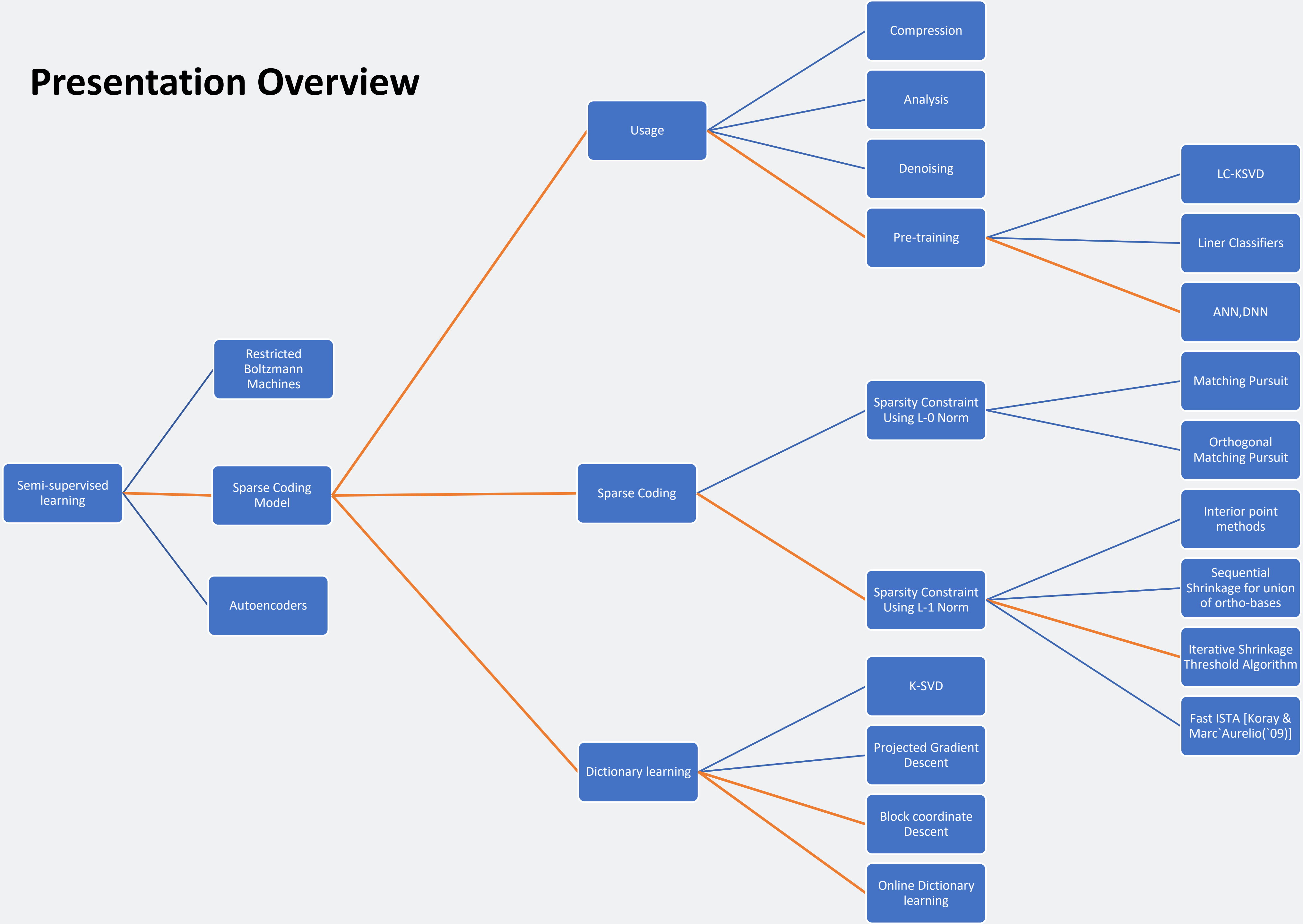
Sparse coding and dictionary learning

Presentation Overview

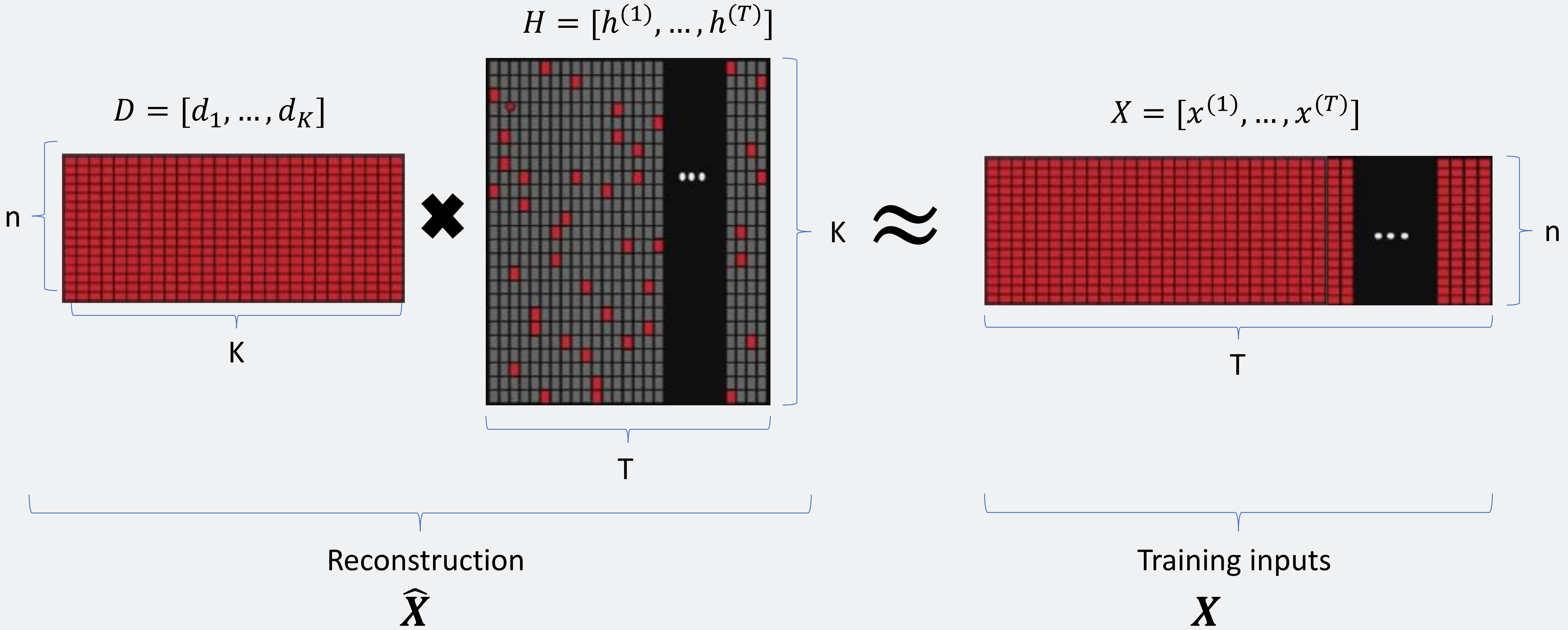


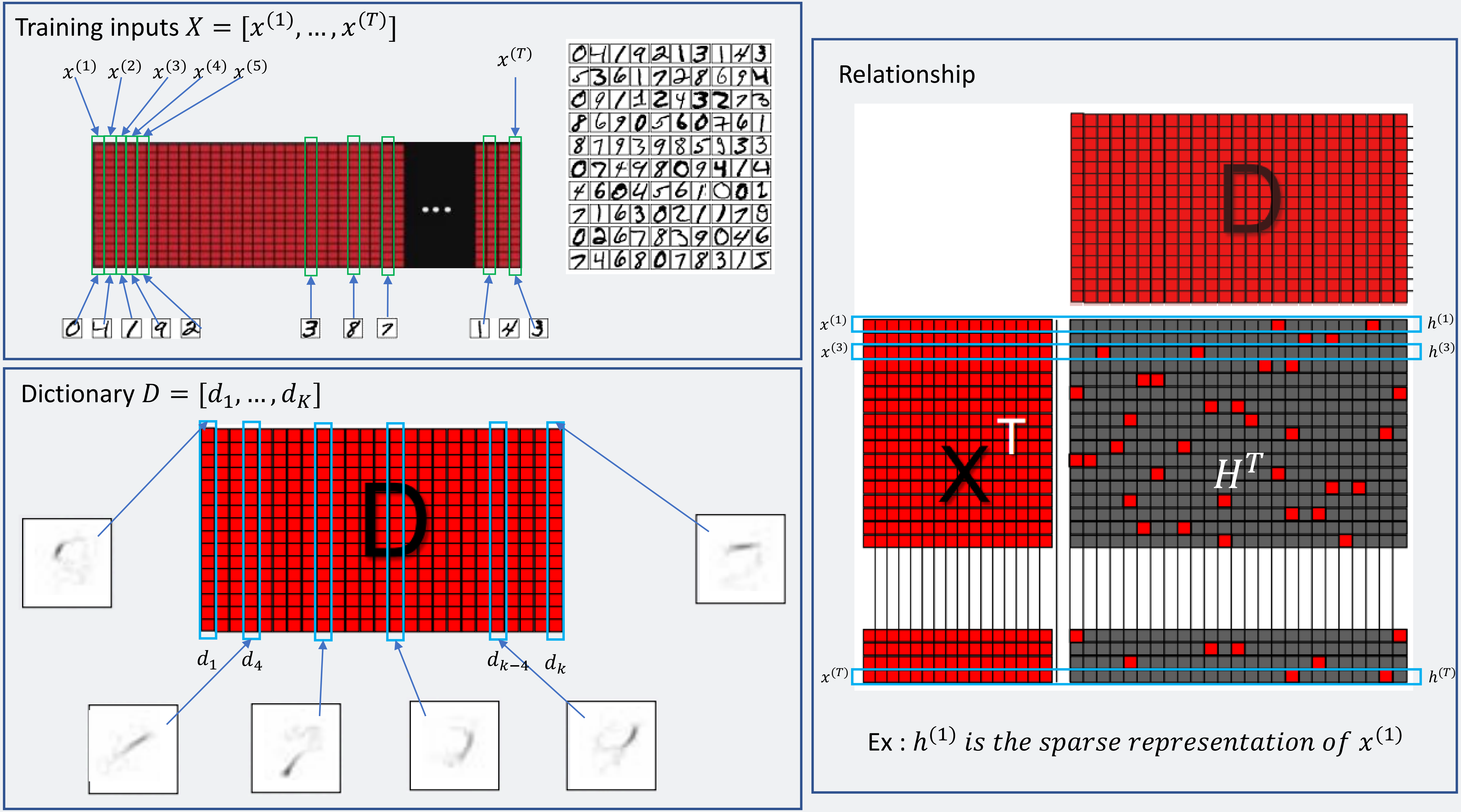
INTRODUCTION

$$x^{(t)} \longrightarrow \boxed{\text{SC Model}} \longrightarrow \hat{x}^{(t)} \qquad \boxed{\boldsymbol{x}^{(t)} \approx \hat{\boldsymbol{x}}^{(t)}}$$

