워크북

교과목명 : 머신 러닝

차시명: 5차시

◈ 담당교수: **장 필 훈**

⊙ 세부목차

- 네트워크 훈련
- 경사하강 최적화
- 오차역전파
- 정규화
- CNN

학습에 앞서

■ 학습개요

신경망 네트워크를 학습시키는 방법에 관해 이론적인 측면을 살펴보고, 선형회귀와 비교해본다. 이를 통해 문제의 종류와 오류함수에 따라 자연스러운 출력함수의 쌍이 있다는 것을 배운다.

매개변수의 최적화문제에서 가장 많이 쓰이는 방법인 경사하강 최적화의 수식을 이해하고 오차역전파의 각 단계를 자세히 이해한다. 우선 선형회귀를 예로 들어 계산해보고, 그 다음 신경망에서 오차역전파가 진행되는 과정을 자세히 살펴본다.

신경망에서도 정규화과정이 필요하므로, 어떤 종류가 있는지 대략 살펴보고 CNN에 대해 자세히 살펴본다.

그 외로 혼합밀도 네트워크, 베이지안 뉴럴 네트워크등 신경망에 관련된 문제들을 간략히 살펴본다.

■ 학습목표

1	학습의 종류에 따른 오류함수와 출력함수를 배운다.
2	신경망의 학습원리를 근본적으로 이해한다.
3	신경망의 한 종류인 CNN 의 구조를 배우고 장점을 배운다.
4	혼합밀도 네트워크, 베이지안 뉴럴 네트워크등 주변문제 관해 알아본다.

■ 주요용어

용어	해설
back propagation	신경망에서, 출력함수를 적절히 선택하면 각 노드의 가중치에 대한 출력값의 미분을 해석적으로 구할 수 있는데, 보통 그 입력, 출력에 모두 의존하게 된다. 따라서 오류함수의 기울기를 구할 때, 맨 마지막단부터 차례대로 거슬러 올라가면서 모든 은닉유닛의 매개변수에 대한 오류함수의 기울기를 구해내는데, feed forward network 의 일반적인 방향(forward)과 반대로(backward) 거슬러 올라가기 때문에 이렇게 부름. 신경망을 학습시키기 위한 알고리즘의 이름이라고 생각하면 됨.
CNN	convolutional neural network. 입력노드와 히든노드간의 연결이 convolution 연산을 기본으로 하도록 디자인된 네트워크. 출력층이 어럿으로 이루어져 feature 들을 나타낸다.(입력차원수와 무관하게 조절 가능하다.) 대략의 구조는 강의자료 참고
혼합밀도 네트워크	하나의 사전분포로 데이터집합을 잘 표현할 수 없거나, 비정형의 데이터분포를 나타내야 할 때, 여러 기저함수를 이용해 비선형함수를 근사해내듯이 여러 분포를 혼합하여 적절한 매개변수를 선택한 뒤 데이터분포를 추측하거나 묘사해낸다. 신경망의 경우 처음부터 베이지안으로 해결하기 어려운 분포를 근사해내기 위해 사용되므로, 혼합밀도 네트워크의 일종으로 보는 것도 가능하다.

학습하기

저번시간에 이어 다층퍼셉트론의 훈련과정을 더 자세히 살펴보겠습니다. 매개변수를 정하려면 목표(=기준)가 있어야 하고, 그것이 손실함수라는 점은 앞서 배웠습니다. 손실함수로 제곱오류가 가장 많이 쓰인다는 것도 익혔습니다. 이번에도 매개변수를 정하기 위해 제곱합오류를 가정하고 살펴보겠습니다.

이미 앞서 우리가 모두 살펴본것처럼, 제곱합오류함수와 가우시안 노이즈를 가정하면, 음의 로그를 최대화 하는 것으로 목표를 달성할 수 있습니다. 물론 오류함수의 최소화를 바로 할 수 있다면, 그렇

게 하면 됩니다.

분류가 아니라 회귀에도 신경망을 사용할 수 있는데, 출력함수를 따로 쓸 필요가 없다는 점을 제외하면 둘은 완전히 동일합니다.

