LMSER self-organizing net

Shikui Tu

Department of Computer Science and Engineering, Shanghai Jiao Tong University

2018-05-31

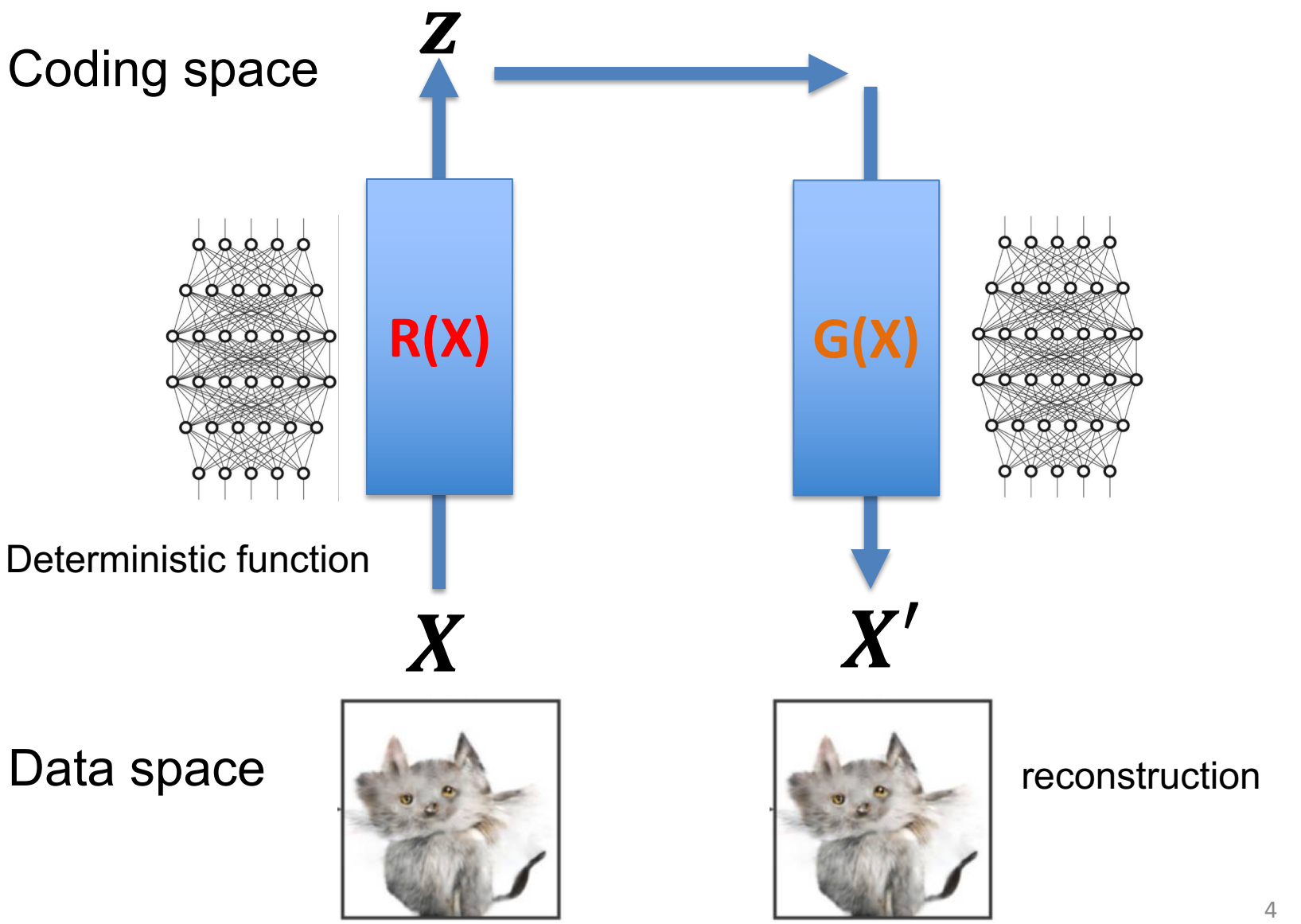
Outline

- Some history and recent advances about the Encoder-Decoder (bi-directional) framework learning
  - Autoencoder (AE) (1988)
  - LMSER net (1991,1993)
  - U-net (2015)
- LMSER network architecture
- LMSER perceptron
- LMSER learning
- LMSER applications

History and recent advances

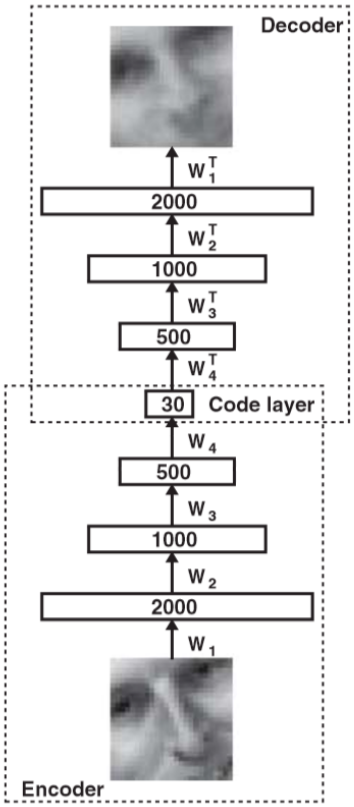
- 早期双向深度学习始于1988年的自编码 auto-encoder
- 多层LMSER自组织学习（Xu1991）是其发展，同时具备自编码和监督学习
- 近年出现且相当火的的U-net上与LMSER结构类似
- Stacked RBMs （Hinton, et al 2007)，结构上有一部分共同点，学习算法的主要部分也类似
- 多层LMSER （Xu1991）曾就注意、回忆、心象等设想

Autoencoder (AE)



早期双向深度学习：auto-encoder

Stacked RBMs
Hinton, et al (2006)

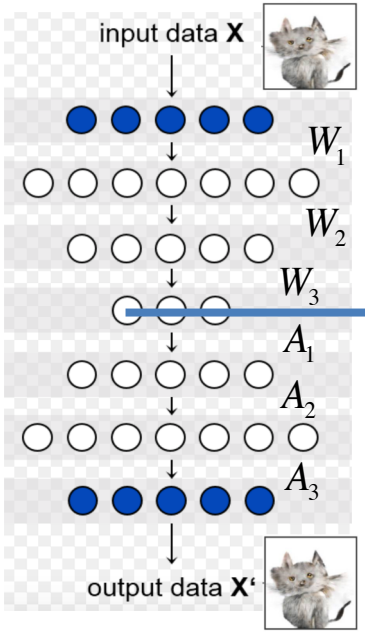


强制了参数对称性

$W_j = A_{ji}^T$

Early auto-associative networks

$W_j \neq A_{ji}^T$

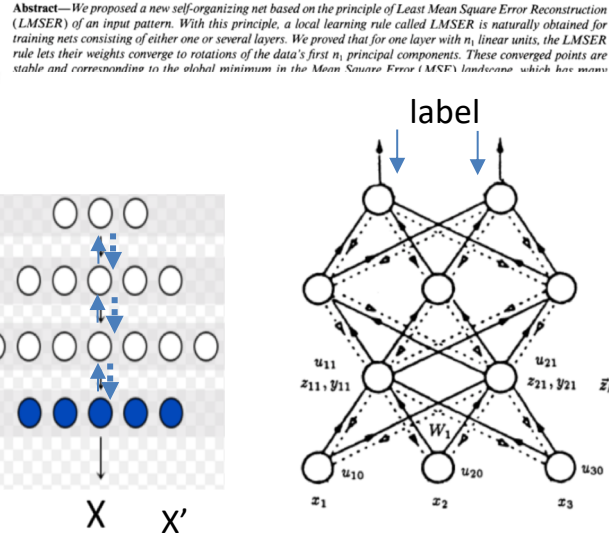


Bourlard, H., Kamp, Y.
Auto-association by multilayer perceptrons … .
Biological cybernetics 59(4-5), 291-294 (1988)

Xu, *Proc. IJCNN 91,* Singapore, pp. 2362-2373
Neural Networks, vol. 6, pp. 627-648, 1993

