Chapter 9. Autoencoders

A way for unsupervised learning of nonlinear manifold

FOUR KEYWORDS | Autoencoder in Wikipedia

Autoencoder

From Wikipedia, the free encyclopedia

An **autoencoder**, **autoassociator** or **Diabolo network**[1]:19 is an artificial neural network used for unsupervised learning of efficient codings.[2][3] The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction. Recently, the autoencoder concept has become more widely used for learning generative models of data.[4][5]

Contents [hide]

1 Structure
- 1.1 Variations
  - 1.1.1 Denoising autoencoder
  - 1.1.2 Sparse autoencoder
  - 1.1.3 Variational autoencoder (VAE)
  - 1.1.4 Contractive autoencoder (CAE)
- 1.2 Relationship with truncated singular value decomposition (TSVD)

2 Training

3 See also

4 References

[KEYWORDS]

❖

❖

❖

❖

FOUR KEYWORDS | Nonlinear dimensionality reduction

Nonlinear dimensionality reduction

From Wikipedia, the free encyclopedia

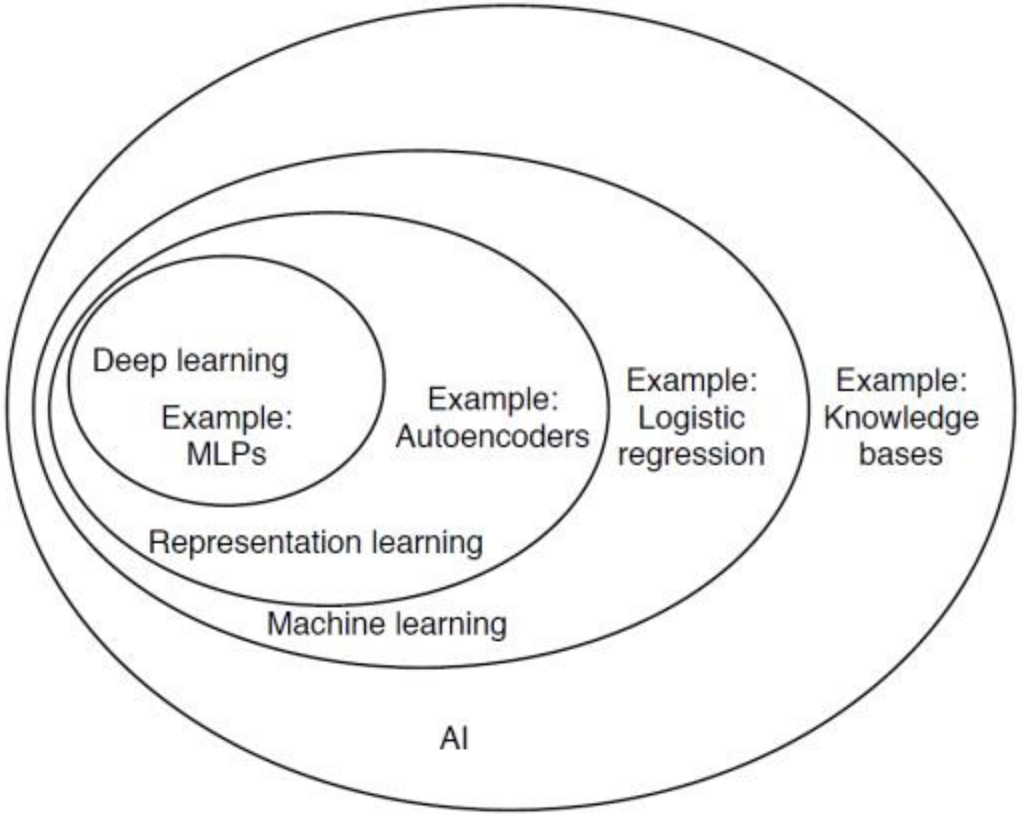
Below is a summary of some of the important algorithms from the history of **manifold learning** and **nonlinear dimensionality reduction** (NLDR).[1][2] Many of these non-linear dimensionality reduction methods are related to the linear methods listed below. Non-linear methods can be broadly classified into two groups: those that provide a mapping (either from the high-dimensional space to the low-dimensional embedding or vice versa), and those that just give a visualisation. In the context of machine learning, mapping methods may be viewed as a preliminary feature extraction step, after which pattern recognition algorithms are applied. Typically those that just give a visualisation are based on proximity data – that is, distance measurements.

