5 장: 딥빌리프네트워크

5.1 RBM 구조
5.2 RBM 학습
5.3 딥빌리프네트워크
5.4 응용 사례

컨볼루션 신경망은 감독 학습을 위한 딥러닝 모델이다. 이 절과 다음 절에서는 무감독 학습을 위한 딥러닝 모델을 살펴본다. 이 절에서 주로 다룰 모델은 딥빌리프네트워크(Deep Belief Network, DBN)로 2006 년에 최초로 컨볼루션 연산을 사용하지 않고 심층 구조 상에서 학습을 성공시킨 모델이다[1]. 오늘날에는 다른 무감독학습 딥러닝 모델들에 비해 그다지 자주 사용되지 않지만, 여전히 아이디어와 역사적인 의미는 음미할 가치가 있다.
딥빌리프네트워크는 제한 볼쯔만 머신(Restricted Boltzmann Machine, RBM)이라는 신경망 모델을 단위 블록으로 사용하여 층을 쌓음으로써 깊은 신경망을 구성하는 딥러닝 모델이다. RBM 은 주어진 입력과 똑 같은 출력을 생성하도록 하는 오토인코딩 과제를 수행하는 모델이다. 또한 컨볼루션 신경망이 계산을 수행할 때 결정적인 계산을 수행하는 것과는 달리 딥빌리프네트워크는 확률적으로 출력값을 계산한다.

5.1 절에서는 DBN 의 빌딩블록인 RBM 의 구조를 살펴본다. 5.2 절에서는 RBM 의 학습 알고리즘을 살펴본다. 5.3 절에서는 RBM 을 순차적으로 학습하며 적층하여 오토인코더 딥빌리프네트워크를 구성하는 방법을 살펴본다. 또한 DBN 을 감독학습에 사용하는 방법을 기술한다. 5.4 절은 딥 빌리프 네트워크의 응용 사례를 살펴본다.

5.1 RBM 구조

제한 볼쯔만 머신은 입력 뉴런층과 하나의 은닉 뉴런층으로 구성된 무감독학습 신경망 모델이다. 원래의 볼쯔만 머신은 모든 뉴런들이 다른 모든 뉴런과 연결된 완전그래프 형태의 망구조를 가지나 딥러닝에 사용되는 제한 볼쯔만 머신은 입력뉴런들간에는 연결선이 없고 또한 은닉뉴런들간에도 연결선이 없는 구조이다. 구조 측면에서 보면 다수의 뉴런을 가진 단순 퍼셉트론 구조와 같다. 그러나 퍼셉트론은 결정적 모델인데 반해서 볼쯔만 머신은 확률 모델이다. 다만 다수의 출력 뉴런을 갖고 출력이 감독학습 신호를 받지 않는다. 이러한 모델은 통계에서 은닉변수 모델(latent variable model)로도 알려져 있으며 마코프랜덤필드와 같은 계열이다.

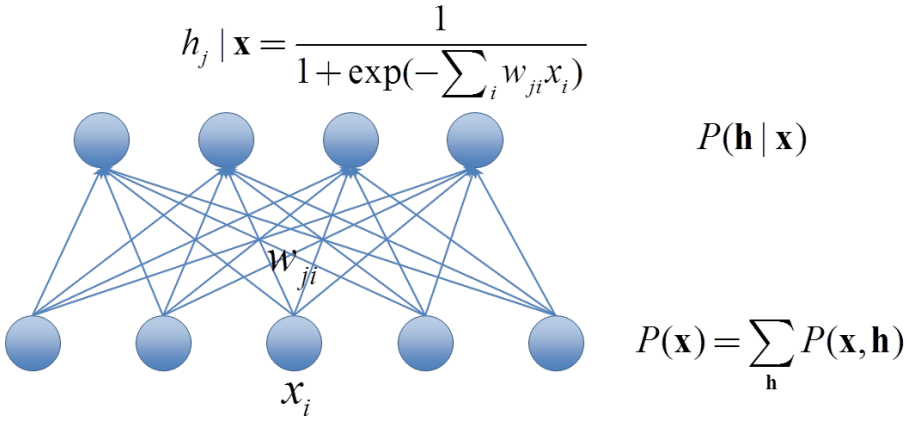

Figure 5.1 제한 볼쯔만 머신(RBMs 의 구조)

각 뉴런이 하는 계산은 퍼셉트론에서와 유사하다. 즉, 입력의 가중치합을 계산하여 이를 시그모이드 함수를 통해서 변환한다. 그러나 이 값은 확률값으로 해석된다. 즉 j 번째 은닉 뉴런은 0 과 1 의 값을 출력한다.

$$net_j = \sum_i w_{ji} x_i$$

$$P(h_j \mid \mathbf{x}) = \sigma\left(\sum_i w_{ji} x_i \right)$$

여기서 $\sigma(net)$는 인공신경망에서 활성화 함수로 사용되는 시그모이드 함수로 다음과 같이 정의된다.

$$\sigma(net) = \frac{1}{1+\exp(-net)}$$

입력 벡터 $\mathbf{x}$가 주어질 때 각각의 뉴런은 서로 독립이라고 가정하면, 은닉벡터 $\mathbf{h}$의 확률은 $P(\mathbf{h} \mid \mathbf{x})$로 표시된다.

$$P(\mathbf{h} \mid \mathbf{x}) = \prod_j P(h_j \mid \mathbf{x})$$

마찬가지로 은닉벡터 $\mathbf{h}$ 가 주어지면 입력 뉴런들은 서로 독립이라고 가정한다.