Least Mean Square Error Reconstruction Principle for Self-Organizing Neural-Nets

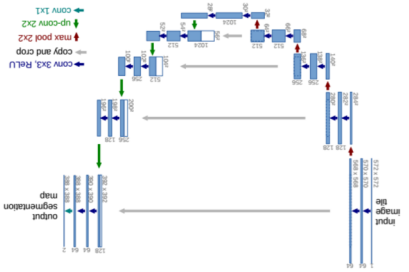
多层LMSER自组织学习

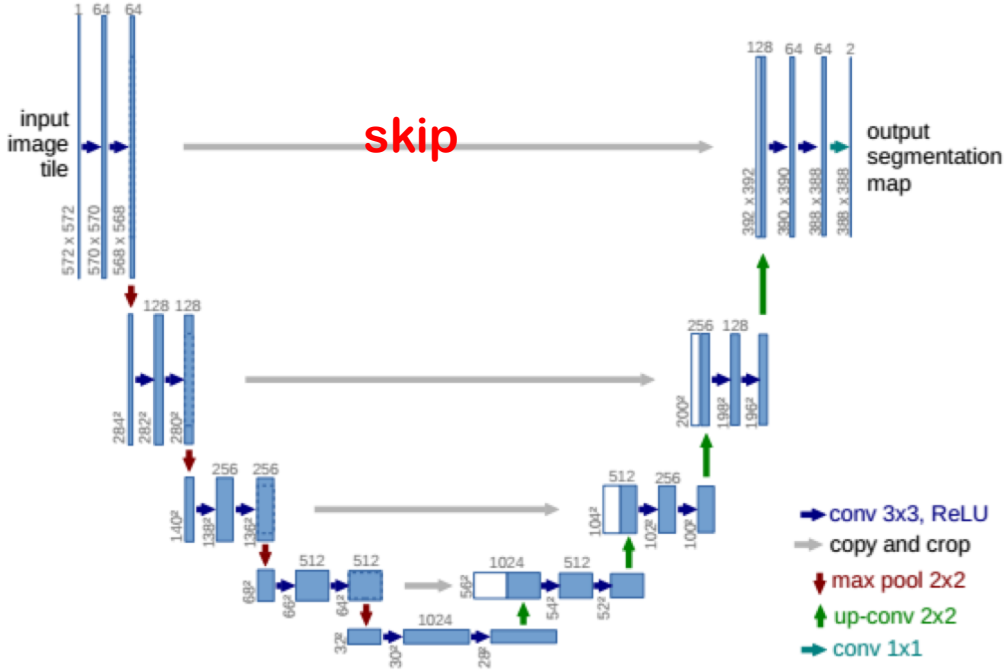


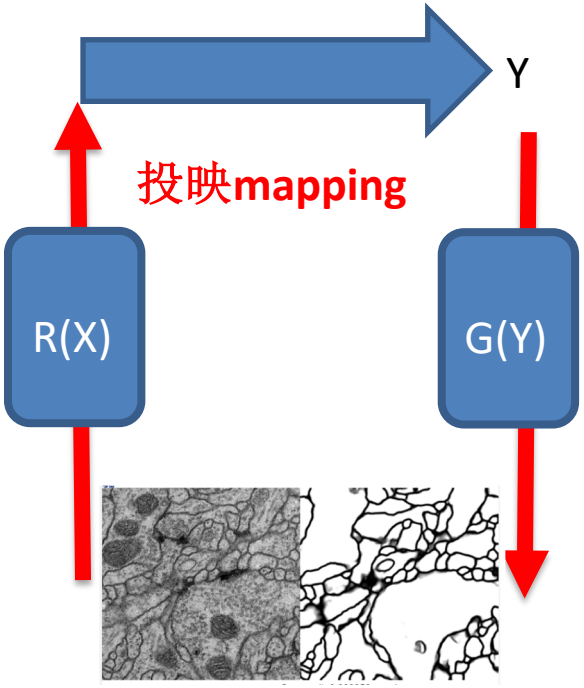
强制了上下两半部的参数对称性
和神经元强度的对称性 $W_j = A_{ji}^T$

U-Net: adding skip connections to Encoder-Decoder

U-Net: Convolutional Networks for Biomedical Image Segmentation
https://arxiv.org › cs ▾
by O Ronneberger - 2015 - Cited by 1719 - Related articles
May 18, 2015 - Moreover, the **network** is fast. **Segmentation** of a 512x512 **image** takes less than a second on a recent GPU. The full implementation (based on ...







Outline

- Some history and recent advances
- **LMSER network architecture**
- LMSER perceptron
- LMSER learning
- LMSER applications

Xu, *Proc. IJCNN 91,* Singapore, pp. 2362-2373
Neural Networks, vol. 6, pp. 627-648, 1993

Neural Networks, Vol. 6, pp. 627–648, 1993
Printed in the USA.

0893-6080/93 $6.00 + .00

ORIGINAL CONTRIBUTION

Least Mean Square Error Reconstruction Principle for Self-Organizing Neural-Nets

LEI XU

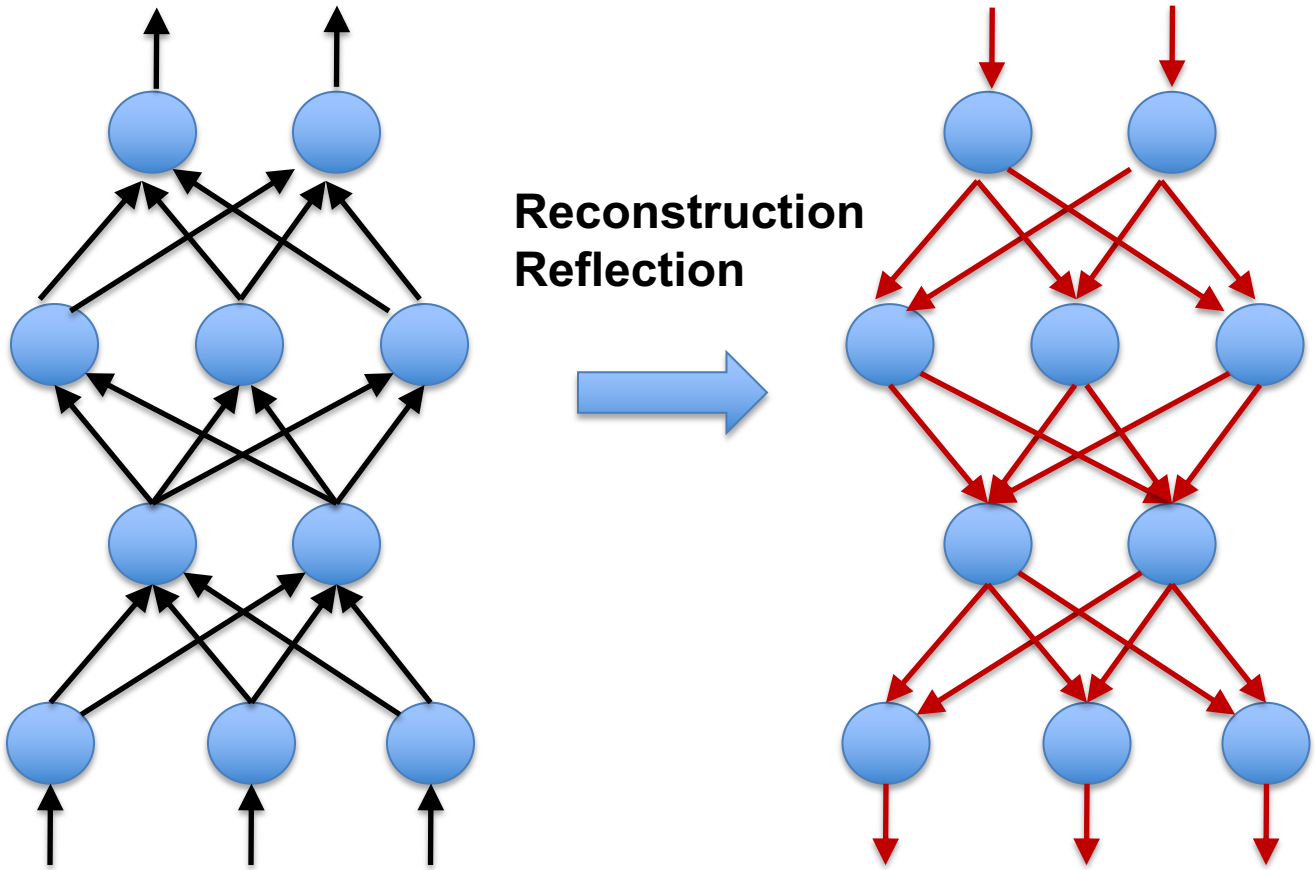
Peking University and Harvard University

(*Received* 29 *July* 1991; *revised and accepted* 16 *October* 1992)

Abstract—*We proposed a new self-organizing net based on the principle of Least Mean Square Error Reconstruction (LMSER) of an input pattern. With this principle, a local learning rule called LMSER is naturally obtained for training nets consisting of either one or several layers. We proved that for one layer with n_1 linear units, the LMSER rule lets their weights converge to rotations of the data's first n_1 principal components. These converged points are stable and corresponding to the global minimum in the Mean Square Error (MSE) landscape, which has many saddles but no local minimum. The results indirectly provided a picture about LMSER's global convergence, which is also suitable for Oja rule since we proved that the evolution direction of the Oja rule has a positive projection on that of LMSER. We have also revealed an interesting fact that slight modifications of the LMSER rule (also the Oja rule) can perform the true Principal Component Analysis (PCA) without externally designing for building asymmetrical circuits required by previous studies.*

Keywords—Self-organization, Least MSE reconstruction, PCA nets, Convergence analysis, Symmetry breaking.

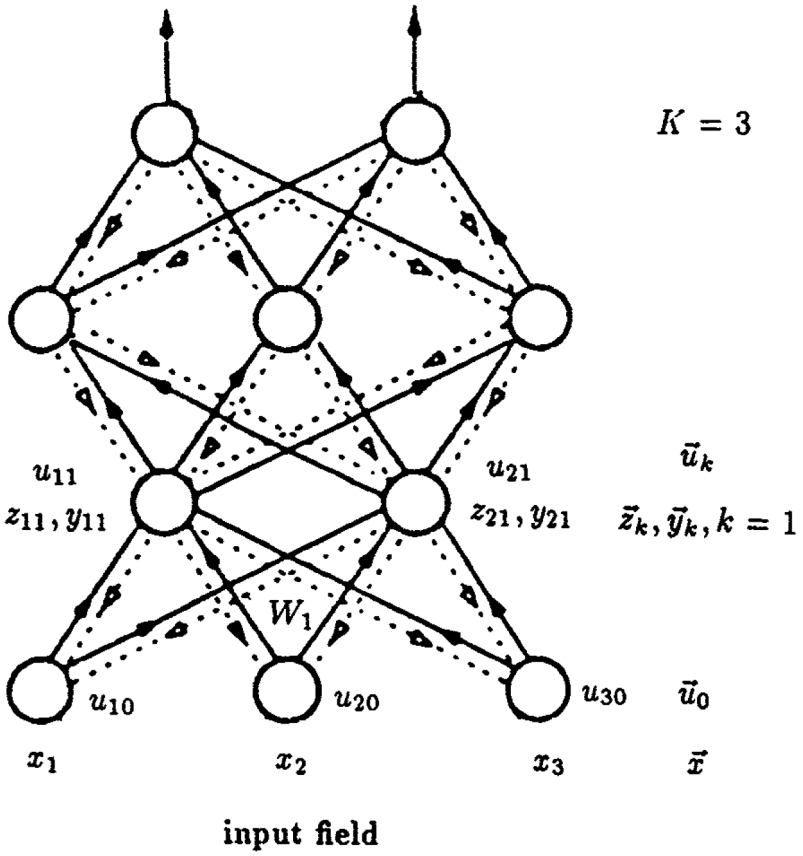
The conventional feedforward architecture



