictionary *atoms are never (or very rarely) used* (have small/zero weights).
- This typically happens due to very bad initializations.
- A common practice is to replace them during the optimization by elements of the training set, which solves this problem in most cases.

4. Convergence Analysis

Map

Step one	Step two	Step three	Step four
Assumptions	Lemma 1 & Preposition 0	Proposition 1	Proposition 2

Step 1
Assumptions

We have three main assumption:

(A) The data admits a distribution with compact support K

(B) The quadratic surrogate functions $\check{f}_t$ are strictly convex with lower-bounded Hessians

(C) A particular sufficient condition for the uniqueness of the sparse coding solution is satisfied

(A)The data admits a distribution with compact support K

- **This means that the probability density function of the distribution is zero outside of the set K.**

Why this assumption valid ?

- In audio, image, and video processing applications, it is often natural to assume that the data admits a distribution with compact support K, because the data acquisition process typically imposes a finite range or resolution on the measurements, which in turn limits the possible values that the data can take on.
- e.g in image and video processing, the data may be captured by a camera with a finite number of pixels or color channels, which limits the possible values that the data can take on

(B) The quadratic surrogate functions $\check{f}_t$ are strictly convex with lower-bounded Hessians.

How to achieve this assumption ?

By assuming that the smallest eigenvalue of the positive semi-definite matrix $^1/_t\, A_t$ is greater than or equal to some constant k_1

$$\arg\min_{\mathbf{D}\in\mathcal{C}} \frac{1}{t}\left(\mathrm{Tr}(\mathbf{D}^T\mathbf{D}\mathbf{A}_t) - \mathrm{Tr}(\mathbf{D}^T\mathbf{B}_t)\right)$$

Why ?

- By assuming this the author ensures that $\boldsymbol{A_t}$ is invertible and that the quadratic surrogate function $\boldsymbol{f_t}$ is strictly convex with a Hessian matrix that is lower-bounded
- The sparse coding coefficients α are typically non-negative because they are computed as the solution to a quadratic optimization problem subject
- verified experimentally after a few iterations of the algorithm when the initial dictionary is reasonable
- it is easy to enforce this assumption by adding a term $\frac{k_1}{2}\|D\|_F^2$ to the objective function, which is equivalent to replacing the positive semi-definite matrix $^1/_t\, \boldsymbol{A_t}$ by $^1/_t\, \boldsymbol{A_t} + k_1\mathrm{I}$

$$\mathbf{A}_t \leftarrow \mathbf{A}_{t-1} + \tfrac{1}{2}\boldsymbol{\alpha}_t\boldsymbol{\alpha}_t^T.$$

(C) A particular sufficient condition for the uniqueness of the sparse coding solution is satisfied

- Refers to the property that for a given set of observations, there exists a unique set of sparse coefficient
- Given by **Fuchs in 2005** the classical optimality conditions for the ℓ1 decomposition problem is :
- $d_j{}^t(x - D\,\alpha) \mid = \lambda \text{sign}(\alpha[j])\ \alpha[j] \neq 0$
- $\mid d_j{}^t(x - D\,\alpha) \mid \leq \lambda$
- Where d_j is a column in D and I have a set of j indices in Λ set
- suppose $\alpha\star$ is a solution where $\mid d_j{}^t(x - D\,\alpha\star) \mid \leq \lambda$
- D_Λ the matrix composed of the columns from D restricted to the set Λ
- It's clear that d_j^t is equal to D_Λ^T then if $\alpha\star_\Lambda$ is the vector containing the values of $\alpha\star$ corresponding to the set Λ and $\varepsilon[j]_\Lambda$ is equal to the sign of $\alpha[j]_\Lambda^\star$ for all j. so with simplification of the previous equation we get

$$\alpha_\Lambda^\star = (\mathbf{D}_\Lambda^T \mathbf{D}_\Lambda)^{-1}(\mathbf{D}_\Lambda^T \mathbf{x} - \lambda \varepsilon_\Lambda).$$

- We assume that there exists κ2 > 0 such that, for all x in K and all dictionaries D in the subset of C considered by our algorithm, the smallest eigenvalue of $D_\Lambda^T D_\Lambda$ is greater than or equal to κ2. This guarantees the invertibility of ($D_\Lambda^T D_\Lambda$) and therefore the uniqueness
- Although it's common that ($D_\Lambda^T D_\Lambda$) invertible is a common assumption in linear regression and in methods such as the LARS algorithm It is also possible to enforce this condition using an elastic net penalization (Zou and Hastie, 2005), replacing $||\alpha||_1$ by $||\alpha||_1 + \frac{K_2}{2} ||\alpha||_2^2$ and thus improving the numerical stability of homotopy algorithms, which is the choice made by Zou et al. (2006).

Step two

Lemma 1

What we want to prove ?

$$\mathbf{D}_{t+1} - \mathbf{D}_t = O\left(\frac{1}{t}\right)$$

- But It does not ensure the convergence of Dt , but guarantees the convergence of the positive sum $\sum_{t=1}^{\infty} ||Dt - Dt-1|.|_2^F$ which is a classical condition in gradient descent convergence proofs (Bertsekas, 1999)
- To proof this we need two equation the first one is by recalling the Taylor expansion

$$f(x) \approx f(a) + f'(a)(x - a) + f''(a)\frac{(x-a)^2}{2}$$

- By replacing x by D_{t+1} and a by D_t in the previous equation we get

➢ $\check{f}_t(D_{t+1}) = \check{f}_t(D_t) + \check{f}_t(D_t)'(D_{t+1} - D_t) + \check{f}_t(D_t)''\frac{(D_{t+1} - D_t)^2}{2}$