그러면 이진분류의 경우를 살펴보겠습니다. 이진분류의 경우 출력함수로 시그모이드 함수를 주로 쓰는데 출력값 자체를 확률로 볼 수 있습니다. 이때 출력값의 조건부 분포도 앞서 살펴본 바와 같습니다. (우리가 이 값들의 곱으로 가능도 함수를 표현했었습니다)

$$p(t|\mathbf{x},\mathbf{w}) = y(\mathbf{x},\mathbf{w})^t\{1-y(\mathbf{x},\mathbf{w})\}^{1-t}$$

이 조건부 분포들의 곱(가능도 함수)에 음의 로그를 취해보겠습니다.

$$E(\mathbf{w}) = -\sum_{\{n=1\}}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$$

앞으로도 매우 자주 접하게 될 '교차(cross)엔트로피'가 바로 이것입니다. 분류문제에서는 이 교차엔트로피를 loss로 쓰는데, 제곱오류함수를 사용할때보다 훈련과정도 빠르고 일반화능력도 개선된다는 것이 알려져 있습니다. 어떻게 보면 자연스러운 결과일수도 있지만(회귀가 아니라 분류이므로 제곱오류함수가 부적절하게 보일 수도 있습니다) 또 어떤면으로는 상당히 놀라운 결과라고 생각할 수도 있습니다.

지금까지 배운 회귀나 분류에 따른 출력과 오류함수를 정리하면 다음 표와 같습니다.

	출력	오류함수
회귀	선형	제곱합오류함수
이진분류	로지스틱 시그모이드	교차엔트로피
다중클래스 분류	소프트맥스	다중클래스 교차엔트로피

(다중클래스 교차 엔트로피는 교차엔트로피의 합입니다. 식으로 나타내면 다음과 같습니다.

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\ln y_k(\mathbf{x_n},\mathbf{w})$$

<매개변수 최적화>

손실함수를 정의했으므로 분류든 회귀든 이제 손실함수를 최소화하면 됩니다. E(w)를 최소화하는 w는 앞서 우리가 계속 해왔듯이 dE(w)/dw=0인 w를 찾으면 됩니다.

신경망의 경우 이 지점이 여러군데 존재합니다. 이것은 비선형성 때문인데요, 이 비선형성이 해석적 해를 찾는것도 불가능에 가깝게 만듭니다. 그러면 어떻게 w를 구할 수 있을까요. 이런 문제는 '연속적 비선형함수의 최적화' 문제입니다. 개념적으로는 '조금씩 답에 근접하도록' w를 조정하는 방법을 씁니다. 다시말해, w의 단계적 변화를 정의하고 w를 구해냅니다. 식으로 보자면, $\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \Delta\mathbf{w}^{(\tau)}$로 나타낼 수 있고, $\Delta\mathbf{w}^{(\tau)}$를 $\Delta E(\mathbf{w})$를 통해 얻어내는 것입니다.

그러면 그 방법을 신경망에 적용하면 되겠습니다. 지역점에서 기울기의 변화를 조금씩 반영하는 방법으로 우리가 원하는 해에 근접하는 방법을 신경망에서는 '경사하강 최적화'라고 부릅니다. gradient

descent라는 용어가 더 자주/광범위하게 쓰입니다. 식으로 나타내면 다음과 같습니다.

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta \nabla E\left(\mathbf{w}^{(\tau)}\right)$$

여기서 eta는 학습률을 나타냅니다. 신경망은 비선형함수이기 때문에 오류를 한번에 반영했을 때, 최적점에 도달한다는 보장이 없습니다. 오히려 최적점을 지나쳐버릴 확률이 큽니다. 그래서 조금씩만 변화를 주면서 최적해에 도달합니다.(이렇게 해도 최적점에 도달한다는 보장은 없습니다만, 적어도 수렴성은 개선할 수 있고, 수렴만 성공적으로 되어도 상당히 좋은 성능을 관찰할 수 있습니다)