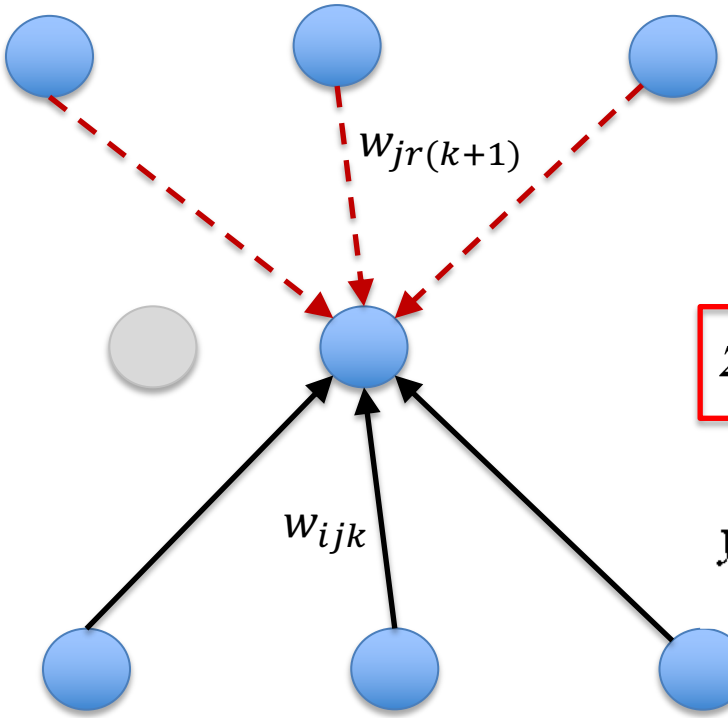
The LMSER net architecture

- The connections are **bidirectional**.
- The weights are **symmetrical**.
- Except for the bottom and the top layers, the activity of each unit is activated by two types of signals:
  - **Bottom-up** signal
  - **Top-down** signal

上一层神经元返回



The neuron of LMSER net



$$u_{jk} = \sum_{r=1}^{n_{k+1}} w_{jr(k+1)} z_{r(k+1)}$$

$$z_{jk} = s(y_{jk} + u_{jk})$$

$$y_{jk} = \sum_{i=1}^{n_{k-1}} w_{ijk} z_{i(k-1)}$$

z_{jk} is the activation signal of j-th neuron at the k-th layer.

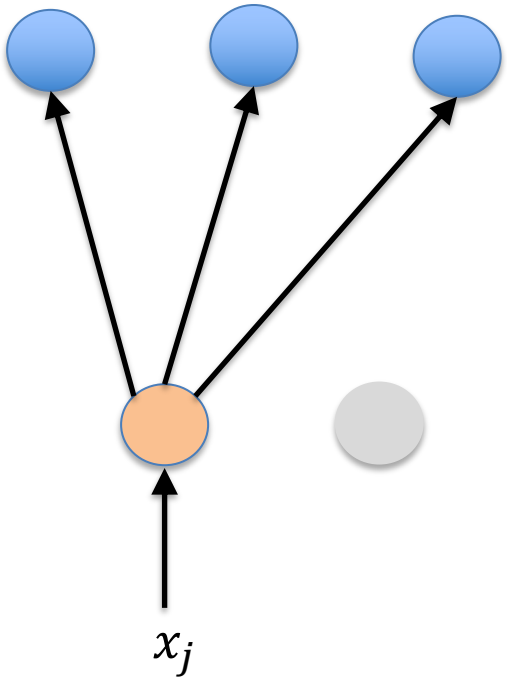
The input layer

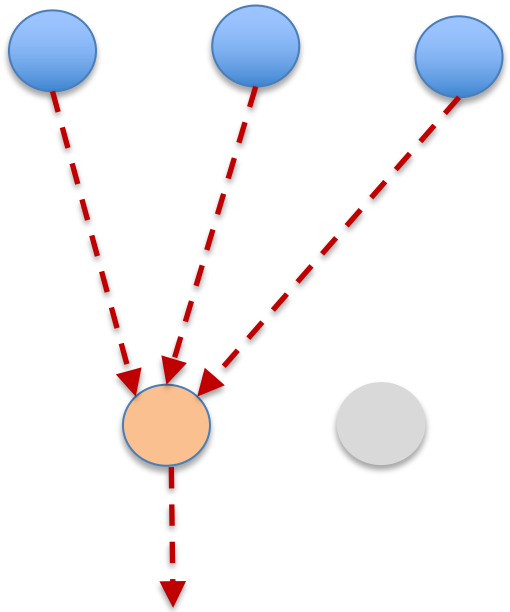
Pass the input data to the 1st layer

Receive the top-down signals from the 1st layer

Layer $k = 1$

Layer $k = 0$
(Input layer)





reconstruction

$$u_{j0} = \sum_{r=1}^{n_1} w_{rj1} z_{r1}$$

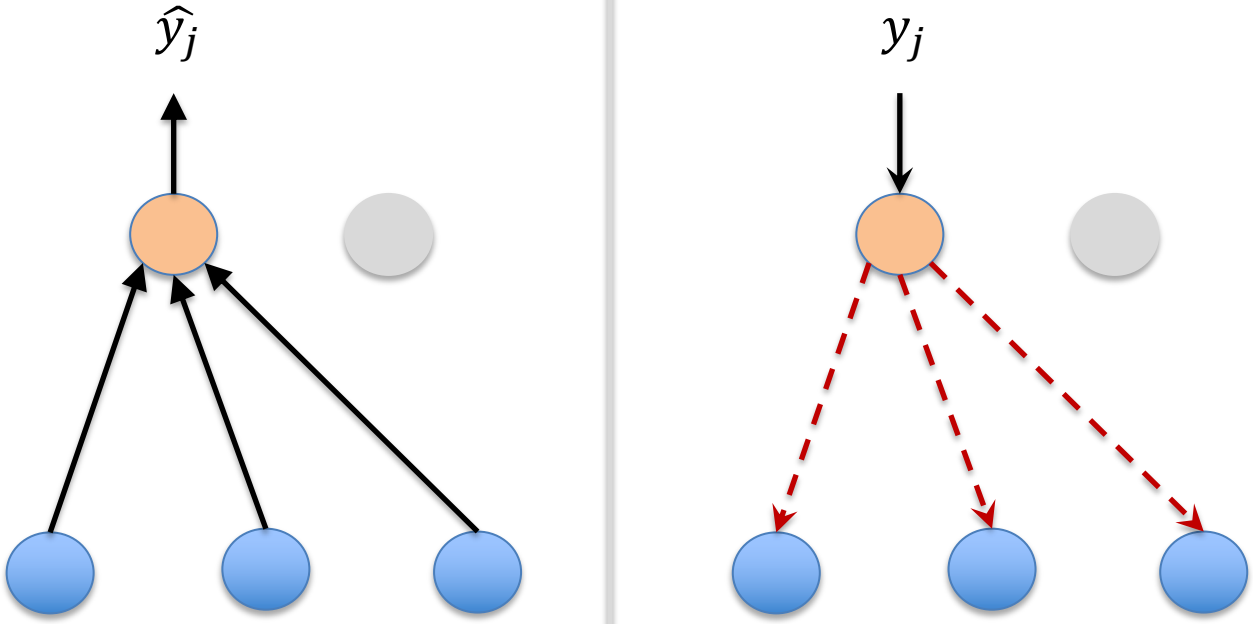
The top layer (supervised)

Predict the label

Receive the label information and pass it down

Top layer $k = K$

Layer $k = K - 1$



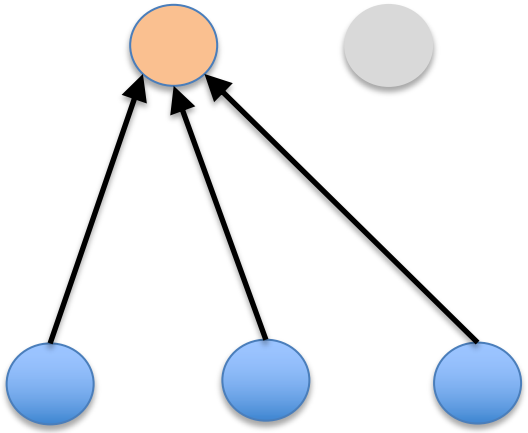
The top layer (unsupervised)

Hidden coding space

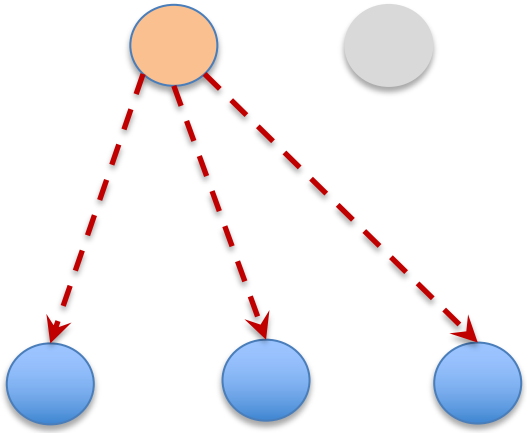
$$z_{jk} = s(y_{jk}), \quad y_{jk} = \sum_{i=1}^{n_{k-1}} w_{ijk} z_{i(k-1)}$$