$$P(\mathbf{x} \mid \mathbf{h}) = \prod_i P(x_i \mid \mathbf{h})$$

$$P(x_i \mid \mathbf{h}) = \sigma\left(\sum_j w_{ji} h_j \right)$$

볼쯔만 머신은 감독학습 시스템이 아니기 때문에 목표 출력값들이 주어지지 않는다. 따라서 에러를 정의할 수 없다. 대신에 볼쯔만 머신은 관측된 데이터를 재생성할 확률, 즉, 우도 $P(\mathbf{x})$를 최대화하도록 학습한다. 그런데 우도를 관측 데이터로부터만 추론하지 않고 볼쯔만 머신은 은닉변수를 사용하여 추정한다. 확률의 합의 법칙을 사용하면

$$\begin{aligned} P(\mathbf{x}) &= \sum_{\mathbf{h}} P(\mathbf{x}, \mathbf{h}) \\ &= \sum_{\mathbf{h}} \frac{\exp[-E(\mathbf{x}, \mathbf{h})]}{\sum_{\mathbf{h}'} \sum_{\mathbf{x}'} \exp[-E(\mathbf{x}', \mathbf{h}')]} \\ &= \frac{\sum_{\mathbf{h}} \exp[-E(\mathbf{x}, \mathbf{h})]}{\sum_{\mathbf{h}'} \sum_{\mathbf{x}'} \exp[-E(\mathbf{x}', \mathbf{h}')]} \end{aligned}$$

여기서 통계 물리에서 에너지 $E(\mathbf{x}, \mathbf{h})$를 가진 시스템의 확률이 볼쯔만 분포로 주어진다는 다음 사실을 이용하였다.

$$P(\mathbf{x}, \mathbf{h}) = \frac{1}{Z} \exp[-E(\mathbf{x}, \mathbf{h})] = \frac{\exp[-E(\mathbf{x}, \mathbf{h})]}{\sum_{\mathbf{x}', \mathbf{h}'} \exp[-E(\mathbf{x}', \mathbf{h}')]}$$

확률과 에너지는 $P(\mathbf{x},\mathbf{h}) \propto \exp[-E(\mathbf{v},\mathbf{h})]$로서 반비례 관계에 있다. 볼쯔만 머신에서는 에너지를 다음과 같이 정의한다.

$$E(\mathbf{x},\mathbf{h}) = -\sum_j \sum_i h_j w_{ji} x_i$$

여기서 w_{ji}은 i번째 입력 뉴런과 j번째 은닉 뉴런을 연결하는 연결선 가중치이며 $w_{ji} = w_{ij}$로 대칭이다.

5.2 RBM 학습

이제 학습식을 유도해 보자. 볼쯔만 머신을 학습하는 것은 최대 우도 값 또는 로그우도 값을 갖는 가중치 벡터를 찾는 문제와 같다.

$$\mathbf{w}^* = \arg\max \log P(\mathbf{x},\mathbf{h}) = \arg\max L(X;\theta)$$

로그우도 추정법의 정의로부터 이를 $\mathbf{w}$ 파라미터에 대한 미분값이 0 으로 수렴하면 학습이 된 것이므로

$$\frac{\partial L(\mathrm{X};\theta)}{\partial w_{ij}} \rightarrow 0$$

이를 구하기 위해서 가중치에 대해서 미분하면

$$\begin{aligned}
&\frac{\partial L(\mathrm{X};\theta)}{\partial w_{ji}} \\
&= \int P(\mathbf{x},\theta) \frac{\partial \log f(\mathbf{x};\theta)}{\partial \theta} d\mathbf{x} - \frac{1}{N}\sum_{d=1}^{N} \frac{\partial \log f(\mathbf{x}^{(d)};\theta)}{\partial \theta} \\
&= \left\langle x_i h_j \right\rangle_{P(\mathbf{x},\theta)} - \left\langle x_i h_j \right\rangle_{\mathrm{X}} \\
&= \left\langle x_i h_j \right\rangle_{\infty} - \left\langle x_i h_j \right\rangle_{0} \\
&\qquad \left\langle x_i h_j \right\rangle_{\infty} - \left\langle x_i h_j \right\rangle_{0} \approx \left\langle x_i h_j \right\rangle_{1} - \left\langle x_i h_j \right\rangle_{0}
\end{aligned}$$

위의 식에서 $\langle f \rangle_P$ 은 분포 P 에 대한 f 의 기대치를 표시한다. $\langle f \rangle_P - \langle f \rangle_X$ 는 두 개의 기대치 차이이다. $\langle x_i x_j \rangle_\infty$ 는 이를 몬테칼로 시뮬레이션으로 추정한 것이다.

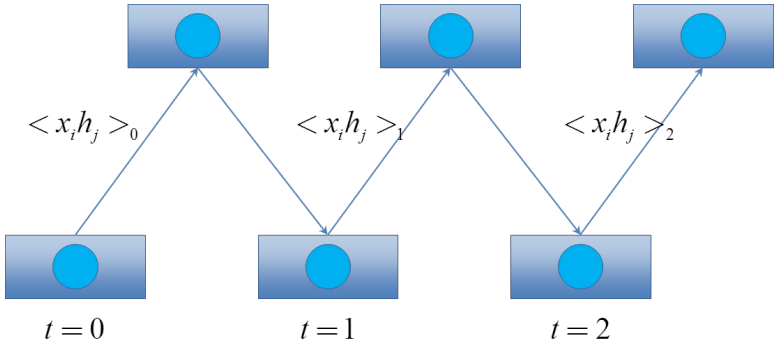

Figure 5.2: 제한 볼쯔만 머신의 학습

딥빌리프네트워크 DBN 은 무한번 반복해야 계산되는 $\langle x_i h_j \rangle_\infty$ 를 한번의 몬테칼로 시뮬레이션으로 대치하여 추정한다. 즉

$$\langle x_i h_j \rangle_\infty - \langle x_i h_j \rangle_0 \approx \langle x_i h_j \rangle_1 - \langle x_i h_j \rangle_0$$

따라서 학습식은 다음과 같이 주어진다.