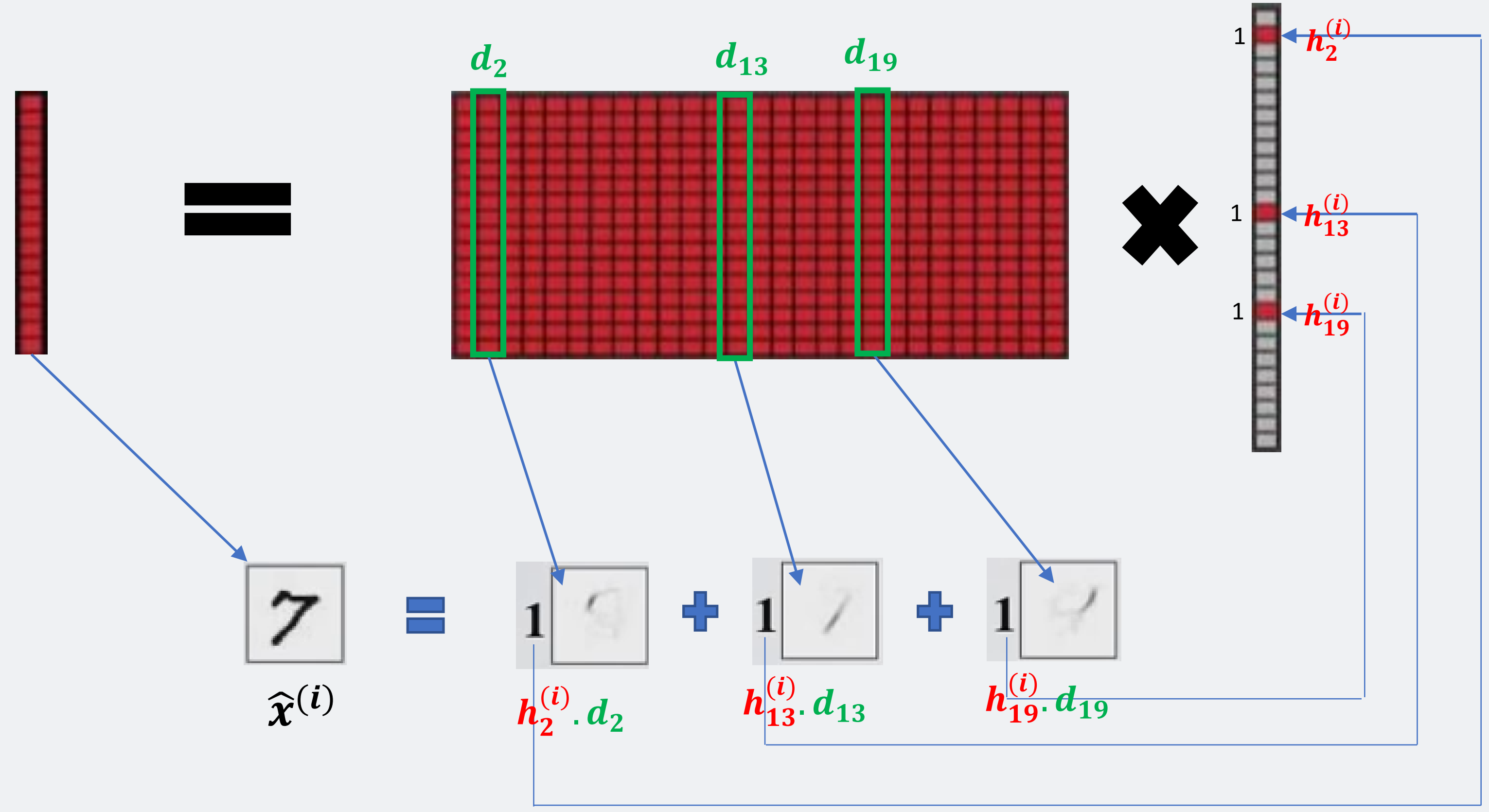
- An Unsupervised training procedure, only need to give the inputs
- Goal is to recreate the given input at the output
- While doing that, model will extract a sparse representation of the input vector
- Sparse representation of an input vector is very unique with respect to a given Dictionary.
- Dictionary consists of most dominant and principle components which will explain entire training set
- Because of that we can use this sparse representation for classification instead of using raw inputs.
- Due to that reason people use Sparse coding with Supervised learning and it’s a very robust and efficient method in classification