Top layer $k = K$

Layer $k = K - 1$



Emit the hidden codes back down to the networks



Outline

- Some history and recent advances
- LMSER network architecture
- **LMSER perceptron**
- LMSER learning
- LMSER applications

LMSER net works in two phases

- **The perceptron phase**
  - A stable dynamic process to pass the signals up or down.

$$z_{jk} = s(y_{jk} + u_{jk}) \quad \text{or} \quad \tau \frac{dz_{jk}}{dt} = -z_{jk} + s(y_{jk} + u_{jk})$$

$$y_{jk} = \sum_{i=1}^{n_{k-1}} w_{ijk} z_{i(k-1)}, \quad \text{and} \quad u_{jk} = \sum_{r=1}^{n_{k+1}} w_{jr(k+1)} z_{r(k+1)}$$

- **The learning phase**
  - Adjusting the weights based on the least mean square error principle

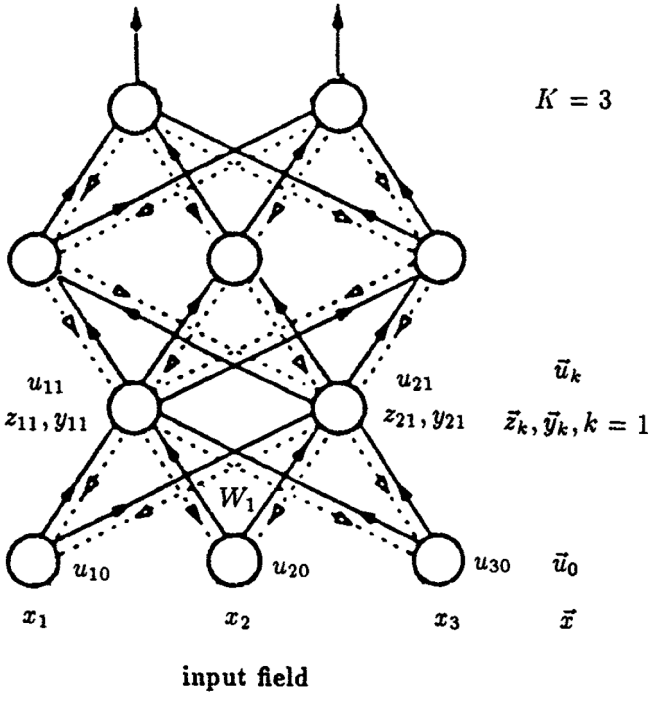
The perceptron phase

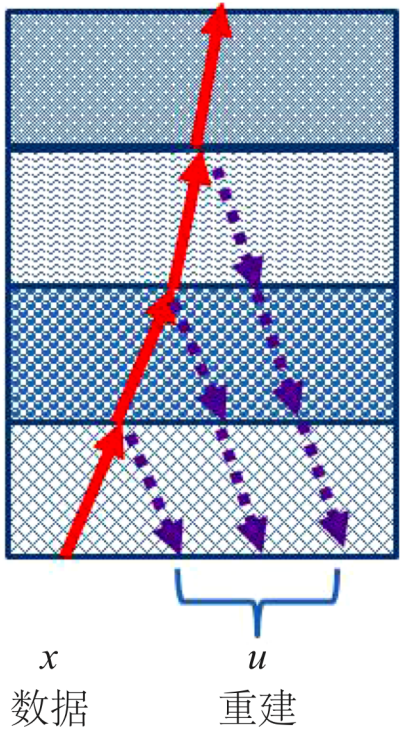
Bottom-up signals

$$z_{jk} = s(y_{jk} + u_{jk}) \quad \text{or} \quad \tau \frac{dz_{jk}}{dt} = -z_{jk} + s(y_{jk} + u_{jk})$$

$$y_{jk} = \sum_{i=1}^{n_{k-1}} w_{ijk} z_{i(k-1)}, \quad \text{and} \quad u_{jk} = \sum_{r=1}^{n_{k+1}} w_{jr(k+1)} z_{r(k+1)}$$

Top-down signals





Keep on changing the values of the signals, until they do not change.

Is it **stable**?

The perceptron phase is stable

THEOREM 1. *When s(.) is a sigmoid-type function, the activities of all the units will finally converge to a set of specific values such that the minimum of the following global Lyapunov or energy function has been reached*

$$J_{\text{net}} = -\frac{1}{2}\sum_{k=2}^{K}\sum_{i=1}^{n_{k-1}}\sum_{j=1}^{n_k} w_{ijk} z_{i(k-1)} z_{jk} + \sum_{k=1}^{K}\sum_{j=1}^{n_k}\int_0^{z_{jk}} s^{-1}(z)\,dz + \sum_{j=1}^{n_1}\sum_{i=0}^{n_0} z_{i1} x_i w_{ij1}. \quad (3)$$

After it reaches the stable state

- If $u_0 - x = 0$:
  - Complete reconstruction
  - No learning will take place.
  - The activities $\{z_{jk}, k > 0\}$ can be regarded as perceptron of the present input pattern x.
- If $u_0 - x \neq 0$:
  - Partial reconstruction
  - u_0 is a recalled pattern, or partial perceptron of x.
  - Learning is triggered to reduce $u_0 - x$.

Outline

- Some history and recent advances
- LMSER network architecture
- LMSER perceptron
- **LMSER learning**
  - Stochastic gradient descent
  - A special case: One layer
  - Analysis on PSA
- LMSER applications

The learning phase

- The task is to adjust all the weights in order to reduce the reconstruction error:

$$\underset{\{W_k, k=1,2\cdots K\}}{Min} \; J \;, J = \frac{1}{2} E(\| \vec{x} - \vec{u}_0 \|^2) = \frac{1}{2} E(\| \vec{x} - W_1' \vec{z}_1 \|^2)$$

- Gradient descent
  - Batch-way
  - Stochastic (adaptive-way)

Stochastic gradient descent

Assume the input x comes randomly and stationary from a distribution:

$$J^0 = \frac{1}{2} \| \vec{x} - W_1 \vec{z}_1 \|^2 = \frac{1}{2} \sum_{i=1}^{n_0} x_i^{\ 2} - \sum_{i=1}^{n_0} \sum_{j=1}^{n_1} w_{ji1} z_{j1} x_i + \frac{1}{2} \sum_{i=1}^{n_0} \left(\sum_{j=1}^{n_1} w_{ij1} z_{j1} \right)^2$$

If we take expectation on the stochastic gradient:

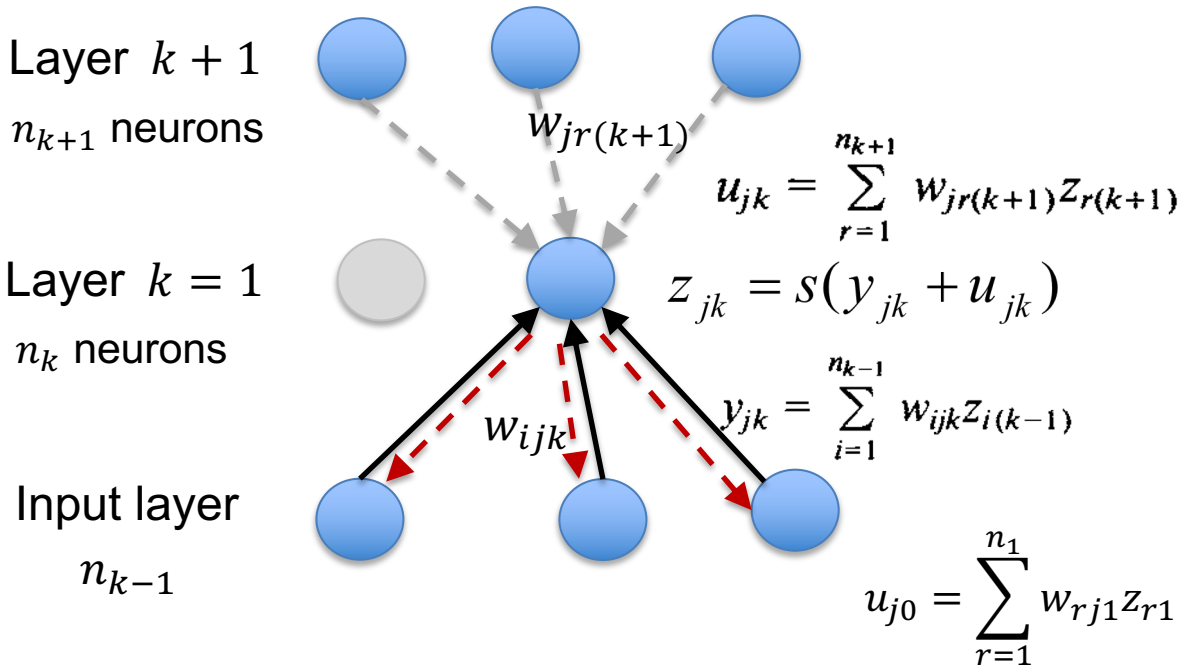
$$\frac{\partial J^0}{\partial w_{ijk}} = \frac{\partial \frac{1}{2} E \| \vec{x} - \vec{u}_0 \|^2}{\partial w_{ijk}} = \frac{\partial J}{\partial w_{ijk}}$$

Therefore, on the average, it also minimize in the gradient descent way.