➢ $\check{f}_t(D_{t+1}) - \check{f}_t(D_t) = \check{f}_t(D_t)'(D_{t+1} - D_t) + \check{f}_t(D_t)''\frac{(D_{t+1} - D_t)^2}{2}$

- using assumption (B) for all t, the surrogate ˆft is strictly convex with a Hessian lower-bounded by k1

➢ $\check{f}_t(D_{t+1}) - \check{f}_t(D_t) \geq K_1 \|D_{t+1} - D_t\|_F^2$ ----------→ (1)

➢ $\check{f}_t(D_{t+1}) - \check{f}_t(D_t) = \check{f}_t(D_{t+1}) - \widetilde{f_{t+1}}(D_{t+1}) + \widetilde{f_{t+1}}(D_{t+1}) - \widetilde{f_{t+1}}(D_t) + \widetilde{f_{t+1}}(D_t) - \check{f}_t(D_t)$

- We know that
- $\widetilde{f_{t+1}}(D_{t+1}) \leq \widetilde{f_{t+1}}(D_t)$ because D_{t+1} minimizes $\widetilde{f_{t+1}}$ on C

➢ $\check{f}_t(D_{t+1}) - \check{f}_t(D_t) \geq \check{f}_t(D_{t+1}) - \widetilde{f_{t+1}}(D_{t+1}) + \widetilde{f_{t+1}}(D_t) - \check{f}_t(D_t)$

- $\breve{\boldsymbol{f}}_{\boldsymbol{t}}(D_t) = \frac{(\frac{1}{2}\,\mathrm{Tr}(D^T \mathrm{D}\, A_t) - \mathrm{Tr}(D^T\, B_t))}{t}$ and $\|D\|_F \leq \sqrt{k}$ which satisfy that each column of D is normalized so maximum value is root k which is the number of columns of D

By substituting

- $\breve{\boldsymbol{f}}_{\boldsymbol{t}}(D_{t+1}) - \widetilde{\boldsymbol{f}_{\boldsymbol{t+1}}}(D_{t+1}) + \widetilde{\boldsymbol{f}_{\boldsymbol{t+1}}}(D_t) - \breve{\boldsymbol{f}}_{\boldsymbol{t}}(D_t) =$
  $D_{t+1}{}^T D_{t+1} A_t - D_{t+1}{}^T B_t - D_{t+1}{}^T D_{t+1} A_{t+1} + D_{t+1}{}^T B_{t+1} + D_t{}^T D_t A_{t+1} - D_t{}^T B_{t+1} - D_t{}^T D_t A_t + D_t{}^T B_t$

- If we take Ct = $\frac{1}{t}(||\, B_{t+1} - B_t\, ||_F + \sqrt{k}\,||\, A_{t+1} - A_t\, ||_F)$

Now we get

- $\breve{\boldsymbol{f}}_{\boldsymbol{t}}(D_{t+1}) - \breve{\boldsymbol{f}}_{\boldsymbol{t}}(D_t) \geq$ Ct $||D_{t+1} - D_t||_F$----------→(2)
- **By dividing 1 over 2 we get the proof we want that**
- $||D_{t+1} - D_t||_F \leq \frac{Ct}{K_1}$, from assumption A and c ensure that the vectors αi and xi are bounded and therefore
- = O(1/t) we proof clearly our goal

Step two

Preposition 0

what we want to proof ?

1. the function ℓ is continuously differentiable and .$\nabla_D \ell(x,D) = -(x - D\,\alpha^\star(x,D))\,\alpha^\star(x,D)^T$.
2. f is continuously differentiable and $\nabla f(D) = E_x[\nabla_D \ell(x,D)]$;
3. $\nabla f(D)$ is Lipschitz on C

Helpful Formula

- For x in the support K of the probability distribution p, and D in the feasible set C, let us define
- $\alpha^\star = \operatorname{argmin}_{\alpha \in \mathbb{R}^k} \frac{1}{2} ||x - D\alpha||_2^2 + \lambda ||\alpha||_1$ ----------→ (3)

1 & 2

- **“Corollary of Theorem 4.1 from Bonnans and Shapiro (1998), due to Danskin (1967)”**

Let f : $R^p \times R^q$ -> R. that for all x ∈ R^p the function f(x) is differentiable, and that f and ∇u f(x,u) the derivative of f(x) are continuous on $R^p \times R^q$.

Let v(u) be the optimal value function v(u) = $\operatorname{argmin}_{x \in C}$ f(x,u), where C is a compact subset of R^p.

Then v(u) is directionally differentiable. Furthermore, if for u0 ∈ R^q , f(.,u0) has a unique minimizer x0 then v(u) is differentiable in u and $\nabla_u v(u)$ = ∇u f(x,u)

- We ensures that the vectors α ⋆ are bounded for x in K and D in C.Therefore, one can restrict the optimization problem above to a compact subset of $\mathbb{R}^k$. Under assumption (C), the solution of the equation is unique and α ⋆ is well-defined.
- borrowing some ideas from Bonnans and Shapiro (1998) can be applied and gives us directly the first statement.
- Now , Since K is compact, and ℓ is continuously differentiable, the second statement follows immediately