$$\min_{\mathbf{D}} \frac{1}{T}\sum_{t=1}^{T}\min_{\mathbf{h}^{(t)}} \frac{1}{2}\overbrace{||\mathbf{x}^{(t)} - \underbrace{\mathbf{D}\,\mathbf{h}^{(t)}}_{\text{reconstruction } \widehat{\mathbf{x}}^{(t)}}||_2^2}^{\text{reconstruction error}} + \underbrace{\lambda}_{\substack{\text{reconstruction vs.} \\ \text{sparsity control}}}\overbrace{||\mathbf{h}^{(t)}||_1}^{\text{sparsity penalty}}$$




Training inputs $X = [x^{(1)}, \dots, x^{(T)}]$
$x^{(1)}$ $x^{(2)}$ $x^{(3)}$ $x^{(4)}$ $x^{(5)}$
$x^{(T)}$
Dictionary $D = [d_1, \dots, d_K]$
D
d_1 d_4 d_{k-4} d_k
Relationship
D
X^T
H^T
$x^{(1)}$ $h^{(1)}$
$x^{(3)}$ $h^{(3)}$
$x^{(T)}$ $h^{(T)}$
Ex : $h^{(1)}$ *is the sparse representation of* $x^{(1)}$

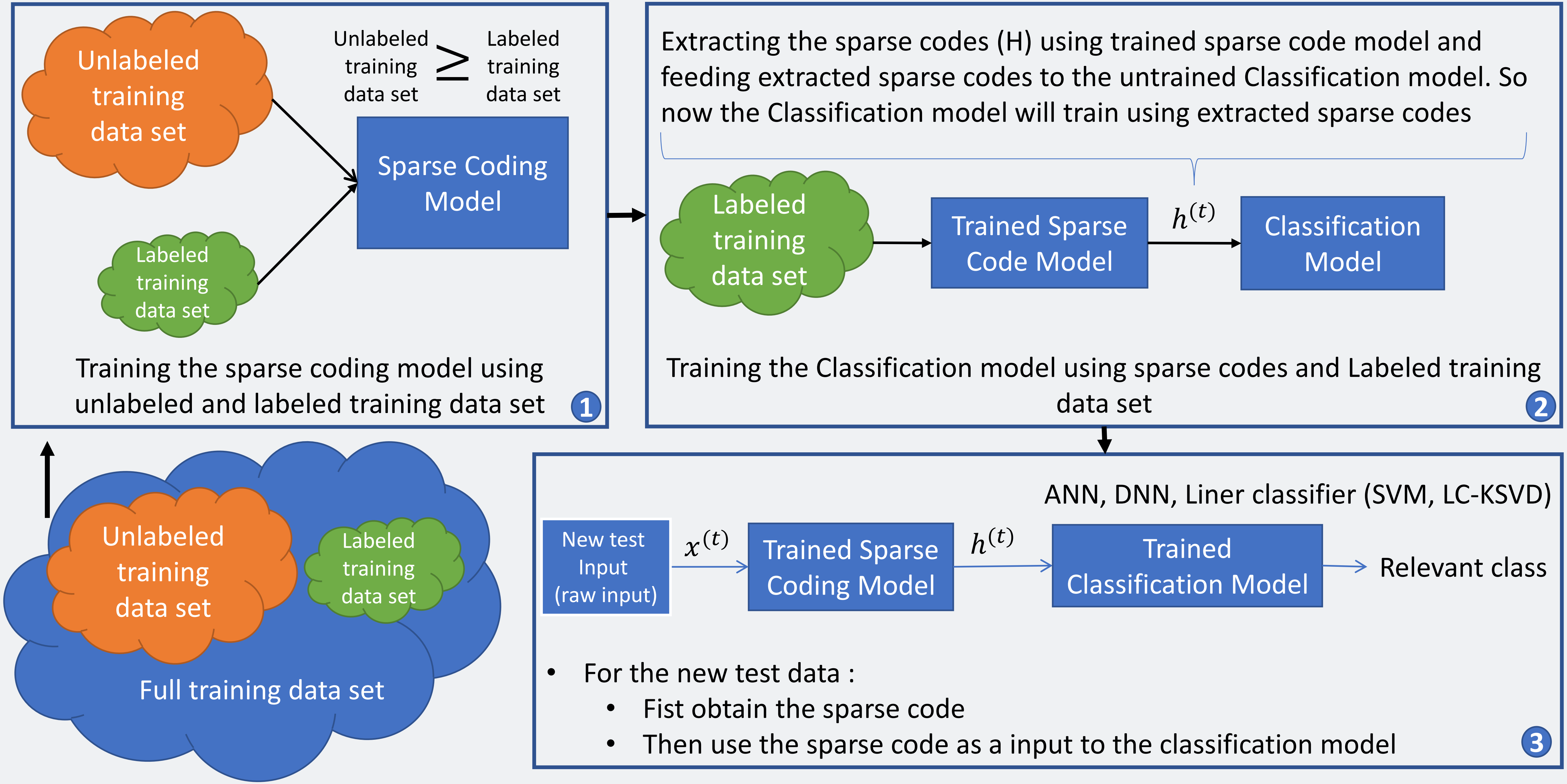
Reconstruction Example of one single input $x^{(i)}$ =