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$
$$\Delta w_{ji} = \eta \frac{\partial L(X;\theta)}{\partial w_{ji}} = \eta\left(\langle x_i h_j \rangle_0 - \langle x_i h_j \rangle_\infty\right)$$

0. 은닉뉴런수 H 를 정하고 가중치를 초기화 한다.
1. 관측 뉴런에 학습 벡터 $\mathbf{x}$ 를 입력한다.
2. For j = 1 to H
3. $P(h_j \mid \mathbf{x})$의 값을 계산한다.
4. 위의 2-3 을 여러번 반복한다.
5. 모든 i, j 쌍에 대해서 v_i와 h_j가 동시에 on 이 되는 수를 계산해서 $< v_i h_j >_0$라 한다.
6. For i =1 to I
7. $P(x_i \mid \mathbf{h})$의 값을 계산한다.
8. 위의 6-7 을 여러번 반복한다.
9. For j = 1 to H
10. $P(h_j \mid \mathbf{x})$의 값을 계산한다. //위의 6-7 의 결과를 이용하여//
11. 위의 9-10 을 여러번 반복한다.
12. 모든 i, j 쌍에 대해서 x_i와 h_j가 동시에 on 이 되는 수를 계산해서 $< x_i, h_j >_1$라 한다.
13. 모든 i, j 쌍에 대해서 가중치를 다음과 같이 변경한다.
    $\Delta w_{ji} = \eta(< x_i h_j >_0 - < x_i h_j >_1)$

5.2.1 응용예: 패턴 생성

제한 볼쯔만 머신을 이용한 응용의 한 가지로 패턴 생성 문제를 소개한다. 학습에 사용했던 데이터를 다시 생성해내는 문제가 왜 흥미로운 문제인지 궁금해하는 사람들이 있을 것이다.

패턴 생성 문제는 SW 에 입력한 데이터만이 아니라 입력하지 않았던 데이터도 생성해낼 수 있도록 하는 문제로서, 단순히 매치되는 정보를 인출하는 문제인 정보검색 문제와는 구별된다. 특히, 제한 볼쯔만 머신은 에너지를 이용하여 정의된 확률 분포를 학습하는 모델이므로, 학습 데이터의 분포에 따라 생성 데이터의 분포도 영향을 받으며, 이로 인해 본 적이 없는 데이터도 생성할 수 있게 된다.

실제 데이터 집합의 하나로 손으로 쓴 숫자 이미지 데이터 집합인 MNIST 데이터집합을 소개한다. 이는 딥러닝 문제에서 자주 사용되는 데이터 집합 중 하나로 기본적으로 28 × 28 크기의 6 만장의 학습용 이미지와 1 만장의 테스트용 이미지로 구성되어 있다. 이 이미지는 상하좌우에 공란이 다고 존재하므로 테두리를 잘라내어 16 × 16 크기로 만든 이미지를 사용할 수도 있다. 이 데이터를 가지고 은닉 뉴런 수가 50 개인 RBM 을 학습시키는 과정을 다시 설명해보자. 이 때, 입력 차원은 256 이므로, 가중치 W 는 256 × 50 인 행렬로 표현이 가능하다.
학습식과 견주어 볼 때, 결과적으로 학습용 이미지의 활성 입력과 은닉 뉴런의 활성 인자 사이의 가중치의 강도는 높이게 되고, 재생성된 이미지의 활성 입력과 은닉 뉴런의 활성 인지 사이의 가중치의 강도는 낮추게 된다. 만약, 학습 이미지와 재생성된 이미지가 동일할 경우, 증가된 가중치와 감소된 가중치가 평형을 이루어 모델은 수렴한다.

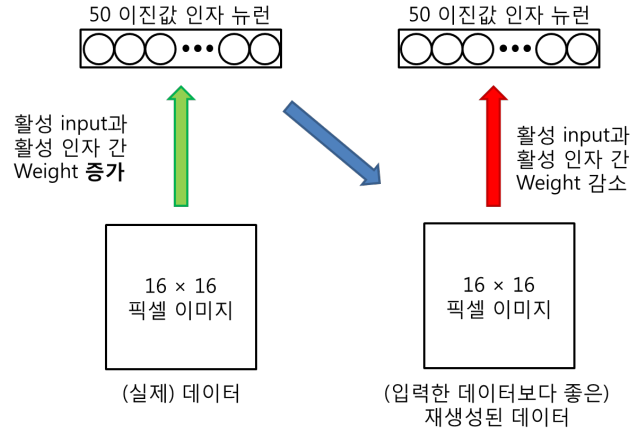

Figure 5.3. 제한 볼쯔만 머신에 의한 숫자 학습 방법

다양한 숫자 2 이미지들을 이용하여 RBM 을 학습시킨 후에, 위의 과정을 통해 숫자를 생성해보자. 아래 그림 좌측과 같이 모델이 학습한 특성을 바탕으로 새로운 숫자 2 이미지를 생성하였다. 반면에, 학습에 사용한 데이터와 상대적으로 모양이 크게 다른 숫자 3 을 입력으로 주었을 때에는, 모델이 이 입력을 2 에 가깝게 해석하려고 하는 경향을 볼 수 있다. 이러한 현상을 보이는 이유는 RBM 안에 있는 은닉 뉴런들이 주로 2 를 표현하는 특성들만을 학습하였기 때문이다.