Contents [hide]

1 Related Linear Decomposition Methods
2 Applications of NLDR
3 Manifold learning algorithms
3.1 Sammon's mapping
3.2 Self-organizing map
3.3 Principal curves and manifolds

[KEYWORDS]

FOUR KEYWORDS | Representation learning

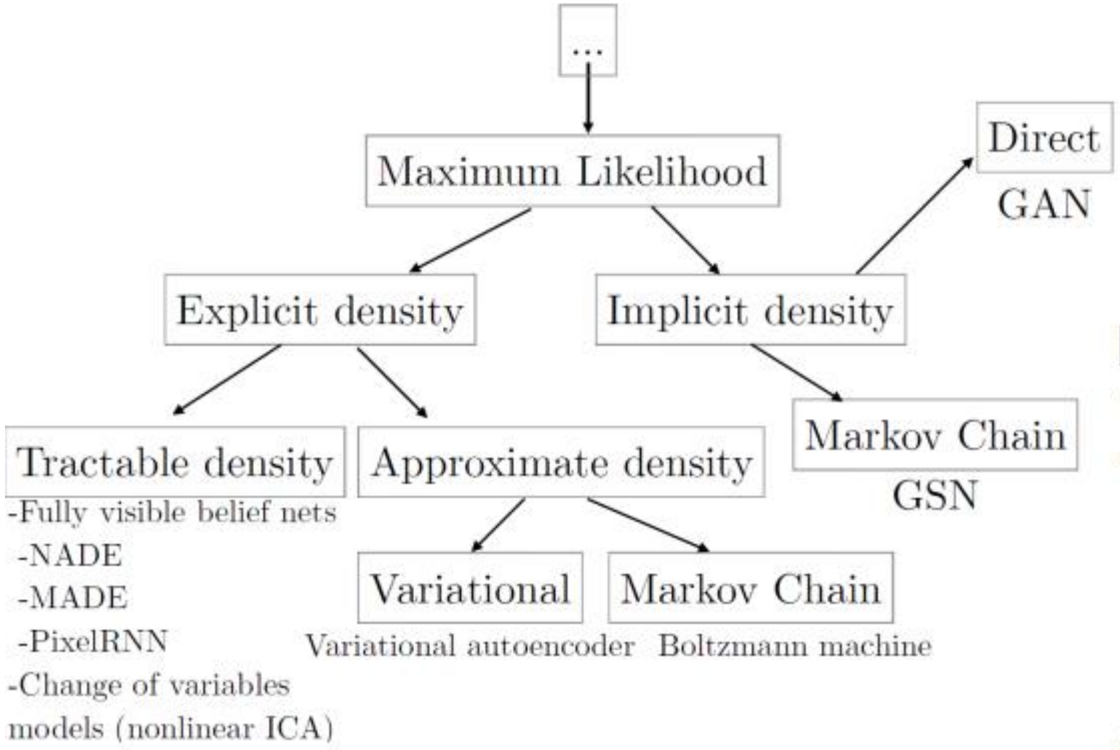


[KEYWORDS]

- ❖
- ❖
- ❖

http://videolectures.net/kdd2014_bengio_deep_learning/

FOUR KEYWORDS | ML density estimation



http://www.iangoodfellow.com/slides/2016-12-04-NIPS.pdf

[KEYWORDS]

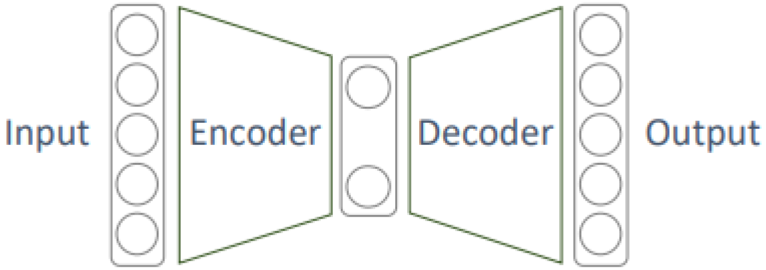
❖

❖

❖

❖

FOUR KEYWORDS | Summary



오토인코더를 학습할 때:

