Sparse Coding and Dictionary Learning

Yuan Yao and Ruohan Zhan

Peking University

Reference: Andrew Ng

- http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial

Sparse Coding

- The aim is to find a set of basis vectors (dictionary) ϕ_i such that we can represent an input vector **x** as a linear combination of these basis vectors:

$$\mathbf{x} = \sum_{i=1}^{k} a_i \phi_i$$

- PCA: a complete basis
- Sparse coding: an *overcomplete* basis to represent $\mathbf{x} \in \mathbb{R}^n$ (i.e. such that $k > n$)
  - The coefficients a_i are no longer *uniquely* determined by the input vector **x**
  - Need additional criterion of **sparsity** to resolve the degeneracy introduced by over-completeness.

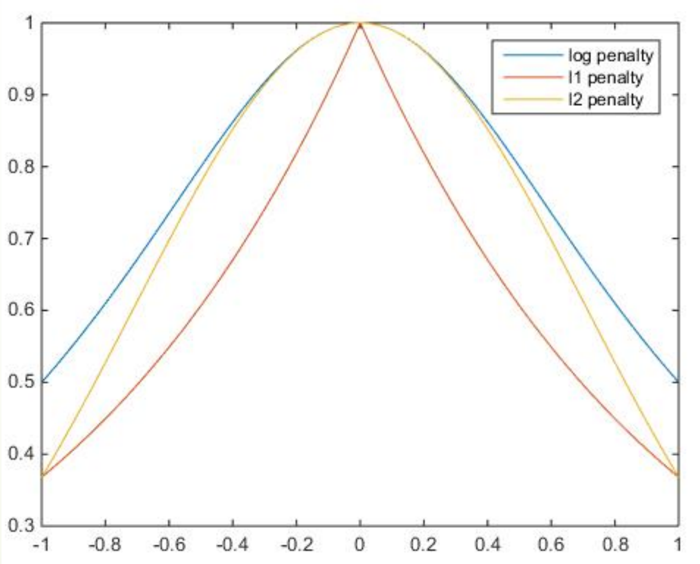
Sparsity Penalty

- We define the sparse coding cost function on a set of *m* input vectors as

$$\text{minimize}_{a_i^{(j)},\phi_i} \sum_{j=1}^{m} \left|\left| \mathbf{x}^{(j)} - \sum_{i=1}^{k} a_i^{(j)} \phi_i \right|\right|^2 + \lambda \sum_{i=1}^{k} S(a_i^{(j)})$$

where $S(.)$ is a sparsity cost function which penalizes a_i for being far from zero.

- "L_0-norm": $S(a_i) = \mathbf{1}(|a_i| > 0)$
- L_1 penalty: $S(a_i) = |a_i|_1$
- log penalty: $S(a_i) = \log(1 + a_i^2)$



Scale freedom

- In addition, it is also possible to make the sparsity penalty arbitrarily small by scaling down a_i and scaling ϕ_i up by some large constant.
- To prevent this from happening,

$$\begin{array}{rc} \text{minimize}_{a_i^{(j)},\phi_i} & \sum_{j=1}^{m} \left|\left|\mathbf{x}^{(j)} - \sum_{i=1}^{k} a_i^{(j)} \phi_i\right|\right|^2 + \lambda \sum_{i=1}^{k} S(a_i^{(j)}) \\ \text{subject to} & ||\phi_i||^2 \leq C, \forall i = 1, ..., k \end{array}$$

Olshausen and Field 1996

Identifiability: Scale

- One can remove the scale degree of freedom either in constraint form

$$\begin{array}{rc} \text{minimize} & \|As - x\|_2^2 + \lambda \|s\|_1 \\ \text{s.t.} & A_j^T A_j \leq 1 \, \forall j \end{array}$$

- Or Lagrangian form:

$$J(A, s) = \|As - x\|_2^2 + \lambda \|s\|_1 + \gamma \|A\|_2^2$$

where $\|A\|_2^2 := trace(A^T A)$ is simply the sum of squares of the entries of A (squared Frobenius norm)

Nonlinear Optimization

$$J(A, s) = \|As - x\|_2^2 + \lambda\|s\|_1 + \gamma\|A\|_2^2$$

- Bi-variate cost: not jointly convex, not differentiable
- But $J(A,s)$ is convex in A fixed s, and convex in s fixed A.