위의 식을 보면 오류함수에 따라 기존 w를 업데이트하는 방식입니다. 그런데 문제는 오류함수는 모든 데이터포인트에 대해 계산되어야 한다는 것입니다. 데이터의 양이 적을때는 그렇게 할 수 있지만 데이터의 양이 많아지면 절대 그렇게 할 수가 없습니다. 시간도 문제지만 계산하는 컴퓨터의 메모리도 문제가 됩니다. 이미지를 입력으로 받는 네트워크의 경우 GPU의 메모리를 모두 사용해도 1~2장밖에 입력을 받을 수 없는 경우가 많습니다. 그러면 훈련데이터가 만장만 되어도 이 모든 이미지에 대해 처리한 뒤 오류를 계산하고, 오류에 따라 w를 업데이트하는 과정 단 한번을 하는데도 상당한 시간이 소요됩니다. 네트워크가 수렴할때까지는 이 과정이 수백~수천번 반복되는것이 흔하므로 그렇게 수행하는것은 불가능에 가깝습니다. 신경망에 관련한 기초이론이 대부분 1980년대에 완성되었음에도 최근까지 거의 발전이 없었던것은 이러한 계산능력의 한계가 큰몫을 했습니다. 그나마 빨라졌기 때문에 지금의 딥러닝이 가능해졌지만, 아직도 데이터 전체를 한꺼번에 계산하기에는 턱없이 부족합니다. 따라서 온라인 학습법을 사용하게 됩니다. 데이터 하나 혹은 한세트(메모리가 허락하는 한도내에서)를 계산하고 바로 w를 업데이트 합니다. 이런 방식을 SGD(stochastic gradient descent)라고 합니다. 소수의 문헌들에서는 '배치방식'이라고 말하기도 합니다. 이렇게 온라인방식을 쓰면 장점이 우선 (1) 데이터상의 중복처리가 효율적이고 (2) 지역적 최솟값에서 탈출하기가 쉽습니다. (1)은 생각해보면, 매번 데이터 전체에 대해 계산하는 것보다 데이터 하나 내지 n개(배치사이즈=n)를 사용하는 것이 당연히 계산량이 적습니다. 그리고 데이터의 수가 많을 경우 이렇게 데이터를 한두번만 쭉 훑으면 거의 곧바로 수렴점에 도달하는 경우가 많습니다. 굳이 데이터 전체를 한 셋트로 보고 계산을 반복할 필요가 없다는 뜻이지요. (2)의 경우를 생각해보면, 신경망의 비선형성때문에 데이터셋 전체를 대상으로 최적화를 수행할 경우 손실이 지역적 최솟값에 한번 빠지면 빠져나오기가 매우 힘듭니다. 하지만 개별데이터의 경우 상대적으로 아주 쉽게 지역적 최솟값을 탈출할 수 있지요. 평균값을 변화시키는 것보다 작은 부분집합의 평균값을 변화시키는게 쉽지 않겠습니까. 데이터포인트 하나라면 더 말할것도 없지요. 이런경우 적절히 learning rate를 조절하면 더 좋은 결과로 수렴할 확률이 높아집니다.

<오차역전파>

그러면 이제 손실함수의 값을 이용해서 어떻게 다층으로 이루어진 신경망의 w를 업데이트 하는지 살펴보겠습니다. error back-propagation이라 불리는 이 과정은 기울기를 계산할 때 뒤의층(출력에 가까운층)의 값이 필요해서 거꾸로 진행하는것처럼 보이기 때문에 붙은 이름이고 기본적으로는 앞에 나온 다음의 식대로 계산하는 것이 전부입니다.

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta \nabla E\left(\mathbf{w}^{(\tau)}\right)$$

그러면 단순히 선형모델 $y_k = \sum_i w_{ki} x_i$을 가정하고 제곱오류함수 $E_n = \frac{1}{2}\sum_k (y_{nk} - t_{nk})^2$ 를 사용했을 때 계산을 예제로 살펴보겠습니다. 오류의 gradient를 계산해보면 다음과 같습니다.

$$\frac{\partial E_n}{\partial w_{ji}} = (y_{nj} - t_{nj}) x_{ni}$$

선형함수이므로 y를 w에 대해 미분하면 x가 나오기 때문에 위의 식으로 쓸 수 있습니다.

신경망의 경우 위의 계산을 층마다 계속 반복하면 됩니다. 자세한 과정은 강의중에 다루었으니 참고하세요. 결국 오차역전파 과정을 요약하면 (1)입력을 주고 모든 은닉유닛과 출력유닛의 값을 계산 한 후 (2) 모든 출력유닛의 오류를 계산합니다. (3) 그런데 중간층의 오류는 결국 마지막층의 오류를 알아야 계산할 수 있으므로 거슬러 올라가는 형태가 되고, (4) 모든 오류가 구해지면 각 유닛의 w를 업데이트합니다.