Unsupervised learning
ML density estimation

학습된 오토인코더에서:

Manifold learning
Generative model learning

01. Revisit Deep Neural Networks

- Machine learning problem
- Loss function viewpoints I : Back-propagation
- Loss function viewpoints II : Maximum likelihood
- Maximum likelihood for autoencoders

01. Revisit Deep Neural Networks

- Machine learning problem
- Loss function viewpoints I : Back-propagation
- Loss function viewpoints II : Maximum likelihood
- Maximum likelihood for autoencoders

KEYWORD : ML density estimation

01. Collect training data

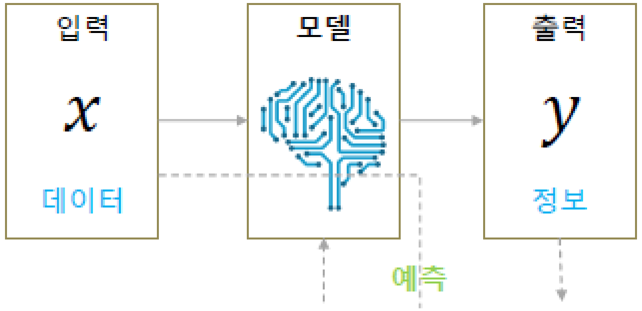
02. Define functions

03. Learning/Training

Find the optimal parameter

04. Predicting/Testing

Compute optimal function output



주어진 데이터를 제일 잘
설명하는 모델 찾기

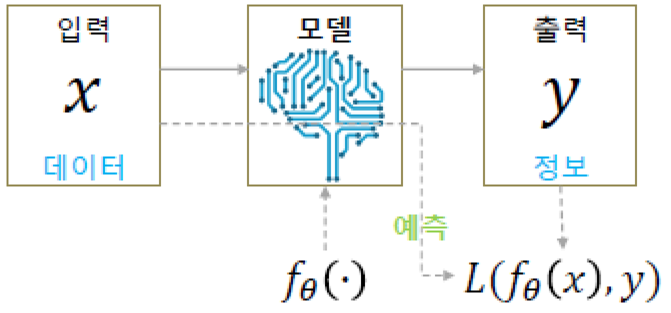
고정 입력, 고정 출력

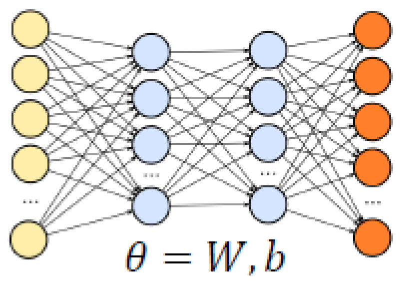
01. Collect training data

02. Define functions

03. Learning/Training

04. Predicting/Testing





Assumption 1.

Total loss of DNN over training samples is the sum of loss for each training sample

Assumption 2.

Loss for each training example is a function of final output of DNN

01. Collect training data

02. Define functions

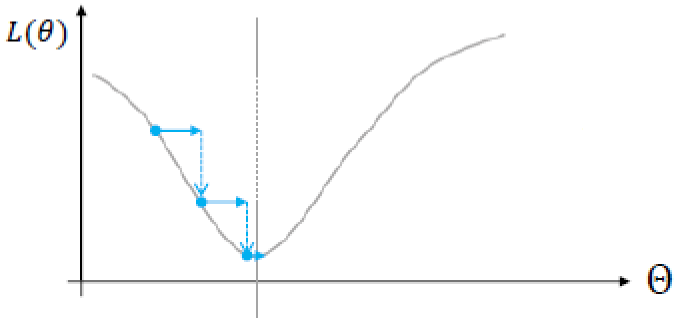
03. Learning/Training

04. Predicting/Testing

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} L(f_\theta(x), y)$$

Iterative Method

Questions	*Strategies*
How to update $\theta \rightarrow \theta + \Delta\theta$	
When we stop to search??	