Compute the gradient for k=1

$$J^0 = \frac{1}{2}\| \vec{x} - W_1\vec{z}_1 \|^2 = \frac{1}{2}\sum_{i=1}^{n_0} x_i^{\,2} - \sum_{i=1}^{n_0}\sum_{j=1}^{n_1} w_{ji1} z_{j1} x_i + \frac{1}{2}\sum_{i=1}^{n_0}\left(\sum_{j=1}^{n_1} w_{ij1} z_{j1}\right)^2$$

Noticing that $z_{jk} = s(y_{jk} + u_{jk})$ and u_{jk} is irrelevant to any weights below the k-th layer, we try to decouple the problem of eqn (5d) layer by layer from the bottom to the top of the net.



$$\tau^w \frac{\partial w_{pq1}}{\partial t} = -\frac{\partial J}{\partial w_{pq1}}$$

$$= -\sum_{i=1}^{n_1} \frac{\partial J^0}{\partial z_{i1}} \frac{\partial z_{i1}}{\partial w_{pq1}} \quad (1)$$

$$-\sum_{j=1}^{n_0} \frac{\partial J^0}{\partial u_{j0}} \frac{\partial u_{j0}}{\partial w_{pq1}} \quad (2)$$

$$J^0 = \frac{1}{2} \| \vec{x} - W_1 \vec{z}_1 \|^2 = \frac{1}{2} \sum_{i=1}^{n_0} x_i^{\,2} - \sum_{i=1}^{n_0} \sum_{j=1}^{n_1} w_{ji1} z_{j1} x_i + \frac{1}{2} \sum_{i=1}^{n_0} \left(\sum_{j=1}^{n_1} w_{ij1} z_{j1} \right)^2$$

The first term:

1

$$\varepsilon_{i1} = -\frac{\partial J^0}{\partial z_{i1}} = \sum_{j=1}^{n_0} w_{ij1} x_j - \sum_{j=1}^{n_0} w_{ij1} \left(\sum_{r=1}^{n_1} w_{rj1} z_{r1} \right) = y_{i1} - y_{i1}^r,$$

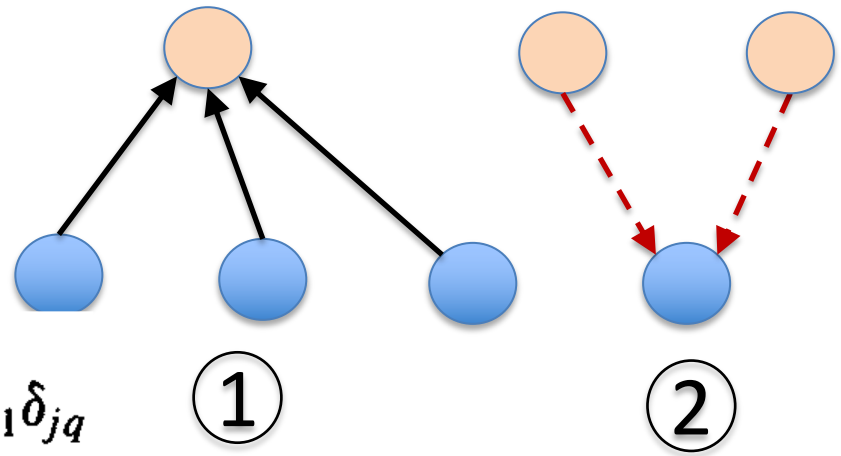
$$\frac{\partial z_{i1}}{\partial w_{pq1}} = \frac{\partial s(y_{i1} + u_{i1})}{\partial w_{pq1}} = s'_{i1} \frac{\partial}{\partial w_{pq1}} \sum_{j=1}^{n_0} w_{ij1} x_j = s'_{i1} \delta_{ip} x_q$$

The second term:

2

$$\varepsilon_{j0} = -\frac{\partial J^0}{\partial u_{j0}} = x_j - u_{j0}$$

$$\frac{\partial u_{j0}}{\partial w_{pq1}} = \frac{\partial}{\partial w_{pq1}} \sum_{r=1}^{n_1} w_{rj1} z_{r1} = z_{p1} \delta_{jq}$$



$$s'_{ik} = s'(y_{ik} + u_{ik}) \qquad \delta_{ip} = 1 \text{ if } i = q;\ \delta_{ip} = 0 \text{ if } i \neq q$$

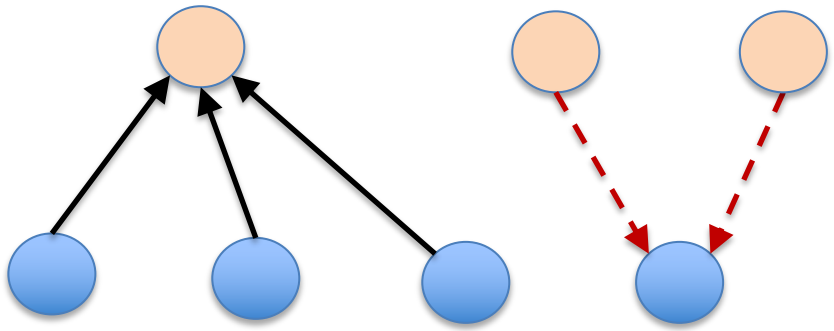
For the layers $k = 2, \ldots, K$

$$\tau^w \frac{\partial w_{pqk}}{\partial t} = -\frac{\partial J}{\partial w_{pqk}} = \sum_{i=1}^{n_k} \varepsilon_{ik} \frac{\partial z_{ik}}{\partial w_{pqk}} + \sum_{i=1}^{n_{(k-1)}} \varepsilon_{i(k-1)} \frac{\partial z_{i(k-1)}}{\partial w_{pqk}}$$

$$\varepsilon_{ik} = -\frac{\partial J^0}{\partial z_{ik}} = -\sum_{j=1}^{n_{(k-1)}} \frac{\partial J^0}{\partial z_{j(k-1)}} \frac{\partial z_{j(k-1)}}{\partial z_{ik}} = \sum_{j=1}^{n_{(k-1)}} \varepsilon_{j(k-1)} \frac{\partial z_{j(k-1)}}{\partial z_{ik}} = \sum_{j=1}^{n_{(k-1)}} \varepsilon_{j(k-1)} w_{ijk}$$

$$\frac{\partial z_{j(k-1)}}{\partial z_{ik}} = \frac{\partial s(y_{j(k-1)} + u_{j(k-1)})}{\partial z_{ik}} = s'_{j(k-1)} \frac{\partial u_{j(k-1)}}{\partial z_{ik}}$$

$$= s'_{j(k-1)} \frac{\partial \sum_{r=1}^{n_k} w_{rjk} z_{rk}}{\partial z_{ik}} = s'_{j(k-1)} w_{ijk}$$

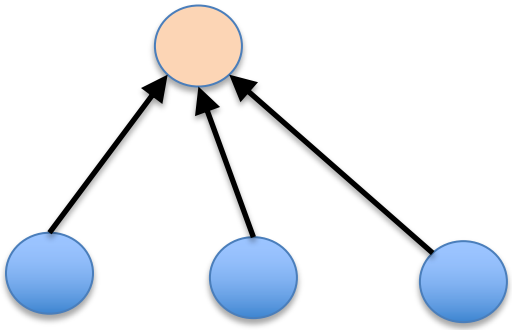


For the layers $k = 2, \ldots, K$

$$\tau^w \frac{\partial w_{pqk}}{\partial t} = -\frac{\partial J}{\partial w_{pqk}} = \sum_{i=1}^{n_k} \varepsilon_{ik} \frac{\partial z_{ik}}{\partial w_{pqk}} + \sum_{i=1}^{n_{(k-1)}} \varepsilon_{i(k-1)} \frac{\partial z_{i(k-1)}}{\partial w_{pqk}}$$

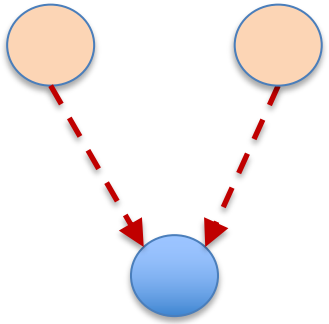
$$\frac{\partial z_{ik}}{\partial w_{pqk}} = s'_{ik} \frac{\partial y_{ik}}{\partial w_{pqk}} = s'_{ik} \frac{\partial}{\partial w_{pqk}} \sum_{j=1}^{n_{k-1}} w_{ijk} z_{j(k-1)}$$

$$= s'_{ik} \delta_{ip} z_{q(k-1)} + s'_{ik} \sum_{j=1}^{n_{k-1}} w_{ijk} s'_{j(k-1)} \frac{\partial u_{j(k-1)}}{\partial w_{pqk}}$$



$$\frac{\partial z_{i(k-1)}}{\partial w_{pqk}} = s'_{i(k-1)} \frac{\partial y_{i(k-1)}}{\partial w_{pqk}} = s'_{i(k-1)} \frac{\partial}{\partial w_{pqk}} \sum_{j=1}^{n_k} w_{jik} z_{jk}$$

$$= s'_{i(k-1)} \delta_{iq} z_{pk} + s'_{i(k-1)} \sum_{j=1}^{n_k} w_{jik} s'_{jk} \frac{\partial y_{jk}}{\partial w_{pqk}} .$$