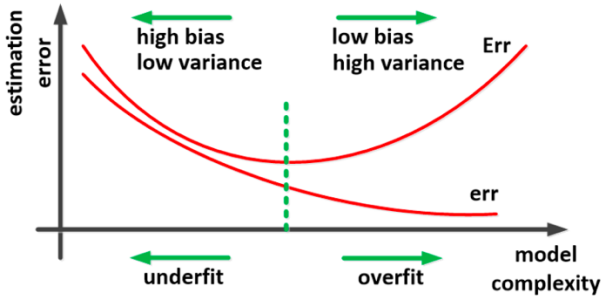
<정규화>

신경망에서 입력과 출력의 차원수는 데이터의 형태와 우리가 원하는 출력의 형태에 의해 정해집니다. 하지만 중간의 은닉층의 경우 정해진 룰이 없고 우리가 마음대로 정할 수 있습니다. 그래서 여러종류를 실험해보고 가장 좋은것을 선택할때가 많습니다. 그러면 신경망에서는 정규화 과정이 필요없는 것일까요. 우리가 앞서 살펴본 선형회귀의 경우에는 일단 차수를 적당히 높게 주고 w에 관한 정규화항을 손실함수에 넣습니다. 무조건 낮은 차수를 주지 않지요. 신경망에서도 정규화가 없다면 무조건 네트워크 구조를 단순하게 하는 방법밖에 없을 것입니다. 모델의 표현력 자체를 제한하는 것이죠. 하지만 그렇게 하지 않습니다. 신경망은 주로 dropout이나 weight decay를 씁니다.

dropout은 학습단계에서 랜덤하게 연결을 끊습니다. 그렇게 하면 모든 학습단계에서 일정량(보통 0.1~0.5정도를 잡습니다)의 w가 학습에서 배제되고 그만큼 모델의 표현력을 줄이는 역할을 합니다. 하지만 배제되는 w 가 계속 변하므로 전체적으로 정규화 효과를 줍니다. 오버 피팅을 막으면서도 모델의 표현력을 최대한 유지 시키는 역할을 하는 것이지요.

weight decay는 weight자체가 증가하는것을 막는 방법인데, 제곱정규화항의 경우 weight의 절대크기에 의존한다는 단점을 가집니다. 강의시간에 보여 드린 것처럼 신경망의 경우 weight 의 크기를 전체적으로 변화시킴으로써 동일한 결과를 얻어낼 수 있는데 그렇게 되면 가중치 감쇠가 의도한 대로 동작하지 않게 됩니다. 같은 결과를 내는 두 네트워크에서 서로 다른 가중치 감쇠가 이루어지게 되는 것이지요. 하지만 달리 마땅한 대안이 없으므로 제곱 정규항을 지금도 많이 사용하고 있습니다. (라이브러리등에서는 L2 penalty로 주로 표현합니다.) 다만, 이러한 단점이 존재한다는 것은 알고 있어야 합니다.

네트워크 복잡도를 조절 하기 위한 다른 방법으로 조기 종료가 있습니다. 이것은 본질적으로 네트워크의 표현력 자체를 떨어트린다기 보다 네트워크의 학습 진행 정도를 강제로 제한하여 오버피팅을 막는 방식입니다. 훈련집합 오류와 검증집합 오류의 오차를 보고 검증 집합 오류가 최소에 도달 하는 모델을 선택 합니다.



[Ghojogh, Benyamin, and Mark Crowley. "The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial." arXiv preprint arXiv:1905.12787 (2019).APA , Fig 6(a)]