01. Collect training data

02. Define functions

03. Learning/Training

04. Predicting/Testing

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} L(f_\theta(x), y)$$

Gradient Descent

Questions	*Strategies*
How to update $\theta \rightarrow \theta + \Delta\theta$	Only if $L(\theta + \Delta\theta) < L(\theta)$
When we stop to search??	If $L(\theta + \Delta\theta) == L(\theta)$
How to find $\Delta\theta$ so that $L(\theta + \Delta\theta) < L(\theta)$?	$\Delta\theta = -\eta \nabla L$, where $\eta > 0$

$L(\theta + \Delta\theta) = L(\theta) + \nabla L \cdot \Delta\theta + second\ derivative + third\ derivative + \cdots$

$L(\theta + \Delta\theta) \approx L(\theta) + \nabla L \cdot \Delta\theta$

$L(\theta + \Delta\theta) - L(\theta) = \Delta L = \nabla L \cdot \Delta\theta$

If $\Delta\theta = -\eta \Delta L$, then $\Delta L = -\eta \|\nabla L\|^2 < 0$, where $\eta > 0$ and called learning rate

∇L is gradient of L and indicates the steepest increasing direction of L

01. Collect training data

02. Define functions

03. Learning/Training

04. Predicting/Testing

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} L(f_\theta(x), y)$$ Gradient Descent

DB: $\mathcal{D}$ → DNN: $\theta_k = w_k, b_k$ → LOSS: $L(\theta_k, \mathcal{D})$

전체 데이터에 대한 로스 함수가 각 데이터 샘플에 대한 로스의 합으로 구성되어 있기에 미분 계산을 효율적으로 할 수 있다.

만약 곱으로 구성되어 있으면 미분을 위해 모든 샘플의 결과를 메모리에 저장해야 한다.

$$L(\theta_k, \mathcal{D}) = \sum_i L(\theta_k, \mathcal{D}_i)$$

$$\nabla L(\theta_k, \mathcal{D}) = \sum_i \nabla L(\theta_k, \mathcal{D}_i)$$

Redefinition → $$\nabla L(\theta_k, \mathcal{D}) \triangleq \sum_i \nabla L(\theta_k, \mathcal{D}_i) / N$$

$$\nabla L(\theta_k, \mathcal{D}) \approx \sum_j \nabla L(\theta_k, \mathcal{D}_i) / M, \text{ where } M < N$$

M : batch size

$$\theta_{k+1} = \theta_k - \eta \nabla L(\theta_k, \mathcal{D})$$

01. Collect training data

02. Define functions

03. Learning/Training

04. Predicting/Testing

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} L(f_\theta(x), y)$$

Gradient Descent + Backpropagation

DB: $\mathcal{D}$

DNN: $\theta_k = w_k, b_k$

LOSS: $L(\theta_k, \mathcal{D})$

$$\theta_{k+1} = \theta_k - \eta \nabla L(\theta_k, \mathcal{D})$$

$$\left.\begin{aligned} w_{k+1}^l &= w_k^l - \eta \nabla_{w_k^l} L(\theta_k, \mathcal{D}) \\ b_{k+1}^l &= b_k^l - \eta \nabla_{b_k^l} L(\theta_k, \mathcal{D}) \end{aligned}\right\}$$ 특정 레이어에서 파라미터 갱신식

로스함수의 미분값이 딥뉴럴넷 학습에서 제일 중요!!

[Backpropagation Algorithm]

1. Error at the output layer

$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

- C : Cost (Loss)
- a : final output of DNN
- $\sigma(\cdot)$: activation function

2. Error relationship between two adjacent layers

$$\delta^l = \sigma'(z^l) \odot \left(\left(w^{l+1} \right)^T \delta^{l+1} \right)$$

3. Gradient of C in terms of bias $\nabla_{b^l} C = \delta^l$

4. Gradient of C in terms of weight $\nabla_{w^l} C = \delta^l \left(a^{l-1} \right)^T$