Remove the high-order terms

$$\frac{\partial z_{ik}}{\partial w_{pqk}} = s'_{ik}\delta_{ip}z_{q(k-1)} + s'_{ik}\sum_{j=1}^{n_{k-1}} w_{ijk}s'_{j(k-1)}\frac{\partial u_{j(k-1)}}{\partial w_{pqk}} \approx s'_{ik}\delta_{ip}z_{q(k-1)},$$

$$\frac{\partial z_{i(k-1)}}{\partial w_{pqk}} = s'_{i(k-1)}\delta_{iq}z_{pk} + s'_{i(k-1)}\sum_{j=1}^{n_k} w_{jik}s'_{jk}\frac{\partial y_{jk}}{\partial w_{pqk}} \approx s'_{i(k-1)}\delta_{iq}z_{pk}.$$

The high-order terms:

$$s'(.) = s(1-s) \le \gamma = \tfrac{1}{4} \Rightarrow s'(.)s'(.) \le \gamma^2$$

$$\tau^w \frac{\partial w_{pqk}}{\partial t} = -\frac{\partial J}{\partial w_{pqk}} = \sum_{i=1}^{n_k} \varepsilon_{ik}\frac{\partial z_{ik}}{\partial w_{pqk}} + \sum_{i=1}^{n_{(k-1)}} \varepsilon_{i(k-1)}\frac{\partial z_{i(k-1)}}{\partial w_{pqk}}$$

$$\approx s'_{pk}\varepsilon_{ik}z_{q(k-1)} + s'_{q(k-1)}\varepsilon_{q(k-1)}z_{pk}.$$

The main learning equations

- Update the network weights by stochastic gradient descent:

$$\tau^w \frac{\partial w_{pq1}}{\partial t} = \varepsilon_{q0} z_{p1} + s'_{p1} \varepsilon_{p1} x_q, \quad \text{for} \quad k = 1$$

$$\varepsilon_{i0} = x_i - u_{i0}, \; \varepsilon_{i1} = y_{i1} - y'_{i1},$$

$$\tau^w \frac{\partial w_{pqk}}{\partial t} \approx s'_{q(k-1)} \varepsilon_{q(k-1)} z_{pk} + s'_{pk} \varepsilon_{ik} z_{q(k-1)}, \quad \text{for} \quad k \geq 2.$$

$$\varepsilon_{ik} = \sum_{j=1}^{n_{(k-1)}} \varepsilon_{j(k-1)} w_{ijk},$$

Outline

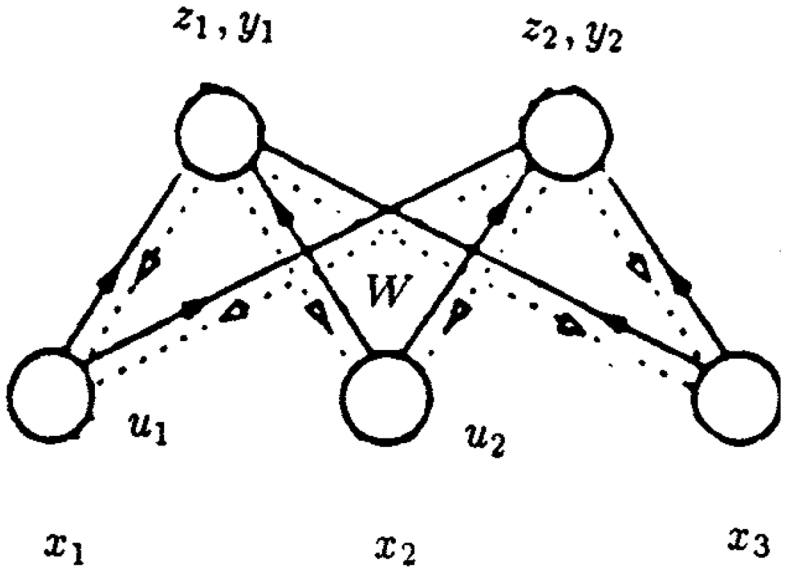
- Some history and recent advances
- LMSER network architecture
- LMSER perceptron
- **LMSER learning**
  - Stochastic gradient descent
  - **A special case: One layer**
  - Analysis on PSA
- LMSER applications

Special case: One layer with linear units

- The objective function becomes:

$$\vec{z} = S(\vec{y}), \quad \vec{y} = W\vec{x}, \quad \vec{u} = W'\vec{z}$$

$$J = \tfrac{1}{2}E(\|\vec{x} - \vec{u}\|^2)$$



Linear unit:

$$s(x) = x.$$

Compute the learning rule

The gradient:

$$\tau^w \frac{dW}{dt} = W\vec{x}\vec{x}^t - W\vec{x}\vec{x}^t W^t W + W\vec{x}\vec{x}^t - WW^t W\vec{x}\vec{x}^t$$

Taking expectations:

$$\tau^w \frac{dW}{dt} = W\Sigma - W\Sigma W^t W + W\Sigma - WW^t W\Sigma.$$

It could be proved that the above learning rule performs automatically **Principal Subspace Analysis** (PSA).

THEOREM 2. *Assume* $n_1 < n_0$ *and* Σ *is nonsingular. Let* $\Phi = [\vec{\phi}_1, \ldots, \vec{\phi}_{n_0}]$ *and* $\Lambda = \mathrm{diag}[\lambda_1, \ldots, \lambda_{n_0}]$ *be the matrices of eigenvectors and eigenvalues of* Σ *respectively, then the critical points of eqn* (10*b*) *are* $W = RDP\Phi^t$, *R is an arbitrary* $n_1 \times n_1$ *rotation matrix, i.e.,* $R^t R = I$. $D = [D_1 | 0]$ *is* $n_1 \times n_0$ *matrix with* D_1 *being an* $n_1 \times n_1$ *diagonal matrix and its diagonal elements only taking value of* $+1, -1, 0$. $P_{n_0 \times n_0}$ *is an arbitrary permutation matrix.*

The landscape of the objective function

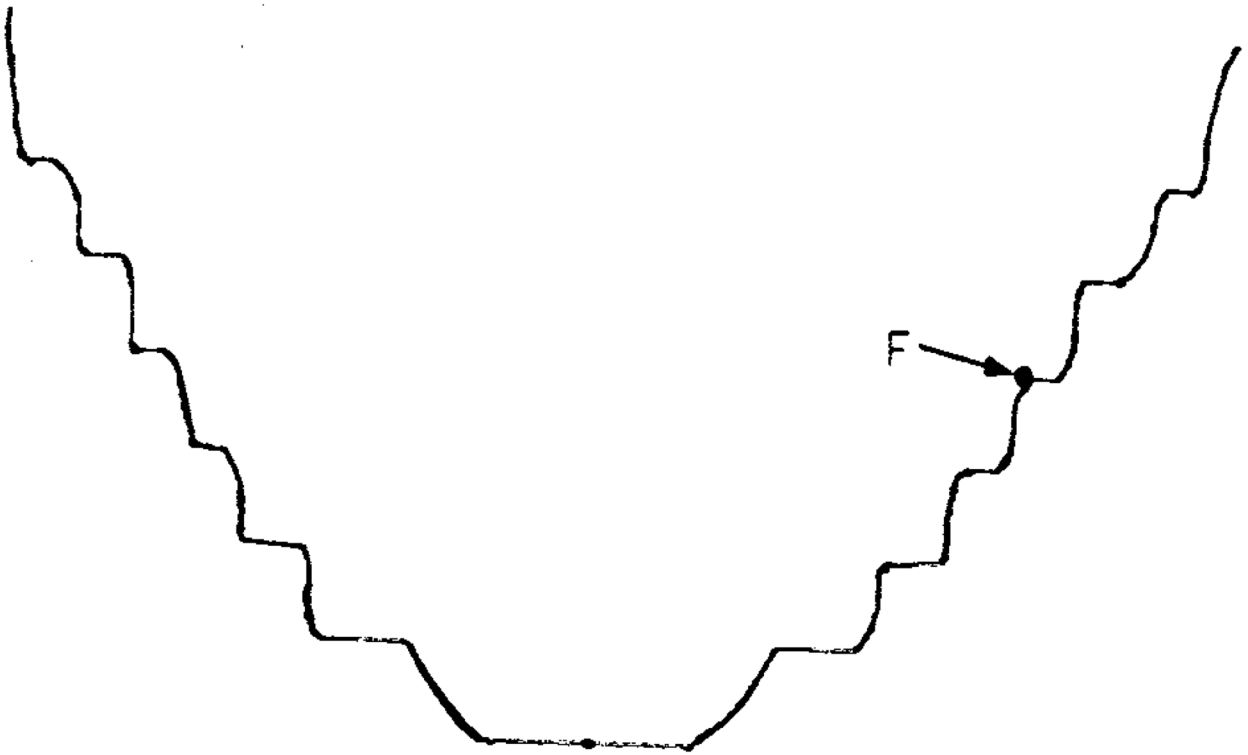


FIGURE 3. The landscape of J. There is only one unique minimum which is the flat bottom, and there are in total $\sum_{l=1}^{n_1} C_{n_0}^{l} - 1$ plateaux which are the saddle points.

Outline

- Some history and recent advances
- LMSER network architecture
- LMSER perceptron
- **LMSER learning**
  - Stochastic gradient descent
  - A special case: One layer
  - **Analysis on PSA**
- LMSER applications

About Oja's subspace rule

- It was shown (in 1991) that Oja's subspace rule cannot be interpreted as a gradient descent search of any energy function.