- we will show that for all x in K, **α⋆ (x,D) is Lipschitz with a constant** , which is a sufficient condition for ∇ f to be Lipschitz , the function optimized in Eq(3) continuous in α, D, x and has a unique minimum, implying that α⋆ is continuous in x band D.
- But first let's discuss the concept of open neighborhoods is related to the work of **LeCun, Bottou, Bengio,** and Haffner (1998) on gradient-based learning applied to neural networks. In their work, they use the concept of open neighborhoods to prove that gradient-based learning is a convergent algorithm for training neural networks.
- In mathematics, an open neighborhood of a point *p* in a topological space *X* is a set *U* that contains an open set *O* such that $p \in O \subseteq U$. In other words, an open neighborhood of a point is a set that contains all points that are sufficiently close to the point
- The condition to have a neighborhood is that the function $d_j^{\,t}(x - D\alpha*)$ must be continuous in *D* and *x* ,and this condition is satisfied so there exists an open neighborhood V around (x,D) such that for all (x ′,D ′) in V,
- $d_j^{\,t}$(x ′−D ′α ⋆′) | < λ and α ⋆′ [j] = 0, where **α ⋆′ = α ⋆ (x ′,D ′)**
- Suppose we have

$$\tilde{\ell}(\mathbf{x}, \mathbf{D}_\Lambda, \alpha_\Lambda) \triangleq \frac{1}{2}||\mathbf{x} - \mathbf{D}_\Lambda \alpha_\Lambda||_2^2 + \lambda||\alpha_\Lambda||_1,$$

- Assumption (C) tells us that $\tilde{l}$ (x, D_Λ,.) is strictly convex with a Hessian lower-bounded by κ2. Let us consider (x ′ ,D ′) in V. as previously we shows that

$\tilde{l}(x, D_\Lambda, \alpha_\Lambda^{\star'})$ - $\tilde{l}(x, D_\Lambda, \alpha_\Lambda^{\star})$ ≥ κ2 || $\alpha_\Lambda^{\star'}$.− $\alpha_\Lambda^{\star}$ ||$_2^2$.

Now for $\tilde{l}(x, D_\Lambda, \alpha_\Lambda^{\star'})$ - $\tilde{l}(x, D_\Lambda, \alpha_\Lambda^{\star})$ = $\frac{1}{2}$ ||x'- D'$_\Lambda$ $\alpha_\Lambda^{\star'}$||$_2^2$ + λ$\left\|\alpha_\Lambda^{\star'}\right\|_1$ - ($\frac{1}{2}$ ||x− D_Λ $\alpha_\Lambda^{\star}$||$_2^2$ + λ$\left\|\alpha_\Lambda^{\star}\right\|_1$)

Using cauchy-schwarz inequality I can simplify this so is easy to show that it's Lipschitz with constant e1|| D_Λ − D′$_\Lambda$ ||$_F$+ e2||x−x ′ ||$_2$

- ||α ⋆′ −α ⋆ ||2 = ||α ⋆ Λ ′ −α ⋆ Λ||$_2$ ≤ (1 /κ2) e1|| D_Λ − D′$_\Lambda$ ||$_F$+ e2||x−x ′ ||$_2$
- Therefore, α ⋆ is locally Lipschitz. Since K × C is compact, α ⋆ is uniformly Lipschitz on K × C, which concludes the proof.
- Now we are ready for our proof

Step 3 : preposition 1

- Given assumptions (A)–(C), let us now show that our algorithm converges to a stationary point of the objective function
- Challenges that the objective function is non-convex
- Proposition 1

  - $\bar{f}_t(\mathbf{D}_t)$ *converges a.s.;*
  - $f(\mathbf{D}_t) - \hat{f}_t(\mathbf{D}_t)$ *converges a.s. to* 0*; and*
  - $f(\mathbf{D}_t)$ *converges a.s.*

1

- By Bottou (1998) We prove the convergence of the sequence $\breve{f}_t$ by showing that the stochastic positive process
  - $u_t \triangleq \breve{f}_t \geq 0$
- Sufficient condition of convergence for a stochastic process,

Bottou (1998) and references therein (Metivier, 1983; Fisk, 1965)]

Let (Ω,F ,P) be a measurable probability space, u_t , for t ≥ 0, be the realization of a stochastic process and Ft be the filtration determined by the past information at

time let δt $\begin{cases} 1 \; if E[u_{t+1} - u_t \,|Ft] > 0 \\ 0 \end{cases}$

If for all t, $u_t \geq 0$ and $\sum_{t=1}^{\infty} E[\delta t(u_{t+1} - u_t)] < \infty$,

then u_t is a quasi-martingale and converges almost surely.

$\sum_{t=1}^{\infty} E[(u_{t+1} - u_t)] < \infty$

- $u_{t+1} - u_t = \widetilde{f_{t+1}}(D_{t+1}) - \breve{f}_t(D_t)$

➢ $= \widetilde{f_{t+1}}(D_{t+1}) - \widetilde{f_{t+1}}(D_t) + \widetilde{f_{t+1}}(D_t) - \breve{f}_t(D_t)$

➢ $\widetilde{f_{t+1}}(D_t) = \frac{1}{t+1}\sum_{i=1}^{t}(\frac{1}{2}||x_i - D\alpha_i||_2^2 + \lambda ||\alpha_i||_1) + \frac{1}{t+1}\ell(x_{t+1}, D_t)$

➢ $\frac{1}{t+1}\sum_{i=1}^{t}(\frac{1}{2}||x_i - D\alpha_i||_2^2 + \lambda ||\alpha_i||_1) + \frac{t\,\breve{f}_t(D_t)}{t+1}$

➢ Such that $\widetilde{f_{t+1}}(D_t) = \frac{\ell(x_{t+1}, D_t)}{t+1} + \frac{t\,\breve{f}_t(D_t)}{t+1}$