In order to reconstruct the input $x^{(i)}$, $h^{(i)}$ will tell you which components needed to be added from the dictionary, and with which frequencies

$$\boxed{x^{(i)} \approx \hat{x}^{(i)} = Dh^{(i)}}$$

For classification – semi supervised learning



Training Process

$$\min_{\mathbf{D}} \frac{1}{T}\sum_{t=1}^{T} \min_{\mathbf{h}^{(t)}} \frac{1}{2}\overbrace{||\mathbf{x}^{(t)} - \underbrace{\mathbf{D}\,\mathbf{h}^{(t)}}_{\text{reconstruction } \widehat{\mathbf{x}}^{(t)}}||_2^2}^{\text{reconstruction error}} + \underbrace{\lambda}_{\text{reconstruction vs. sparsity control}}\overbrace{||\mathbf{h}^{(t)}||_1}^{\text{sparsity penalty}}$$

- Learning will be alternating between computing $h^{(t)}$ (sparse codes for each training input) and updating the Dictionary
- This process will go back and forth until it satisfies the convergence requirement
- Constrain the columns of D to be of unit norm $||d_i||_2 = 1$
- If not D could grow bigger while $h^{(t)}$ becomes small to satisfy the sparsity constraints
- First, consider optimizing **H** and after that optimize **D**

Optimizing $h^{(t)}$

Solving for $h^{(t)}$

Sparsity Constraint Using l_0-Norm

- Matching Pursuit
- Orthogonal Matching Pursuit

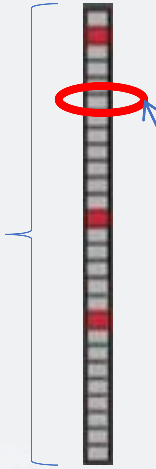
Relaxation

Sparsity Constraint Using l_1-Norm

- Interior point methods [Chen,Donoho, & Saunders (`95)]
- Sequential Shrinkage for union of ortho-bases[Bruce et.al(`98)]
- Iterative Shrinkage Threshold Algorithm [Figuerido & Nowak(`03)]
- Fast ISTA [Koray & Marc`Aurelio(`09)]
- Coordinate descent algorithm [Learning fast approximation for sparse coding Gregor and Lecun (`10)]

Iterative Shrinkage Threshold Algorithm(ISTA)

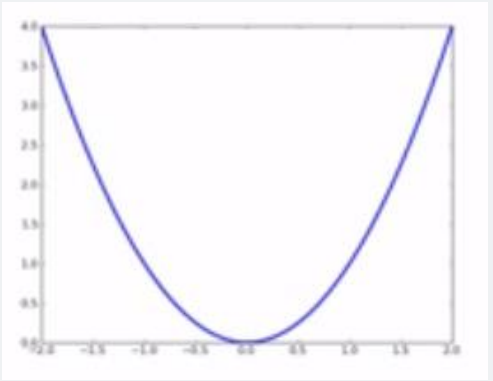
$$\mathbf{h}(\mathbf{x}^{(t)}) = \arg\min_{\mathbf{h}^{(t)}} \frac{1}{2}||\mathbf{x}^{(t)} - \mathbf{D}\,\mathbf{h}^{(t)}||_2^2 + \lambda||\mathbf{h}^{(t)}||_1$$