- It is only known that the rotations of the first n principal eigenvectors are the local stable points. There is no global picture about how it works.

$$\tau^w \frac{dW}{dt} = \vec{y}\vec{x}^t - \vec{y}\vec{y}^t W, \quad \vec{y} = W\vec{x}.$$

New results on Oja's subspace rule

THEOREM 4. *On the average, the evolution direction of Oja's subspace rule in eqn* (12) *has a positive projection on the evolution direction of the LMSER rule given in eqn* (10*a*). *Precisely,* $E(vec[G_0])^t E(vec[G]) = 2E(vec[G_0])^t E(vec[G_0]) > 0$, *where* $G_0 = \vec{y}\vec{x}^t - \vec{y}\vec{y}^t W$, $G = \vec{y}(\vec{x} - \vec{u})^t + (\vec{y} - \vec{y}')\vec{x}^t$, *and "vec" transforms a matrix into a column vector by stacking the columns of the matrix one underneath the other.*

Break the symmetry of PSA

$$\tau^w \frac{dW}{dt} = \vec{y}\vec{x}^t - \vec{y}\vec{u}^t + \vec{y}\vec{x}^t - \vec{y}'\vec{x}^t = W\vec{x}\vec{x}^t - W\vec{x}\vec{x}^t W^t W + W\vec{x}\vec{x}^t - WW^t W\vec{x}\vec{x}^t \quad (10a)$$

- Amplify the output y by different factors

$$\vec{z} = S(\vec{y}) = A_m\vec{y}, \quad A_m = \mathrm{diag}[a_1, \ldots, a_{n_1}]$$

$$a_1 > a_2 > \ldots > a_{n_1} \text{ are all positive.}$$

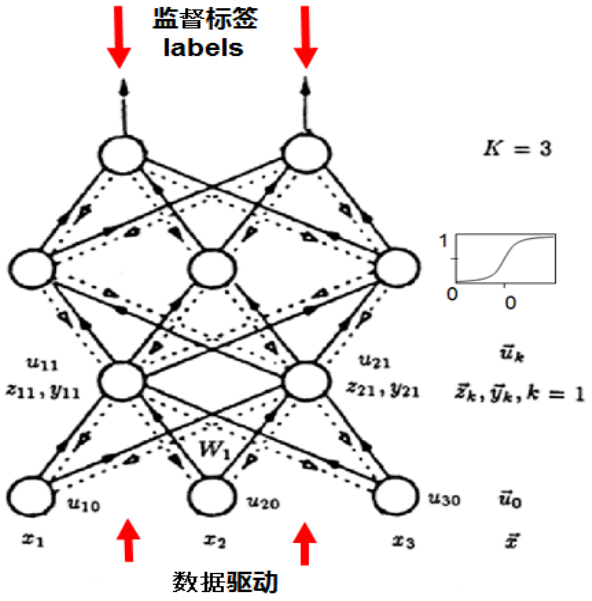
- Include the sigmoid type nonlinearity s(.) in each unit.

$$s(x) = \tanh(\beta x) = \frac{e^{\beta x} - e^{-\beta x}}{e^{\beta x} + e^{-\beta x}}$$

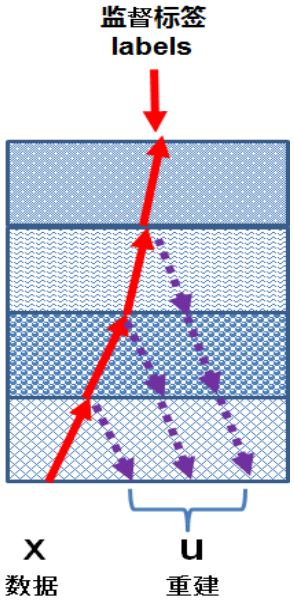
Outline

- Some history and recent advances
- LMSER network architecture
- LMSER perceptron
- LMSER learning
- **LMSER applications**
  - **Reconstruction**
  - **Generation**
  - **Attention**
  - **Recall**

Multilayer LMSER learning



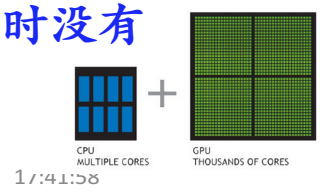
(a)



(b)

数据从输入层传入的逐层自组织学习与从监督层进入的逐层反向传播学习。在逐层反向传播学习中，监督标签由上而下从监督层进入，通过改变各层使得误差不断减小。

当时没有

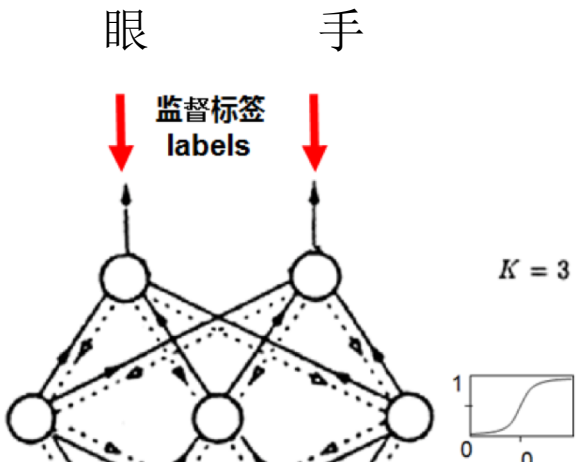


Super-computing ability

Big-data available

遗憾的是，当时计算实验只做了单层

Attention，Recall，Imaginary recall
注意、回忆、心象



(d). In the top layer or the other upper layers, by activating some unit which represents some abstract concept (e.g. the name of one your friend, or a car), through the top-down signals passes downward, on the input field an imaginary picture will be reconstructed. Somewhat like our human, when thinking about one friend name, you usually also his configuration emerging in your brain. This imaginary recall may also help visualizing your thinking.

(e). The activation of some top layer unit will pass down the top-down signals which may make some lower layer units and the first layer units pre-activated if these units represent the features of the pattern represented by that top unit. This will make these units become more sensitive to discover the features what they want to discover. This somewhat like our human's attentional recognition ability.

Xu, 1991, 93 *Proc. IJCNN 91,* Singapore, pp. 2362-2373
Neural Networks, vol. 6, pp. 627-648, 1993

Image reconstruction and classification

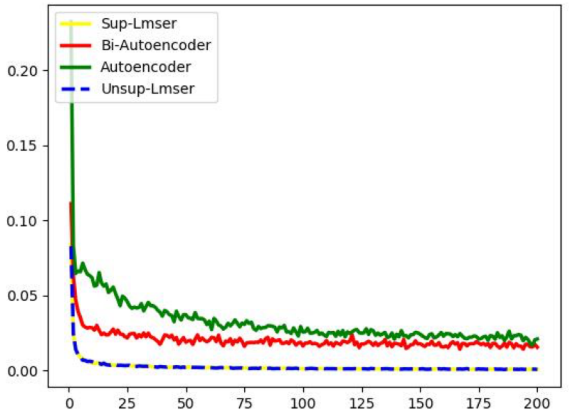
图 4-3. 图像重构效果图

a-d 为迭代 500 次时，各模型对图像的重构; e-h 为迭代 5000 次时，各模型对图像的重构

(a)Autoencoder(b)Bi-Autoencoder(c)Unsup -Lmser(d)Sup-LMSER

(e)Autoencoder(f)Bi-Autoencoder(g)Unsup -Lmser(h)Sup-LMSER

Reconstruction error



模型	重构误差（500）	重构误差（5000）	重构误差（20000）	分类准确率
Sup-LMSER	0.0078	0.0025	0.0007	0.9988
Unsup-LMSER	0.0067	0.0018	0.0006	None
Bi_Autoencoder	0.030	0.021	0.016	0.9988
Autoencoder	0.071	0.036	0.02	None