➢ $\widetilde{f_{t+1}}(D_{t+1}) - \widetilde{f_{t+1}}(D_t) + \frac{\ell(x_{t+1}, D_t) - f_t(D_t)}{t+1} + \frac{f_t(D_t) - \breve{f}_t(D_t)}{t+1}$ ----------→5

➢ again $\widetilde{f_{t+1}}(D_{t+1}) < \widetilde{f_{t+1}}(D_t)$ we also have $f_t(D_t) < \breve{f}_t(D_t)$

- $E[u_{t+1} - u_t \mid Ft] \leq \frac{E[\ell(x_{t+1}, D_t) \mid Ft] - f_t(D_t)}{t+1}$
- $\leq \frac{f(D_t) - f_t(D_t)}{t+1}$
- $\leq \frac{||f - f_t||_\infty}{t+1}$

$$\hat{f}_t(\mathbf{D}) \triangleq \frac{1}{t}\sum_{i=1}^{t}\left(\frac{1}{2}||\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i||_2^2 + \lambda||\boldsymbol{\alpha}_i||_1\right),$$

"A corollary of Donsker theorem see Van der Vaart, 1998, chap. 19.2, lemma 19.36 and example 19.7].

Let F = $\{f_\theta : \chi \to R, \theta \in \Theta\}$ be a set of measurable functions indexed by a bounded subset Θ of R^d

. Suppose that there exists a constant K such that $| f_{\theta 1}(x) - f_{\theta 2}(x)| \leq K ||\theta 1 - \theta 2||_2$, for every θ1 and θ2 in Θ and x in χ.

Then, F is P-Donsker. For any f in F, Let us define f $P_n f$, Pf and Gn f as

P_n f = (1/ n) $\sum_{i=1}^{n}$ f(Xi),Pf = E[f(X)], Gn f = √ n(P_n f −Pf). Let us also suppose that for all f , Pf 2 < δ 2 and $||f||_\infty$ < M and that the random elements X1,X2,... are Borel-measurable.

Then, we have E_P ||Gn||F = O(1), where ||Gn||F = sup ||Gn f|| "

- It is easy to show that in our case, all the hypotheses are verified,
- ℓ(x,.) is uniformly Lipschitz and bounded since it is continuously differentiable on a compact set, the set C ⊂ R m×k is bounded,
- and Ex[ℓ(x,D)^2] exists and is uniformly bounded.
- Therefore, Donsker theorem applies and there exists a constant κ > 0 such that
  - **E[E[u_{t+1} −u_t |Ft]] ≤ $\frac{\kappa}{t^{3/2}}$.**

Thus, we can apply Theorem 6, which proves that ut converges almost surely and that

$$\sum_{t=1}^{\infty} E[u_{t+1} - u_t \mid Ft]] \leq \infty.$$

2

Using Eq. (5) we can show that it implies the almost sure convergence of the positive sum $\frac{\sum_{t=1}^{\infty} f_t(D_t) - \tilde{f}_t(D_t)}{t+1}$.

similar to Bertsekas (1999, prop 1.2.4)

Let an, bn be two real sequences such that for all n, an ≥ 0,bn ≥ 0, $\sum_{n=1}^{\infty}$ an = ∞, $\sum_{n=1}^{\infty}$ an bn < ∞, ∃K > 0 s.t. |bn+1 −bn| < Kan. Then, limn→+∞ bn = 0

- **Using Lemma 1 and the fact that the functions ft and $\tilde{f}_t$ are bounded and Lipschitz, with a constant independent of t**

. Since $\tilde{f}_t$ converges almost surely, this shows that f_t converges in probability to the same limit

$$f_t(\mathbf{D}_t) - \hat{f}_t(\mathbf{D}_t) \underset{t \to +\infty}{\longrightarrow} 0 \text{ a.s.}$$

Now f_t converges almost surely, which proves the second and third points.

With Proposition 3 in hand, we can now prove our final and strongest result, namely that first order necessary optimality conditions are verified asymptotically with probability one

Propostion 2 : Convergence to a stationary point

- Under assumptions (A) to (C), the distance between Dt and the set of stationary points of the dictionary learning problem converges almost surely to 0 when t tends to infinity
- Let us assume for a moment that these sequences converge respectively to two matrices A∞ and B∞. In that case, Dt converges to a matrix D∞ in C Let U be a matrix in $R^{m\times k}$. Since $\tilde{\boldsymbol{f}}_{\boldsymbol{t}}(D_t)$ upperbounds $\boldsymbol{f}_{\boldsymbol{t}}$ on $R^{m\times k}$. , for all t

$$\tilde{\boldsymbol{f}}_{\boldsymbol{t}}(D_t + U) \geq \boldsymbol{f}_{\boldsymbol{t}}(D_t + \boldsymbol{U})$$

Taking the limit when t tends to infinity

$$\widetilde{\boldsymbol{f}_{\infty}}(D_\infty + U) \geq \boldsymbol{f}_{\boldsymbol{t}}(D_\infty + \boldsymbol{U})$$

Let $h_t > 0$ be a sequence that converges to 0. Using a first order Taylor expansion, and using the fact that ∇ f is Lipschitz and $\widetilde{\boldsymbol{f}_{\infty}}(D_\infty) = \boldsymbol{f}_{\boldsymbol{t}}(D_\infty)$ we have

$$\widetilde{\boldsymbol{f}_{\infty}}(D_\infty) + \mathrm{Tr}(h_t U^T \nabla \widetilde{\boldsymbol{f}_{\infty}}(D_\infty)) + o(h_t U) \geq \boldsymbol{f}_{\boldsymbol{t}}(D_\infty) + \mathrm{Tr}(h_t U^T \boldsymbol{f}_{\boldsymbol{t}}(D_\infty)) + o(h_t U),$$