$h^{(t)}$

$h_k^{(t)}$

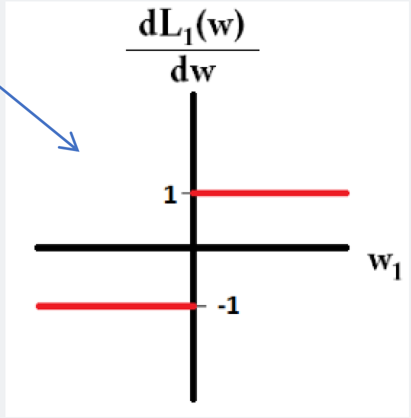
$\frac{1}{2}||\mathbf{x}^{(t)} - \mathbf{D}\,\mathbf{h}^{(t)}||_2^2$

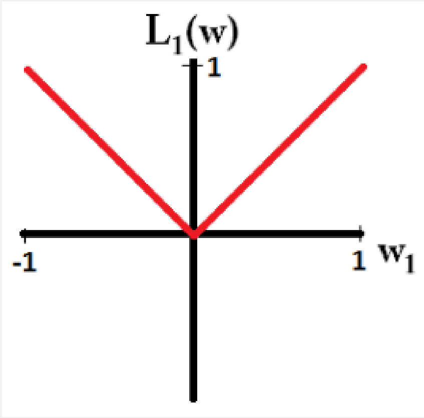


Now the optimization will be $l(\mathbf{x}^{(t)}) = \frac{1}{2}||\mathbf{x}^{(t)} - \mathbf{D}\,\mathbf{h}^{(t)}||_2^2 + \lambda||\mathbf{h}^{(t)}||_1$ w.r.t. $\mathbf{h}^{(t)}$

Gradient Descend method can be apply $\nabla_{\mathbf{h}^{(t)}} l(\mathbf{x}^{(t)}) = \mathbf{D}^\top(\mathbf{D}\,\mathbf{h}^{(t)} - \mathbf{x}^{(t)}) + \lambda\,\mathrm{sign}(\mathbf{h}^{(t)})$

For a single element in $h^{(t)}$ $\frac{\partial}{\partial h_k^{(t)}} l(\mathbf{x}^{(t)}) = (\mathbf{D}_{\cdot,k})^\top(\mathbf{D}\,\mathbf{h}^{(t)} - \mathbf{x}^{(t)}) + \underline{\lambda\,\mathrm{sign}(h_k^{(t)})}$





Algorithm try to minimize $h_k^{(t)}$ to zero in order to get a sparse representation

But L_1 norm not differentiable at 0 so its very unlikely for gradient descent to land on zero even if it is the solution

Solution is if $h_k^{(t)}$ changes the sign because of L_1 norm gradient, then it should be clamp to zero

$\boxed{\text{Assume } \boldsymbol{w_1} = \boldsymbol{h_k^{(t)}}}$

$||\mathbf{h}^{(t)}||_1$

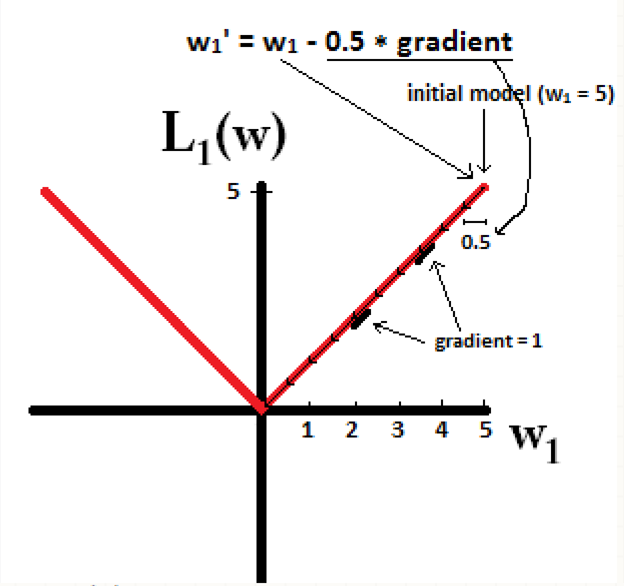
- **Each element $h^{(t)}{}_k$ update :**