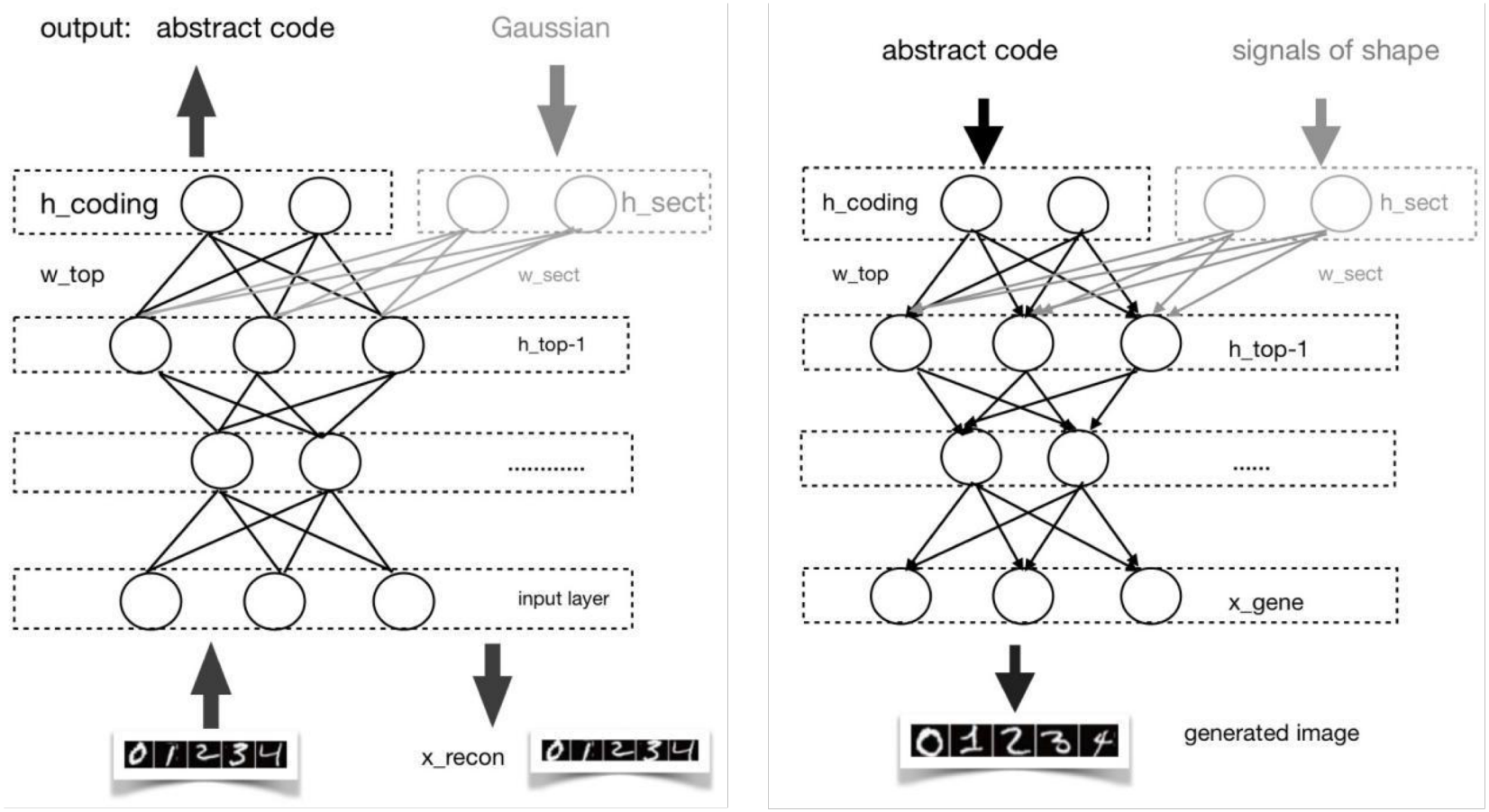
By Wenjing Huang

On Fashion-MNIST

By Wenjing Huang

图 4-5. a-d 分别为 Autoencoder 迭代 1000，5000，10000，20000 次时的重构效果图；
e-h 分别为 Unsup-LMSER 迭代 1000，5000，10000，20000 次时的重构效果图。

LMSER to generate images

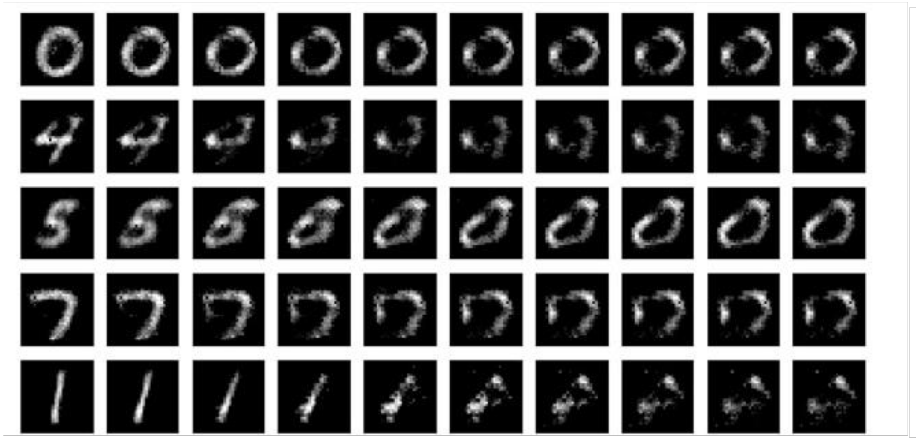


(a) (b)

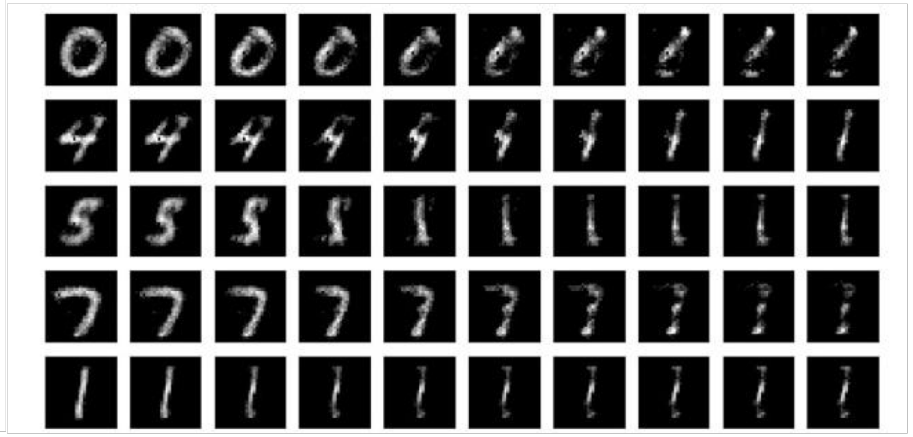
图 5-1. LMSER 生成器的网络结构(a)数据传输阶段(b)图像生成阶段

By Wenjing Huang

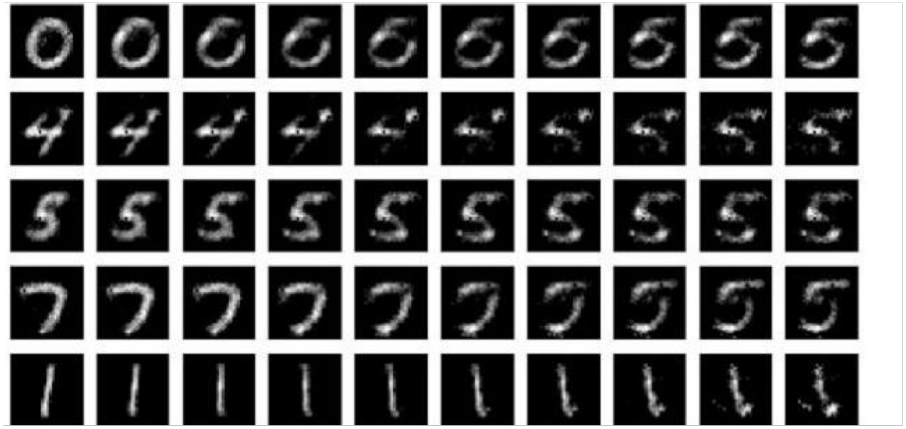
Control the shapes by sup-LMSER



(a)



(b)



(c)

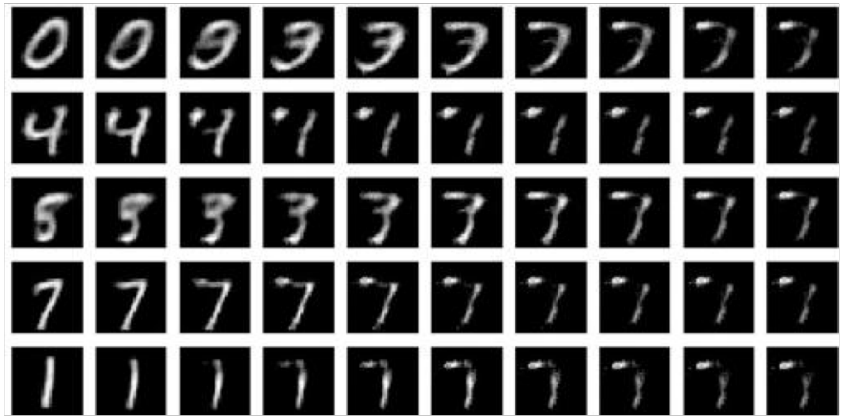
(d)

Sup-LMSER

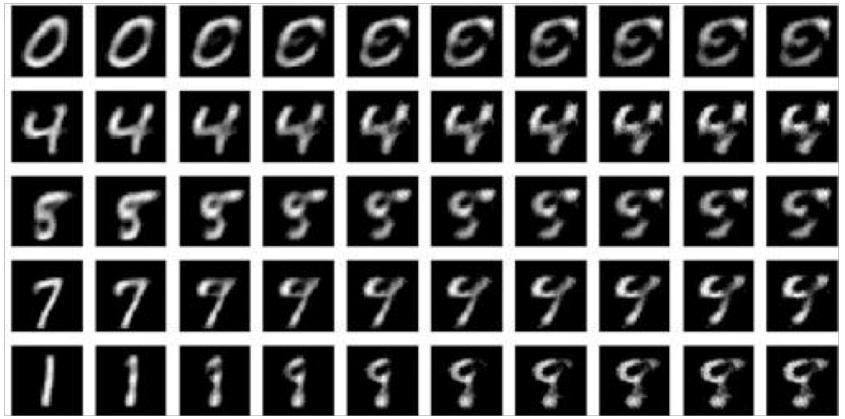
By Wenjing Huang

Control the shapes by unsup-LMSER

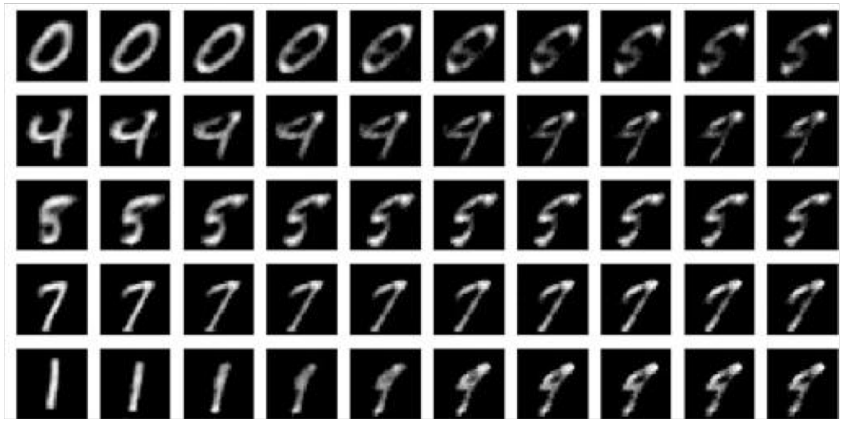
(a)



(b)



(c)



(d)

unsup-LMSER

By Wenjing Huang

Thank you!