- Since this inequality is true for all U, $\nabla \widetilde{f_\infty}(D_\infty) = \nabla f_t(D_\infty)$
- A first-order necessary optimality condition for D_∞ being an optimum of $\widetilde{f_\infty}$ is that $-\nabla \widetilde{f_\infty}$ **is in the normal cone of the set C at** $\boldsymbol{D_\infty}$ (Borwein and Lewis, 2006).
- Therefore, this first-order necessary conditions is verified for f at D_∞ as well.
- Since At ,Bt are asymptotically close to their accumulation points,

$-\nabla f(D_t)$ is asymptotically close the normal cone at Dt and these first-order optimality conditions are verified asymptotically with probability one

EXPERIMENTAL VALIDATION

AGENDA

Performance Evaluation for Dictionary Learning

Application to Large-Scale Image Processing

Introduction

In this section, we present experiments on natural images to demonstrate the efficiency of our method for dictionary learning.

TOPIC ONE

Performance Evaluation for Dictionary Learning

Performance Evaluation

- For our experiments, we randomly selected 1.25×10^6 patches from images in the Berkeley segmentation dataset, which is a standard image database.
- 10^6 of these are kept for training, and the rest for testing.
- We used these patches to create three datasets A, B, and C with increasing patch and dictionary sizes representing various typical settings in image processing applications:

Data	Signal size *m*	Nb *k* of atoms	Type
A	8 x 8 = 64	256	b&w
B	12 x 12 x 3 = 432	512	color
C	16 x 16 = 256	1024	B&w

Performance Evaluation

- We have normalized the patches to have unit l_2-norm and used the regularization parameter $\lambda = 1.2 / \sqrt{m}$ in all of our experiments.
- The $1 / \sqrt{m}$ term is a classical normalization factor, and the constant 1.2 has been experimentally shown to yield reasonable sparsities (about 10 nonzero coefficients) in these experiments.
- We have implemented the proposed algorithm in C++ with a Matlab interface.
- All the results presented in this section use the mini-batch refinement from Section 3.4 since this has shown empirically to improve speed by a factor of 10 or more.

Performance Evaluation

- This requires to turn the parameter η, the number of signals drawn at each iteration.
- Trying different powers of 2 for this variable has shown that $\eta = 256$ was a good choice (lowest objective function values on the training set — empirically, this setting also yields the lowest values on the test set), but values of 128 and 512 have given very similar performances.
- Our implementation can be used in both the online setting it is intended for, and in a regular batch mode where it uses the entire dataset at each iteration (corresponding to the mini-batch version with $\eta = n$).

Performance Evaluation

- We have also implemented a first-order stochastic gradient descent algorithm that shares most of its code with our algorithm, except for the dictionary update step.
- This setting allows us to draw meaningful comparisons between our algorithm and its batch and stochastic gradient alternatives, which would have been difficult otherwise.
- For example, comparing our algorithm to the Matlab implementation of the batch approach from (Lee et al., 2007) developed by its authors would have been unfair since our C++ program has a built-in speed advantage.

Performance Evaluation

- Although our implementation is multi-threaded, our experiments have been run for simplicity on a single-CPU, single-core 2.4Ghz machine.
- To measure and compare the performances of the three tested methods, we have plotted the value of the objective function on *the test set*, acting as a surrogate of the expected cost, as a function of the corresponding training time.

Online vs Batch

- Figure 1 compares the online and batch settings of our implementation.

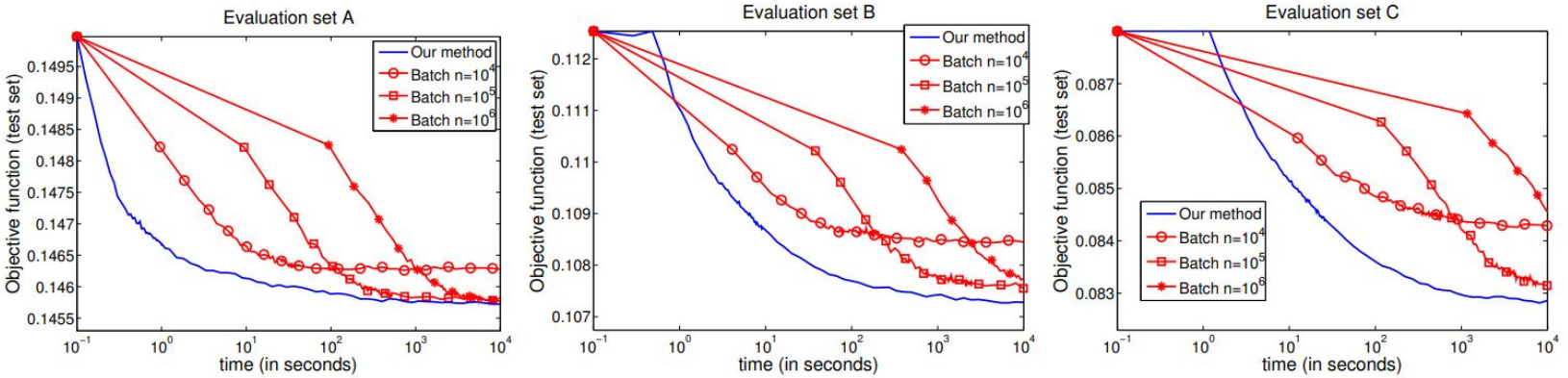


Figure 1. Comparison between online and batch learning for various training set sizes.