Alternating Optimization

- Initialize A randomly
- Repeat until convergence
  - Find the s that minimizes $J(A,s)$ for the A found in the previous step (LASSO)
  - Solve for the A that minimizes $J(A,s)$ for the s found in the previous step (Ridge Regression -> SVD)

Smoothing

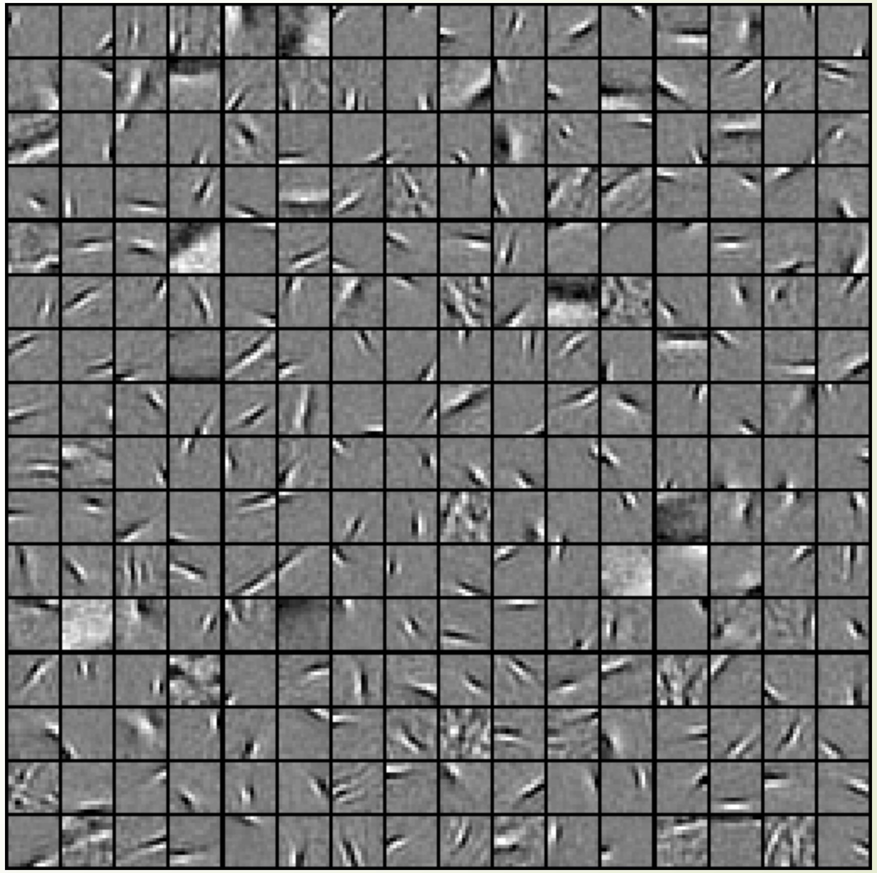
- So our final objective function:

$$J(A, s) = \|As - x\|_2^2 + \lambda\sqrt{s^2 + \epsilon} + \gamma\|A\|_2^2$$

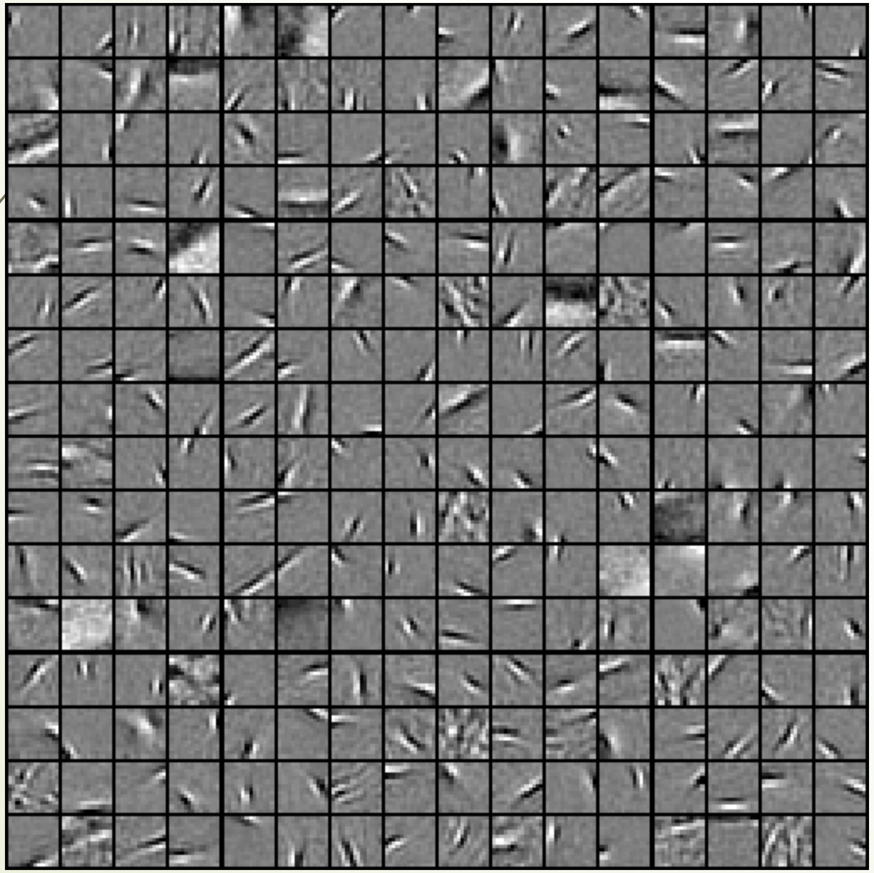
  - where $\sqrt{s^2 + \epsilon}$ is shorthand for $\sum_k \sqrt{s_k^2 + \epsilon}$
  - the third term $\|A\|_2^2$ is simply the sum of squares of the entries of A (squared Frobenius norm)
- Then you have a smooth objective function, restricted convex in A and s
- Gradient descent such as BP algorithm can be applied here

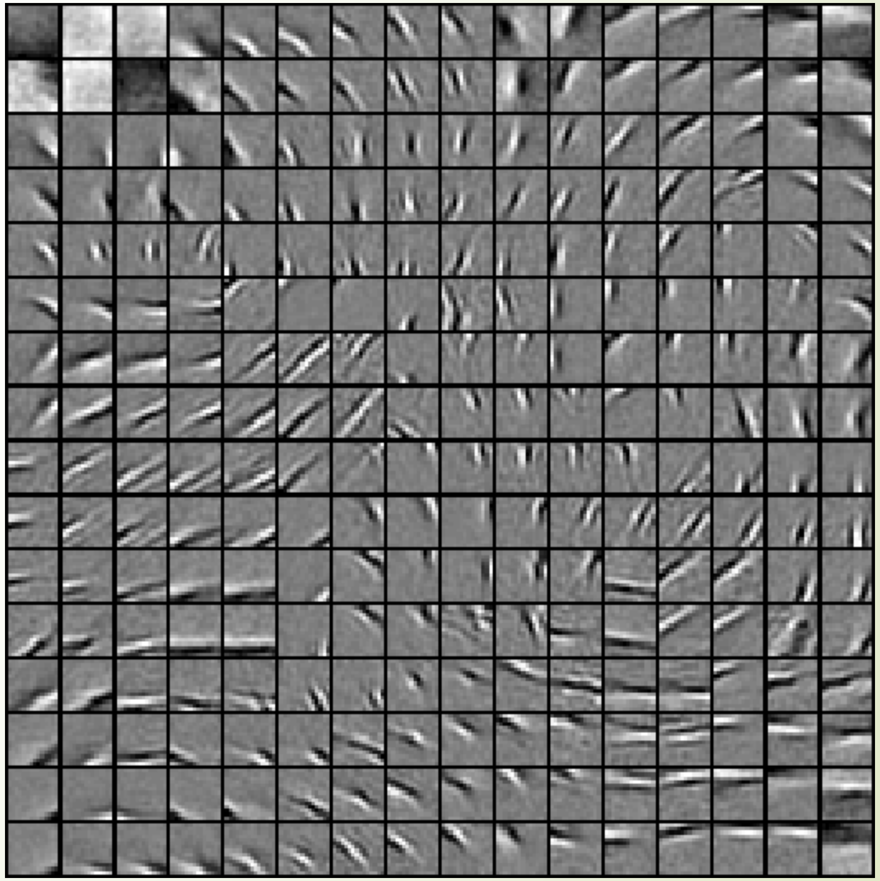
Dictionary Learned from natural image patches

- http://ufldl.stanford.edu/wiki/index.php/Exercise:Sparse_Coding
- http://ufldl.stanford.edu/wiki/resources/sparse_coding_exercise.zip



Adjacent features should be similar?