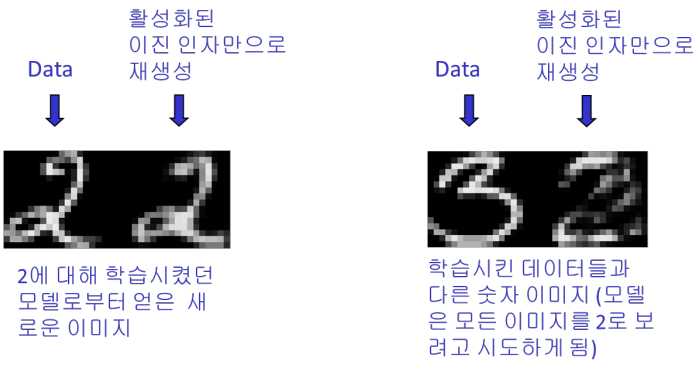

Figure 5.4 문자 생성 실험

5.3 딥빌리프네트워크

딥빌리프네트워크는 제한 볼쯔만 머신을 빌딩블록으로 해서 여러층을 쌓은 딥러닝 아키텍쳐이다. 심층의 신경망을 학습하는데 있어서 오차 소멸(vanishing gradient) 문제가 있다. 즉 심층 신경망에 대해서 오류역전파 알고리듬을 적용하면 출력층에 대해서는 가중치가 잘 교정되나 입력층 쪽으로 감에 따라서 오차값이 점차 적어져서 입력층에 대한 가중치가 교정이 잘 되지 않는 문제가 있었다. 이 문제를 해결하기 위해서 DBN 은 아래층(입력에 가까운 층)에서부터 위층으로 가면서 순차적으로 선훈련을 진행한다. 즉 아래층에 대해서 가중치를 먼저 학습한 후 이 가중치를 고정한 다음 그 다음 층의 가중치를 학습한다. 이 기술은 층별 선훈련 (layerwise pre-training)이라고 하며 딥러닝을 위한 핵심 기술로 제안되었다.

딥빌리프네트워크는 크게 두 가지로 구분된다. 하나는 입력과 같은 출력을 재생성하도록 하는 오토인코더이고 다른 하나는 분류기로 사용하는 것이다. 전자는 무감독학습 구조이고 후자는 감독학습 구조이다. 딥빌리프네트워크는 기본적으로 무감독학습 구조이다. 그러나 분류 문제를 풀기 위해서는 무감독학습의 마지막 은닉층을 입력으로 하고 출력층을 하나 추가하여 감독학습을 시킴으로써 분류 문제를 풀 수 있다.

그림 6.5 는 x-h1-h2-h3 구조의 딥빌리프네트워크를 보여준다. 먼저 입력층에 대한 은닉층 h1 을 모든 학습데이터집합 D 에 대해서 학습한다. 학습된 $w_{ji}^{(1)}$를 고정한다. 이를 이용하여 h2 를 학습한다. 방식은 학습 데이터 x 를 입력층에 할당한 후 고정된 $w_{ji}^{(1)}$를 사용하여 h1 을 계산한다. 이 h1 값을 입력으로 생각하고 은닉층 h2 의 값들을 볼쯔만머신 학습 방법을 사용하여 무감독 학습으로 학습함으로써 $w_{ji}^{(2)}$값들을 구한다. 이와 같은 방식으로 순차적으로 빌리프네트워크 구조의 은닉층을 쌓아가며 심층 신경망을 구성한다.

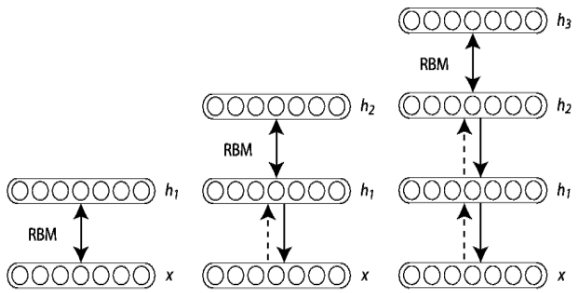

Figure 5.5 층별 선훈련 방법

그림 6.6 은 딥빌리프네트워크 구조를 이용하여 문자영상을 재구성하는 오토인코더를 구성한 예이다. 이 예에서는 28 x 28 = 784 차원의 이진값을 갖는 영상벡터 x 를 입력으로 사용하여 x-1000-500-250-30-250-500-1000-x 의 구조를 갖는 딥네트워크 구조이다.

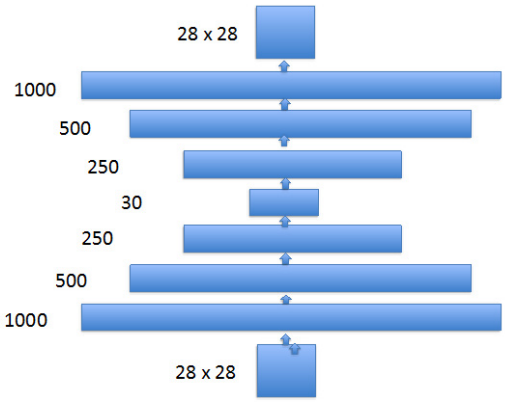

Figure 5.6 딥빌리프네트워크 오토인코더

딥빌리프네트워크는 기본적으로 무감독 학습 구조이다. 그러나 분류 문제를 풀기 위한 감독학습으로도 사용할 수 있다. 마지막 층의 은닉층을 입력으로 하고 출력층을 추가하여 감독학습을 시킴으로써 분류 문제를 풀 수 있다. 그림 6.6 은 딥빌리프네트워크 구조를 이용하여 문자 인식을 하기 위한 분류기를 구성한 예이다. 이 예에서는 28 x 28 = 784 차원의 이진값을 갖는 영상벡터 x 를 입력으로 사용하여 x-1000-500-250-30-250-500-1000-y 의 구조를 갖는 딥네트워크 구조이다.