Online vs Batch

- The full training set consists of 10^6 samples.
- The online version of our algorithm draws samples from the entire set, and we have run its batch version on the full dataset as well as subsets of size 10^4 and 10^5.
- The online setting systematically outperforms its batch counterpart for every training set size and desired precision.

Online vs Batch

- We use a logarithmic scale for the computation time, which shows that in many situations, the difference in performance can be dramatic.
- Similar experiments have given similar results on smaller datasets.

Comparison with Stochastic Gradient Descent

- Our experiments have shown that obtaining good performance with stochastic gradient descent requires using both the mini-batch heuristic *and* carefully choosing the learning rate ρ.
- To give the fairest comparison possible, we have thus optimized these parameters, sampling η values among powers of 2 and ρ values among powers of 10.
- The combination of values $\rho = 10^4, \eta = 512$ gives the best results on the training and test data for stochastic gradient descent.

Comparison with Stochastic Gradient Descent

- Figure 2 compares our method with stochastic gradient descent for different ρ values around 10^4 and a fixed value of $\eta = 512$.

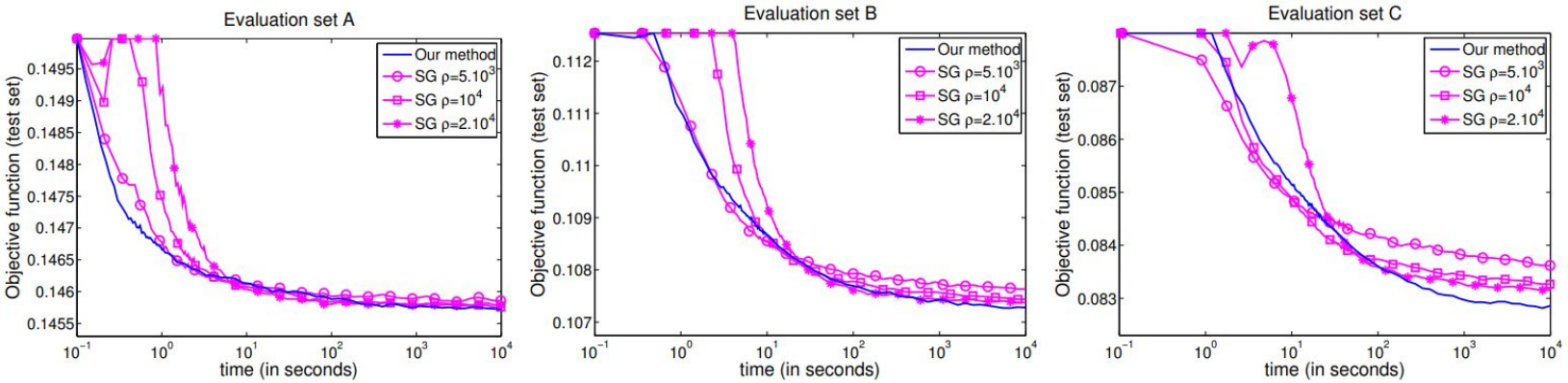


Figure 2. Comparison between our method and stochastic gradient (SG) descent with different learning rates ρ.

Comparison with Stochastic Gradient Descent

- We observe that the larger the value of ρ is, the better the eventual value of the objective function is after many iterations, but the longer it will take to achieve a good precision.
- Although our method performs better at such high-precision settings for dataset C, it appears that, in general, for a desired precision and a particular dataset, it is possible to tune the stochastic gradient descent algorithm to achieve a performance similar to that of our algorithm.

Comparison with Stochastic Gradient Descent

- Note that both stochastic gradient descent and our method only start decreasing the objective function value after a few iterations.
- Slightly better results could be obtained by using smaller gradient steps during the first iterations, using a learning rate of the form $\rho / (t + t_o)$ for the stochastic gradient descent, and initializing $\mathbf{A}_0 = t_0\mathbf{I}$ and $\mathbf{B}_0 = t_0\mathbf{D}_0$ for the matrices $\mathbf{A}_t$ and $\mathbf{B}_t$, where t_0 is a new parameter.

TOPIC TWO

Application to Large-Scale Image Processing

Application to Inpainting

- Our last experiment demonstrates that our algorithm can be used for a difficult large-scale image processing task, namely, removing the text (*inpainting*) from the damaged 12-Megapixel image of Figure 3.
- Using a multi-threaded version of our implementation, we have learned a dictionary with 256 elements from the roughly 7×10^6 undamaged 12×12 color patches in the image with two epochs in about 500 seconds on a 2.4GHz machine with eight cores.

Application to Inpainting

Figure 3. Inpainting example on a 12-Megapixel image. Top: Damaged and restored images. Bottom: Zooming on the damaged and restored images.

Application to Inpainting

- Once the dictionary has been learned, the test is removed using the sparse coding technique for inpainting of Mairal et al. (2008b).
- Our intent here is of course *not* to evaluate our learning procedure in inpainting tasks, which would require a thorough comparison with state-of-the-art techniques on standard datasets.
- Instead, we just wish to demonstrate that the proposed method can indeed be applied to a realistic, non-trivial image processing tasks on a large image.

Application to Inpainting

- Indeed, to the best of our knowledge, this is the first time that dictionary learning is used for image restoration on such large-scale data.
- For comparison, the dictionaries used for inpainting in the state-of-the-art method of Mairal et al. (2008b) are learned (in batch mode) on only 200,000 patches.