Topographic Sparse Coding: Group LASSO

- Group adjacent features in group LASSO norm

$$J(A,s) = \|As - x\|_2^2 + \lambda \sum_{\text{all groups } g} \sqrt{\left(\sum_{\text{all } s\in g} s^2\right) + \epsilon} + \gamma\|A\|_2^2$$

- Example: 3-by-3 neighborhood as ‘adjacency’

$$\sqrt{s_{1,1}^2 + \epsilon} \Rightarrow \sqrt{s_{1,1}^2 + s_{1,2}^2 + s_{1,3}^2 + s_{2,1}^2 + s_{2,2}^2 + s_{3,2}^2 + s_{3,1}^2 + s_{3,2}^2 + s_{3,3}^2 + \epsilon}$$

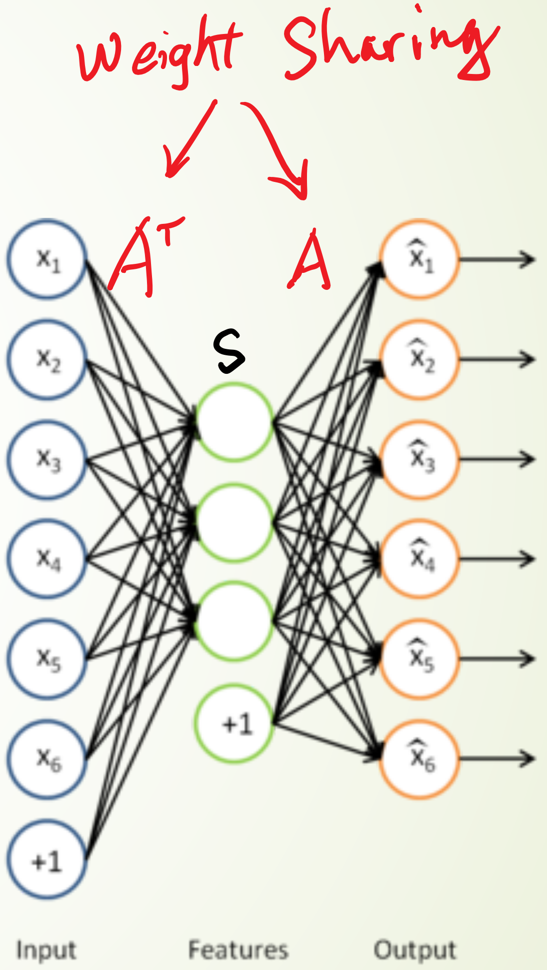
A Neural Network Interpretation: Sparse Autoencoder

- Single-hidden layer NN, to learn features A to reconstruct signals x
- Encoding: $s = A^T x$
- Decoding: As
- *Sample torch codes: https://github.com/torch/tutorials/blob/master/3_unsupervised/2_models.lua*