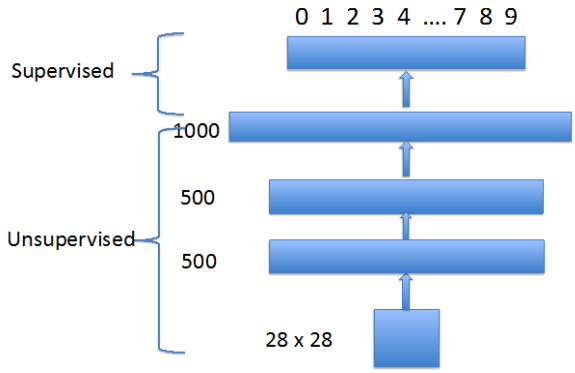

Figure 5.6 딥빌리프네트워크 분류기

다루는 문제에 따라 다양한 네트워크 구성이 가능하지만, 구조가 결정된 후에는 딥네트워크 상에 정의되는 확률식은 특정 형태를 따른다. 그림 6.5 에서 사용된 변수를 참고하여 일반적인 네트워크를 표현해보자. 관측가능한 입력 벡터 x 와 l 개의 은닉 계층을 가지며, k 번째 계층의 값을 h^k 라고 하자. 이 때, 딥네트워크 구조가 결정된 딥빌리프네트워크는 다음과 같은 결합확률 분포를 갖는 확률 그래프 모델로 정의된다.

$$P(x,h^1,\cdots,h^l)=\left(P(x\mid h^1)\prod_{k=1}^{l-2}P(h^k\mid h^{k+1})\right)P(h^{l-1},h^l)$$

계층 간의 관계가 주로 조건부 확률을 이용해 표현되었다. 하지만, 최상단의 두 계층인 h^{l-1}과 h^l의 관계는 조건부 확률이 아님을 유의하자. 계층별 관계를 나타내는 조건부 확률 및

결합확률(최상단의 두 계층)은 RBM 을 이용한 층별 선훈련을 통해 얻는다. 이와 같이 계층별 관계 및 네트워크를 확률을 이용해 표현할 경우, 학습데이터에 포함되지 않은 패턴도 샘플링을 통해 생성할 수 있다는 장점을 갖는다.

영상벡터를 학습하는 딥빌리프네트워크의 일반적인 학습과정을 정리하면 아래와 같다.

1. 영상의 픽셀값들을 입력벡터로 사용하여 층별 선훈련을 통해 첫 번째 계층을 학습한 후, 은닉층의 출력벡터를 구한다.
2. 하위 은닉층의 출력벡터를 입력벡터로 취급하여 다음 계층을 학습한다.
3. 2 의 과정을 딥네트워크 구조에 따라 상위 계층으로 올라가면서 반복한다.
4. (오토인코더 학습의 경우) wake-sleep 알고리즘을 통해 미세조정을 수행한다.
   (분류기 학습의 경우) 오류 역전파 알고리즘을 통해 미세조정을 수행한다.

위의 과정을 살펴보면 층별 선훈련이 입력계층에서 상위 계층 방향으로 탐욕적(greedy)인 방식으로 작동됨을 알 수 있다. 이로 인해 학습 과정의 결과물이 과연 문제 해결에 적합한 모델이 될 지 우려가 있을 수 있다. 이에 대한 다소 간접적인 설명으로서, 토론토 대학의 힌튼 교수는 제안한 학습과정이 학습데이터에 대한 딥빌리프네트워크의 로그우도의 하한을 증가시키는 과정임을 증명하였다.

위의 4 번 과정은 1~3 과정에서 학습한 딥빌리프네트워크를 응용에 따라 미세조정하여 좀더 좋은 성능을 얻는 과정이다. 분류기 학습을 위한 미세조정은 다양한 다른 딥네트워크에서 사용되는 오류 역전파 방식을 사용한다. 오토인코더 학습을 위한 미세조정에는 wake-sleep 알고리즘의 변형들을 사용해볼 수 있다. 이 알고리즘은 그림 5.7 과 같이 인식용 연결과 생성용 연결이 별도로 주어진 네트워크에서 정의되며, wake-sleep 의 두 단계를 번갈아 수행함으로써 데이터를 학습한다. wake 단계에는 인식용 연결을 이용해 layer 의 값을 결정하고 생성용 연결의 가중치를 조정하여 우도를 높힌다. Sleep 단계에는 생성용 연결을 이용해 은닉 계층의 값을 결정하고 인식용 연결의 가중치를 조정하여 우도를 높힌다. 그림 5.5 를 보면 입력단계부터 최상위 단계 아래까지는 그림 5.7 의 구조와 동일한 연결선을 가지므로, RBM 의 연결 가중치를 인식용/생성용 연결 가중치의 초기값으로 동일하게 설정한 후에 이 알고리즘을 사용할 수 있다. 다만, 딥빌리프네트워크에서는 최상위 두 계층은 무방향성 연결만을 가지는 모듈이므로 한 계층 아래의 값을 이용하여 우도를 높힌다.

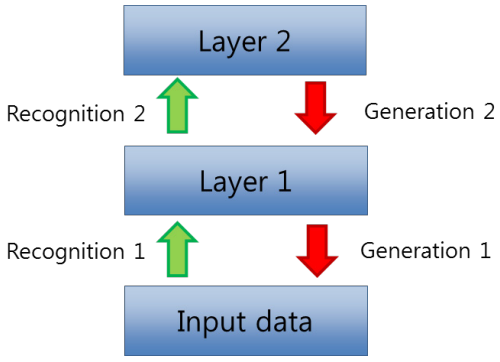

Figure 5.7 wake-sleep 알고리즘을 위한 전형적인 네트워크

5.3.1 응용 예: 필기체 숫자 인식문제와 생성문제

딥빌리프네트워크를 이용하여 필기체 숫자 인식 문제와 생성 문제에의 적용 예를 소개한다. 이를 위해 5.2 절의 응용에서 사용한 필기체 숫자 데이터집합인 MNIST를 다시 사용한다. 인식 문제와 생성 문제를 함께 다루기 위해 그림 5.8 의 딥빌리프네트워크 구조인 28 x 28 - 500-500-2000-10 의 구조를 이용하여 MNIST 데이터를 학습하였다. 여기서 500-2000 과 2000-10 은 연결선에 방향이 없는 RBM으로 되어 있어, 연합 메모리와 같은 역할을