- $h_k^{(t)} \Leftarrow h_k^{(t)} - \alpha\left(D_{.,k}\right)^T\left(Dh^{(t)} - x^{(t)}\right)$ Update from Reconstruction
- $if\ sign\left(h_k^{(t)}\right) \neq sign\left(h_k^{(t)} - \alpha\lambda sign\left(h_k^{(t)}\right)\right)$
  - $then: h_k^{(t)} \Leftarrow 0$
- $else$
  - $h_k^{(t)} \Leftarrow h_k^{(t)} - \alpha\lambda sign\left(h_k^{(t)}\right)$

Update from sparsity

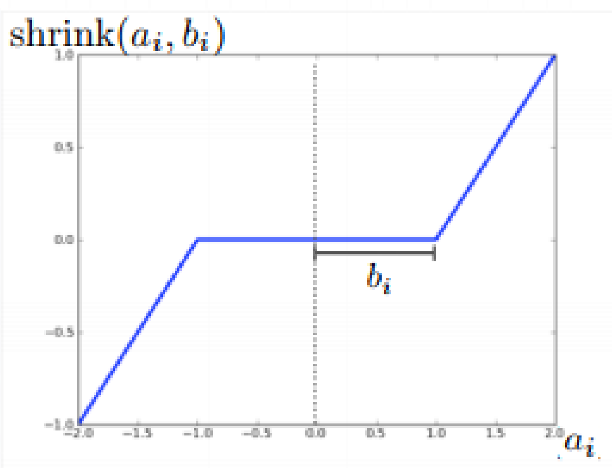
- **Algorithm**

- $initialize\ h^{(t)}\ (for\ instance\ to\ 0)$
- $while\ h^{(t)}\ has\ not\ converged$
  - $h^{(t)} \Leftarrow h^{(t)}\ -\ \alpha D^T(Dh^{(t)} - x^{(t)})$
  - $h^{(t)} \Leftarrow shrink\ (h^{(t)}, \alpha\lambda)$
- $return\ h^{(t)}$



$w_1 := w_1 - \eta \cdot \frac{dL_1(w)}{dw} = w_1 - \frac{1}{2} \cdot 1$, until reaching a model with $w_1 = 0$

Assume $\boldsymbol{w_1} = \boldsymbol{h_k^{(t)}}$



$shrink(a, b) = [sign(a_i)\ \max(|a_i| - b_i\ , 0)]$

Will converge if $\frac{1}{\alpha}$ is bigger than the largest eigenvalue of $D^T D$

Dictionary learning

- K-SVD
  - The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representations [M. Aharon, M. Elad, and A. M. Bruckstein (`06)]
- Projected Gradient Descent
- Block coordinate Descent
- Online Dictionary learning
  - Online Dictionary Learning for Sparse Coding [Mairal,Bach,Ponce and Sapiro (`09)]

Block Coordinate Descent

- Now the optimization will be
- $min_D \ \frac{1}{T}\sum_{t=1}^{T} min_{h^{(t)}} \ l\left(x^{(t)}\right) = \ min_D \ \frac{1}{T}\sum_{t=1}^{T}\frac{1}{2}\left\|x^{(t)} - Dh^{(t)}\right\|_2^2 + \ \lambda\left\|h^{(t)}\right\|_1$

- For a given $h^{(t)}$
- $min_D \ \frac{1}{T}\sum_{t=1}^{T}\frac{1}{2}\left\|x^{(t)} - Dh^{(t)}\right\|_2^2$

- we must also constrain the columns of D to be of unit norm ($\left\|D_{.,j}\right\|_2 = 1$)
- This algorithm doesn't need a learning rate
- Idea of this algorithms is to minimize one column at a time

Set the gradient of optimization function to zero with respect to $D_{.,j}$

$$0 = \nabla_{D_{.,j}} \left(\frac{1}{T}\sum_{t=1}^{T}\frac{1}{2}\left\|x^{(t)} - Dh^{(t)}\right\|_2^2\right)$$

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

$$0 = \sum_{t=1}^{T}\left(x^{(t)} - \left(\sum_{i\neq j} D_{.,i}h_i^{(t)}\right) - D_{.,j}h_j^{(t)}\right)h_j^{(t)}$$

$$\sum_{t=1}^{T} D_{.,j}{h_j^{(t)}}^2 = \sum_{t=1}^{T}\left(x^{(t)} - \left(\sum_{i\neq j} D_{.,i}h_i^{(t)}\right)\right)h_j^{(t)}$$

$$D_{.,j} = \frac{1}{\sum_1^T {h_j^{(t)}}^2}\sum_{t=1}^{T}\left(x^{(t)} - \left(\sum_{i\neq j} D_{.,i}h_i^{(t)}\right)\right)h_j^{(t)}$$