SUMMARY

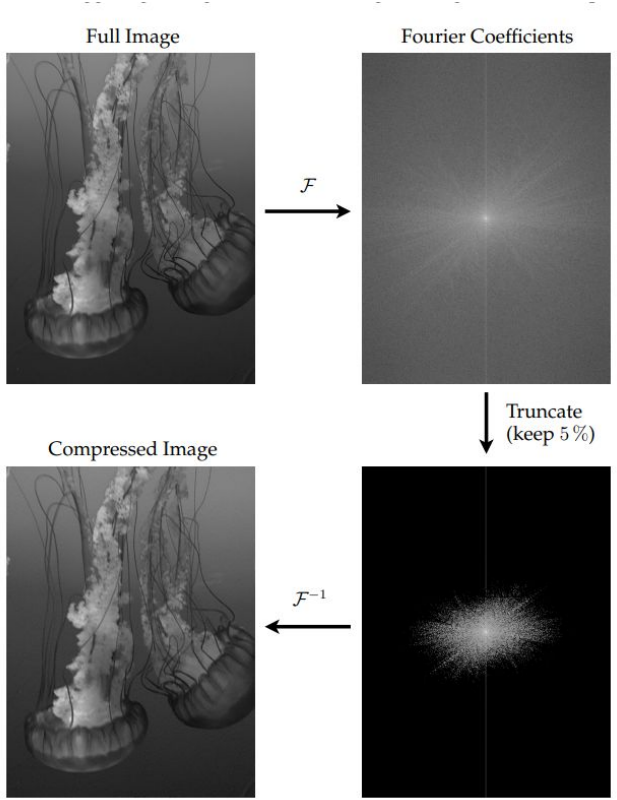
- The authors presented an experimental validation of their online dictionary learning algorithm.
- They assessed the performance on a standard image database known as the Berkeley segmentation dataset.
- They compared their implementation to Batch and SGD techniques.
- Finally, they demonstrated that their method can be applied to a realistic, non-trivial image processing task on a large image.

Compressed sensing

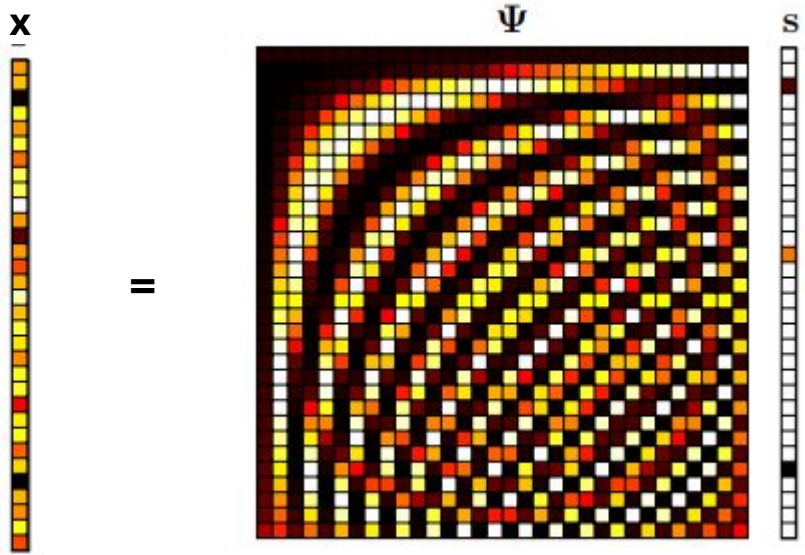
A signal processing technique for efficiently acquiring and reconstructing a signal, by finding solutions to underdetermined linear systems

if a signal is sparse or compressible in some domain, we can recover it from far fewer samples than required by the Nyquist–Shannon sampling theorem, which is the classical criterion for perfect reconstruction

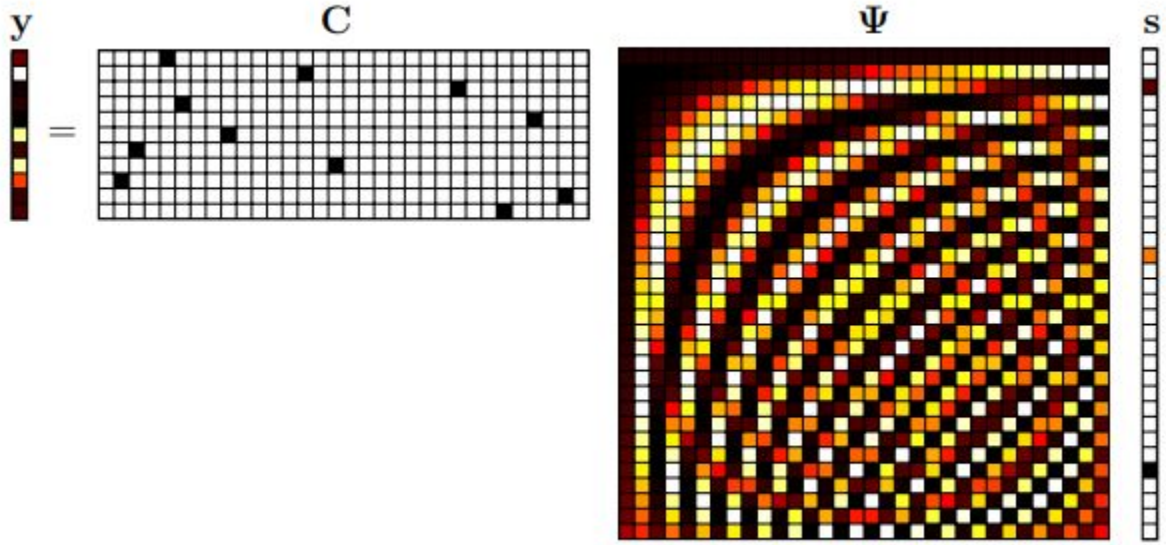
Compression with known basis like fourier or wavelet basis