수행한다. 힌튼 교수가 제공하는 데모가 다음 웹사이트에 제공되어 있다.
(http://www.cs.toronto.edu/~hinton/adi/index.htm)

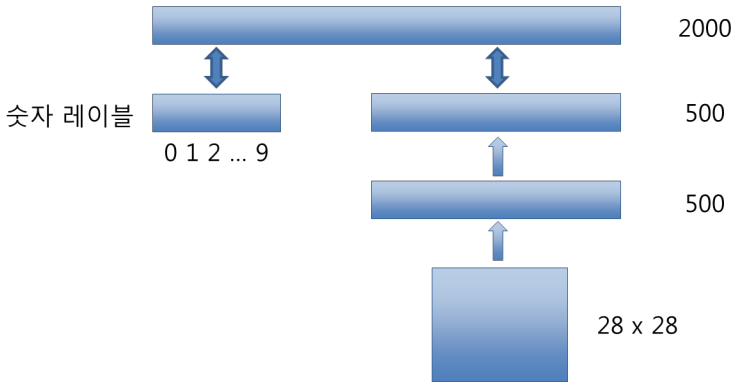

Figure 5.8 필기체 숫자 인식 및 생성을 위한 딥빌리프 네트워크 구조

0~9 의 숫자 레이블 부분 중 하나를 골라 고정하고 연합 메모리에서 샘플링을 통해 샘플을 생성할 수 있다. 그림 5.9 는 각 숫자 레이블 별로 깁스 샘플링을 통해 얻은 샘플들의 예이다. 반대로 28 x 28 부분에 학습 데이터에 포함되지 않았던 숫자 이미지를 고정하고 레이블 부분의 출력을 통해 분류를 수행할 수 있다. 그림 5.10 은 그 이미지의 예이다. 그림 5.11 은 분류 성능을 비교한 것으로 단순히 MLP 를 오류역전파를 통해 학습하거나 지지벡터머신을 사용한 결과보다 우수하다.

Figure 5.9 딥빌리프네트워크가 생성한 필기체 숫자 이미지

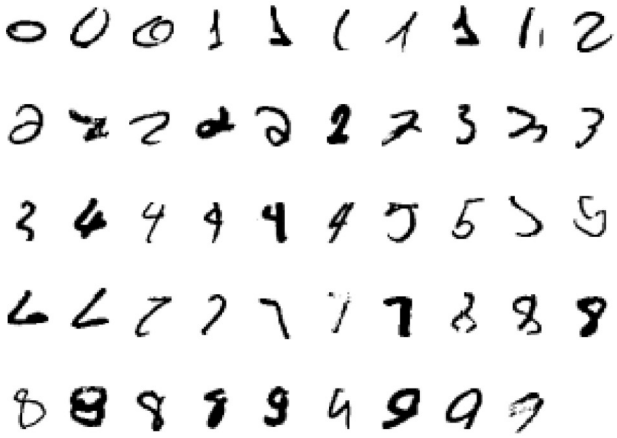

Figure 5.10 딥빌리프네트워크를 통해 옳게 인식된 필기체 숫자 이미지들의 예

방법	에러율
RBM 기반 생성모델	1.25%
지지벡터머신	1.4%
오류역전파(1000 은닉 뉴런)	~1.6%
오류역전파(500 → 300 은닉 뉴런)	~1.6%
K-Nearest Neighbor	~3.3%

Figure 5.11 MNIST 데이터에 대한 에러율 비교

5.3.2 Dropout

Dropout 은 딥네트워크를 학습하는 데에 있어 은닉 계층 안의 일부 노드를 강제로 꺼버리는 과정을 통해 분류 성능을 크게 향상시키는 트릭이다. ImageNet 에 딥러닝을 성공적으로 적용했던 사례인 AlexNet 에서도 사용되었다.

일반적으로는 매 학습 데이터 하나마다 각 은닉 뉴런 별로 0.5 의 확률로 강제로 꺼질 뉴런을 무작위로 선택한다. 이러한 과정을 통해, 만약 H 개의 은닉 뉴런을 가지고 있었다면, 마치 2^H 개의 다른 네트워크를 가지고 앙상블을 만든 효과를 갖게 된다. 하지만, 앙상블의 모든 네트워크의 가중치는 공유되기 때문에 강력한 정규화 효과를 지니게 된다. 즉, 여러 개의 모델을 한번에 다루는 효과를 부여하게 되어 훨씬 설명력이 좋은 다수의 모델을 학습하게 된다.

학습을 마친 후에 dropout 을 수행한 모델을 가지고 추론할 경우에는 어떻게 계산해야 할까? 다수의 모델의 결과를 합쳐서 이용하는 효과를 가지므로, 가중치의 값들을 반으로 줄여서 연산을 수행하면 된다. 이 과정은 출력들의 기하평균을 취하여 다음 계층으로 보내는 것과 동일한 효과를 낸다.

이론적인 연구는 현재까지 제한적으로 수행되어 있어서, 현재까지는 경험적인 지식들이 좀더 알려져 있다. Dropout 을 사용할 때 가질 수 있는 실제적인 궁금증에 대해 가능한 답을 해보자.
만약, 은닉 계층이 여러 개일 경우에는 어떻게 하는 것이 바람직한 dropout 방법인가? 매 은닉 계층마다 0.5 dropout 을 사용하는 것이 좋다.
그렇다면, 입력 계층에도 dropout 을 사용해야 하는가? 실험적으로 밝혀진 바에 따르면 입력계층에도 dropout 을 사용하는 것이 좋다.