If $A = \sum_{t=1}^{T} h^{(t)}{h^{(t)}}^T$ and $B = \sum_{t=1}^{T} x^{(t)}{h^{(t)}}^T$

$$D_{.,j} = \underbrace{\frac{1}{\sum_1^T {h_j^{(t)}}^2}}_{A_{j,j}}\left(\underbrace{\left(\sum_{t=1}^{T} x^{(t)}h_j^{(t)}\right)}_{B_{.,j}} - \sum_{i\neq j} D_{.,i}\underbrace{\left(\sum_{t=1}^{T} h_i^{(t)}h_j^{(t)}\right)}_{A_{i,j}}\right)$$

$$D_{.,j} = \frac{1}{A_{j,j}}\left(B_{.,j} - \sum_{i\neq j} D_{.,i}A_{i,j}\right)$$

Find the best value for j^{th} column with respect to all the other parameters (including fixing of other columns of **D** matrix)

Then iterate cycles over each columns $1^{st} \rightarrow 2^{nd} \rightarrow 3^{rd} \rightarrow 4^{th} \ldots \rightarrow 1^{st} \ldots$ until we get an stable values for **D**

Pseudocode for block coordinate descent algorithm

$while\ \boldsymbol{D}\ hasn't\ converged$

$for\ each\ column\ D_{.,j}\ perform\ updates$

$D_{.,j} = \frac{1}{A_{j,j}}\left(B_{.,j} - \sum_{i \neq j} D_{.,i}\, A_{i,j}\right)$ ← coordinate descent update

$D_{.,j} = \frac{D_{.,j}}{\left\|D_{.,j}\right\|_2}$ ← projection ; normalizing to satisfy the column constrain of D (unit norm)

Implementation note :-

Since **A** and **B** not depend of **D**, we can calculate **A** and **B** before doing any optimization to **D** and keep it in memory. It will be efficient and save lot of resources

Complete Pseudo-code For Sparse Coding And Dictionary Learning

- $Initialize\ D\ and\ normalize\ the\ columns$
- $while\ \boldsymbol{D}\ has\ not\ converged:$
    - $find\ the\ sparse\ codes\ h^{(t)}\ \forall\ x^{(t)}\ in\ training\ set\ using\ \boldsymbol{ISTA}\ for\ a\ given\ \boldsymbol{D}$
    - $update\ the\ dictionary$
        - $compute\ \boldsymbol{A}\ and\ \boldsymbol{B}\ for\ calculated\ h^{(t)}$
        - $run\ \boldsymbol{block\ coordinate\ descend}\ algorithm$

- As we can see no learning rate will be required by Block coordinate descent
- Learning will be alternating between computing $h^{(t)}$ (sparse codes for each training input) and updating the Dictionary.
- This process will go back and forth until it satisfies the convergence requirement
- This method is related to batch learning category
- It means for a single update, it will pass through the whole given training set
- This method could be inefficient for a big amount of training data
- The solution is an online learning algorithm and this means algorithm update while data is coming
- This can apply context where constantly obtaining new data in online fashion (like downloading images, solar irradiance)

Online Dictionary Learning Algorithm

- This method required an update of running averages for **A** and **B**
- $B_{new} \Leftarrow \beta B_{pre} + (1-\beta) x^{(t)} {h^{(t)}}^{T}$
- $A_{new} \Leftarrow \beta A_{pre} + (1-\beta) h^{(t)} {h^{(t)}}^{T}$
- Now the previously mentioned A and B has changed with a running average
- β is a hyper parameter and if β close to 1 then model will give more priority to previous **B** and give less priority to incoming new data

Initializing D

- Not to 0
  - Because columns of D vectors should have unit norm
  - It is impossible to reconstruct and do the learning
- Use random matrix and then normalize each of the column
- Normalize first few training vectors and use those as columns in matrix D

Online Dictionary Learning Algorithm

- $Initialize\ \boldsymbol{D}$
- $While\ \boldsymbol{D}\ hasn't\ converged$
  - $for\ each\ x^{(t)}$
    - $infer\ sparse\ codes\ h^{(t)}\ using\ \boldsymbol{ISTA}$

      Run ISTA to extract the sparse code for new data
    - $update\ Dictionary$
      - $B_{new} \Leftarrow \beta B_{prev} + (1-\beta) x^{(t)} {h^{(t)}}^T$
      - $A_{new} \Leftarrow \beta A_{prev} + (1-\beta) h^{(t)} {h^{(t)}}^T$

      Calculate the running average
      - $while\ \boldsymbol{D}\ hasn't\ converged$
        - $for\ each\ D_{.,j}\ column\ perform\ Gradient\ update$
          - $D_{.,j} = \frac{1}{A_{j,j}}\left(B_{.,j} - \sum_{i \neq j} D_{.,i} A_{i,j}\right)$
          - $D_{.,j} = \frac{D_{.,j}}{\left\|D_{.,j}\right\|_2}$

      Run Block Coordinate Descent

Thank You