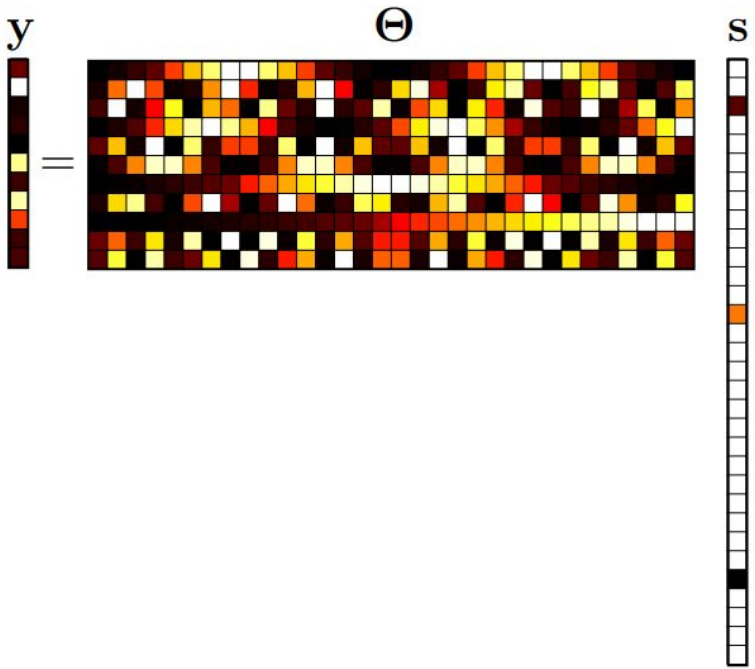
The original signal



After taking random samples



By multiplying the first 2 matrices



We can clearly see that this will result in an underdetermined system of equations which will result to an infinite number of solutions

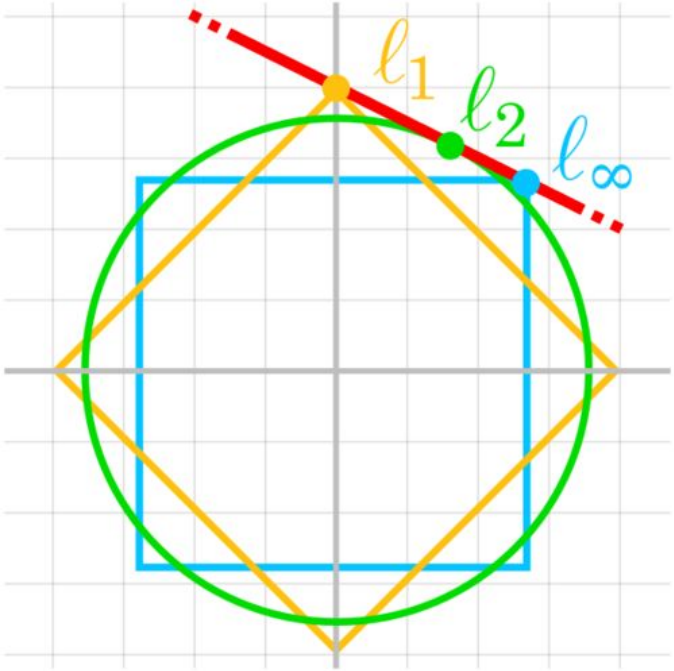
But how can we solve the equation

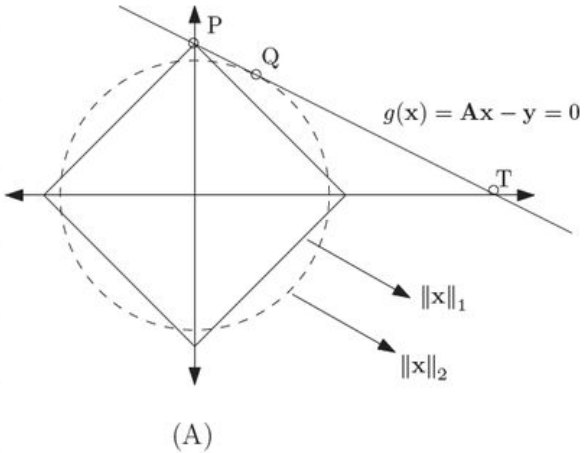
For variable S

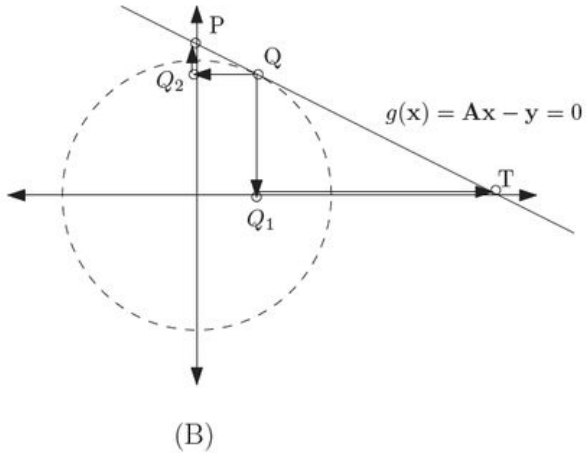
Minimize $\|S\|_1$

Subject to $\theta S = y$

Why does the l_1 norm finds the sparsest solution







For images we have noise

$Y = \theta S + e$

$e = y - \theta S$

For variable S

Minimize $\|S\|_1$

Subject to $\| \theta S - y \|_2 < eps$