딥빌리프네트워크에서는 dropout 을 어떻게 사용하는가? 미세조정과정에서 감독학습을 할 때 적용할 수 있다.

5.4 응용 사례

딥빌리프네트워크의 활용 사례로 두 가지를 소개한다. 이는 얼굴 인식을 위한 감독 학습문제와 문서 검색을 위한 문서의 코드 학습문제이다. 그림 5.12 는 얼굴 방향 인식에 사용된 데이터를 보여준다. 무감독 학습을 위해서 11,000 장의 무표지 얼굴 영상을 사용하였다. 감독 학습은 세 가지 경우로 나뉘어 각각, 100, 500, 1000 장의 표지된 얼굴 영상을 사용하였다. 학습 성능의 평가를 위해서는 새로운 얼굴 사진을 사용하였다. 3 개의 RBM 층으로 무감독 학습을 수행한 후 마지막에 한 개의 감독학습층을 추가한 4 층짜리 딥빌리프네트워크 구조를 사용하였다. RBM 층은 각각 1000 개의 은닉 뉴런으로 구성된 무감독 학습을 통해 오토인코더를 학습하였다. 마지막의 감독학습층에서는 가우시안 프로세스(Gaussian Process, GP)를 사용하였다.

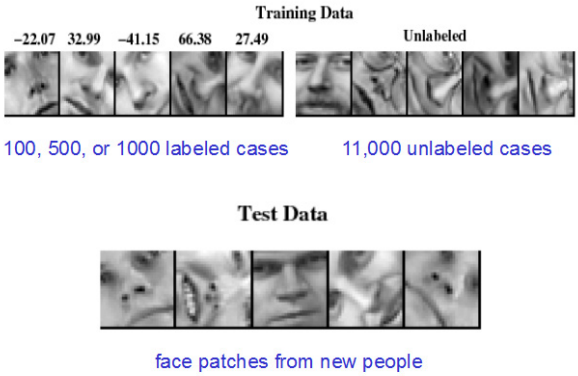

Figure 5.12 얼굴 방향 인식 문제의 학습 및 테스트 데이터

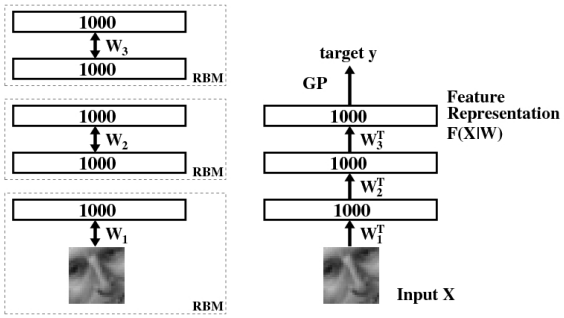

Figure 5.13 얼굴방향인식 문제의 딥빌리프네트워크 구조

그림 5.14 는 얼굴 인식 성능을 비교하여 보여 준다. 세 가지 경우를 비교하였으며, 이것은 i) 원래 픽셀 영상에 대해 GP 를 적용한 경우, ii) 딥빌리프네트워크로 무감독 학습한 특징벡터에 대해서 GP 를 적용한 경우, iii) 딥빌리프네트워크로 무감독 학습한 후 여기에 GP 층을 쌓아 전체적으로 다시 감독학습으로 전체 가중치를 미세 조정하는 경우이다. 결과는 i)에서 iii)으로 갈수록 정확도가 향상되는 것을 알 수 있다. 특히 i) 보다는 ii)와 iii)이 성능이 좋은 것은 딥빌리프네트워크로 무감독 학습한 특징이 분류 문제 해결에 도움이 된다는 것을 의미한다. ii)와 iii)을 비교할 때 마지막에 감독 학습층을 쌓은 후 전체 망을 미세조정하는 것이 또한 성능 향상에 도움이 된다는 것을 알 수 있다. 전체적인 성능은 예상되는 바와 같이 표지된 데이터의 수가 100, 500, 1000 개로 늘어날 수록 향상됨을 확인할 수 있다. 즉, 무감독학습으로 추출한 특징들은 표지 데이터의 수가 늘어나도 여전히 유효함을 알수 있다.

그림 5.15 은 무감독학습으로 재생성된 얼굴 이미지를 보여준다. 30 차원의 PCA 로 복원한 이미지 보다 30 차원의 오토인코더 딥빌리프네트워크의 성능이 더 우수함으로 시각적으로 알 수 있다. 아래쪽 패널의 그림들은 RBM 의 층을 쌓아감에 따라서 추출된 특징들이 점차 전역적이고 복잡해지는 것을 시각적으로 확인할 수 있다.

	GP on the pixels	GP on top-level features	GP on top-level features with fine-tuning
100 labels	22.2	17.9	15.2
500 labels	17.2	12.7	7.2
1000 labels	16.3	11.2	6.4