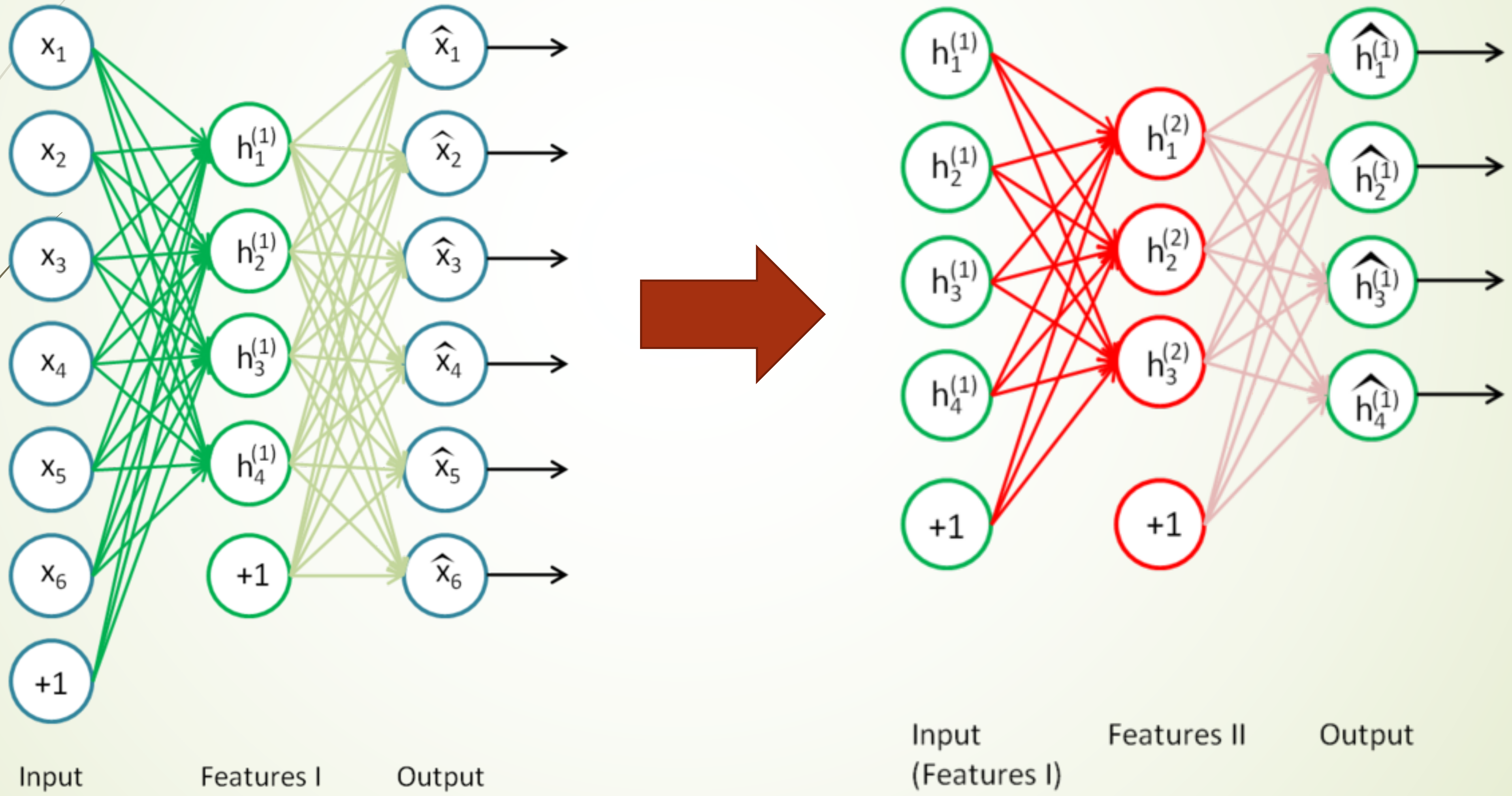
```
-- encoder
encoder = nn.Sequential()
encoder:add(nn.Linear(inputSize,outputSize))
encoder:add(nn.Tanh())
encoder:add(nn.Diag(outputSize))

-- decoder
decoder = nn.Sequential()
decoder:add(nn.Linear(outputSize, inputSize))

-- tied weights
if params.tied and not params.hessian then
-- impose weight sharing
decoder:get(1).weight = encoder:get(1).weight:t()
decoder:get(1).gradWeight = encoder:get(1).gradWeight:t()
```



Deep Networks



Other structures?
-- Tight frame

- Encoding: $A^T x$ (easy)
- Decoding: As

$$J(A,s) = \|As - x\|_2^2 + \lambda\sqrt{s^2+\epsilon} + \gamma\|A\|_2^2$$

- That A is a basis requires: $AA^T = A^T A = I$
- That A is a tight frame satisfies: $AA^T = I \Leftrightarrow \|x\|_2 = \|A^T x\|_2, \forall x$
- Replace reconstruction error to representation error:

$$J(A,s) = \|s - A^T x\|_2^2 + \lambda\sqrt{s^2+\epsilon} + \gamma\|A\|_2^2$$

  - LASSO (fixed A, find s) is simply a soft thresholding
  - L0 regularization leads to hard thresholding

When does Dictionary Learning work?

- Daniel Spielman, Huan Wang, and John Wright, **Exact Recovery of Sparsely-Used Dictionaries,** arXiv:1206.5882
- Agarwal, Anandkumar, Jain, Netrapalli, **Learning Sparsely Used Overcomplete Dictionaries via Alternating Minimization**, arXiv:1310.7991
- Sanjeev Arora, Rong Ge, and Ankur Moistra, **New Algorithms for Learning Incoherent and Overcomplete Dictionaries,** arXiv:1308.6273, 2013
- Barak, Kelner, and Steurer, **Dictionary Learning and Tensor Decomposition via the Sum-of-Squares Method,** http://arxiv.org/abs/1407.1543, 2014