Figure 5.14 얼굴방향인식 성능 비교

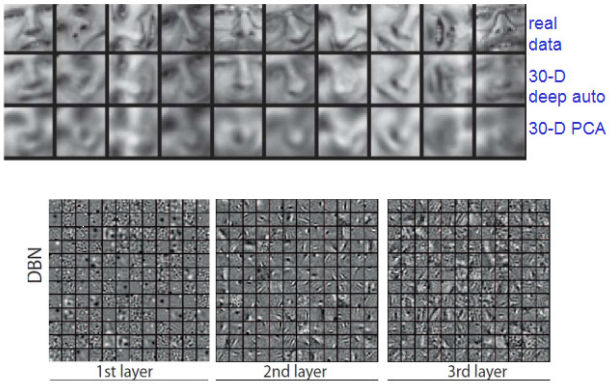

Figure 5.15 얼굴 재생성 결과 및 층별 특징 추출 결과

Hinton et al.은 또한 딥빌리프네트워크의 성능을 텍스트 데이터에 대해서도 평가하였다. 여기서 다룬 응용은 유사한 문서검색 문제로, 딥빌리프네트워크를 이용해 문서의 특징을 드러내는 코드를 생성할 수 있는지가 관심사였다. 즉, 유사한 문서에는 유사한 코드가 부여하여, 쿼리 문서를 입력하였을 때 유사한 문서들을 검색하는 문제에 활용할 수 있다. 40 만개의 뉴스그룹 문서를 수집하여 학습한 후, 학습에 사용되지 않은 40 만개의 문서를 쿼리로 삼아 동일 카테고리에 속한 문서가 검색되는지를 기준으로 하여 성능을 평가하였다. 이를 위해 무감독 학습으로 오토인코더 딥빌리프네트워크를 학습 시켰으며, 마지막에 감독학습으로 미세조정을 수행하였다. 딥빌리프네트워크의 구조는 2000-500-250-10-250-2000 의 5 층짜리 딥구조를 사용하였다. 먼저 차원을 2000 차원에서부터 서서히 10 차원까지 축소한 후 후에 이를 다시 서서히 2000 차원으로 차원 확장하는 방법으로 오토인코딩 학습을 수행하였다. 학습을 마친 후 문서 쿼리에 대해 10 차원짜리 코드를 생성하고 코사인 유사도를 통해 코드가 유사한 문서들을 1~10000 개까지 검색해와서 카테고리의 정확도를 평가한 결과가 그림 5.16 에 표시되어 있다. 가장 유사한 소수의 문서를 검색해올수록 정확도가 높고 많은 수를 검색해올수록 정확도가 떨어진다. 즉, 부여된 코드가 카테고리 정보를 잘 반영하고 있음을 알 수 있다.

그림 5.17 은 코드의 시각화를 위해 2000-500-250-125-2 의 구조로 오토인코더를 학습하여 도시한 결과이다. 문서의 카테고리에 따라 코드가 잘 나누어져 있음을 확인할 수 있다.

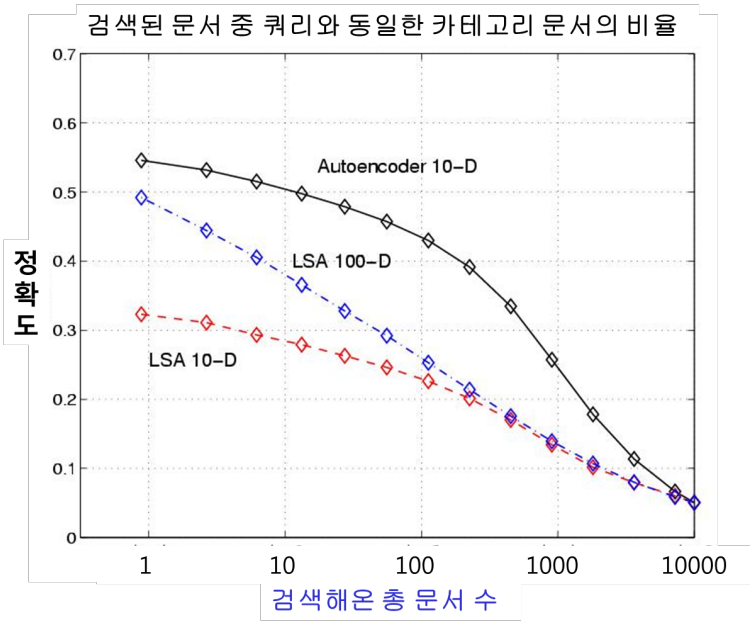

Figure 5.16 딥빌리프네트워크에서 얻은 코드를 이용한 문서 검색 성능

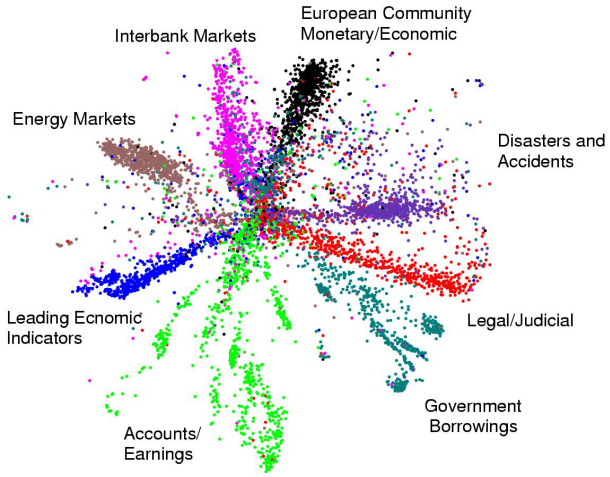

Figure 5.17 딥빌리프네트워크를 이용하여 문서에 코드부여 후 도시한 결과

참고문헌

[1] G. Hinton, and R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science*, vol.313.5786, pp. 504-507, 2006.
[2] G. Hinton, "Training Products of Experts by Minimizing Contrastive Divergence." *Neural Computation*. vol.14(8): pp. 1771-1800, 2002.
[3] I. Sutskever, T. Tieleman, "On the convergence properties of contrastive divergence," in *Proc. 13th Int'l Conf. on AI and Statistics* (AISTATS), 2010.
[4] G. Hinton, "A Practical Guide to Training Restricted Boltzmann Machines", Technical Report, 2010-003, University of Toronto.
[5] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets." *Neural Computation*. vol.18(7): pp. 1527-1554, 2006.
[6] G. Hinton, P. Dayan, Peter, B. Frey, R. Neal, "The wake-sleep algorithm for unsupervised neural networks." *Science*, vol.268(5214), pp. 1158-1161, 1995.
[7] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol.15(1), pp.1929-1958, 2014.