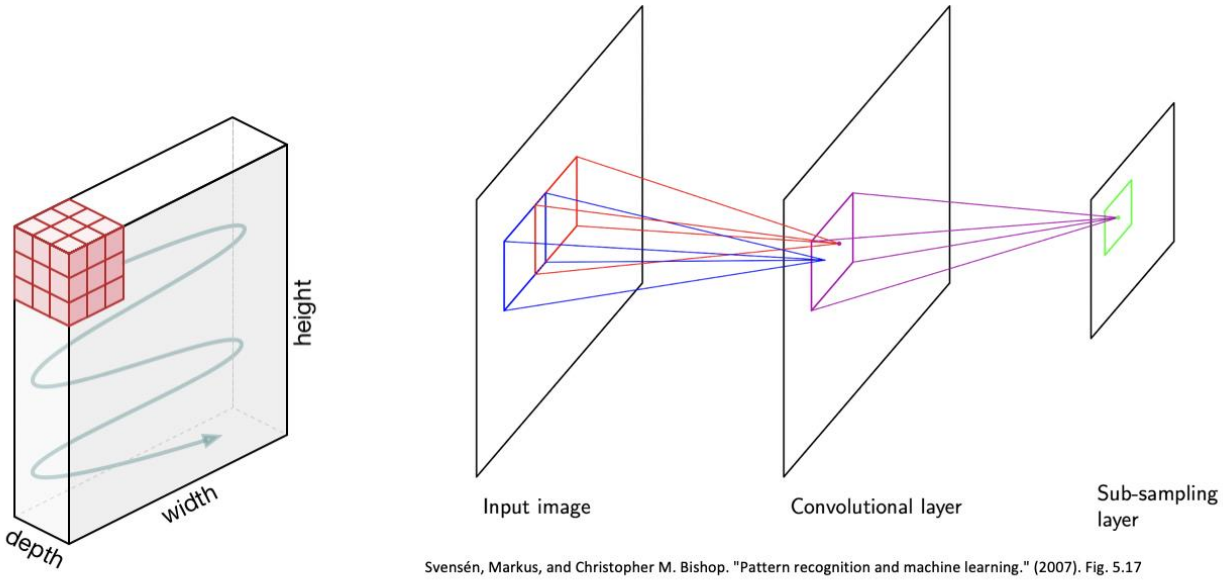
정규화 과정은 결국 오버 피팅을 막고 일반성을 획득하기 위한 과정입니다. 그래서 불변성과 필연적으로 연결 됩니다. 불변성이란 입력에서 일정한 변환이 주어져도 안정적인 결과를 내는 성질을 말합니다. 예를 들어 이미지 안에 글씨 위치가 변한다고 해도 글씨를 검출하는 모델은 같은 결과를 주어야 할 것입니다. 크기가 변한다거나 모양이 약간 변할 때도 마찬가지입니다. 이런 이동불변성, 크기불변성등을 모델에 학습시키는 시키는 방법은 여러 가지가 있습니다만, 대부분 상식적으로 생각할 수 있는 것들입니다. 아예 학습 시점부터 데이터를 이리 저리 변환해서 입력을 준다던가 입력이 약간 변화했을때 출력이 바뀌면 패널티를 준다든가 할 수 있습니다. 하지만 이런 직관적인 방법이 이외에 신경망 자체에 불변성을 포함 할 수 있는데 그것이 바로 CNN입니다.

<CNN>

Convolutional Neural Net. CNN은 처음에 object detection을 위해 개발되었습니다. 이미지에 경우 입력의 차원이 매우 크기 때문에 네트워크를 최대한 단순화 할 필요가 있는데 작은 크기의 필터들을 통해 단계적으로 이미지의 특성을 추출했을때 매우 강력한 추상화능력을 가지는 것을 관찰할 수 있습니다. 다시 말해서 이미지 크기에 비해서 작은 크기의 이미지패치들만을 관찰하는 '작은 네트워크'를 여러층 쌓으면 우리가 원하는 결과를 얻을 수 있습니다. CNN의 구조는 다음페이지의 그림과 같습니다. 하나의 필터가 이미지 전체를 스캔하므로 우리가 앞서 이야기했던 불변성도 자연스럽게 획득됨을 볼 수 있습니다.

<혼합밀도 네트워크>

지금까지는 조건부 분포를 모델링 할 때, 노이즈에 대한 분포로 가우시안을 가정했습니다. 하지만 가우시안이 아니면 어떻게 할까요. 이럴때는 근사하는 방법 밖에 없습니다. 우리가 알고 있는 분포 여러개의 합으로 근사해서 추정하는 방법을 씁니다.



Svensén, Markus, and Christopher M. Bishop. "Pattern recognition and machine learning." (2007). Fig. 5.17

Sumit Saha, 「A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way」,
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

	연습문제

1. (OX문제) 회귀문제에서 출력함수는 tanh함수를 쓴다

   X 회귀문제의 네트워크는 출력함수가 따로 없다.

2. 이진분류문제에서 출력함수를 시그모이드 함수로 쓰면, 출력값이 바로 확률값이라고 가정할 수 있다.

   O 출력값 자체를 확률로 간주할 수 있다.

3. 분류에서는 제곱오류함수보다 크로스엔트로피 오류함수를 쓰는 것이 좋다.

   O 훈련과정도 더 빠르고 일반화도 개선된다고 알려져 있다.

4. 다중클래스분류의 출력이 소프트맥스함수면, 오류함수는 주로 다중클래스 교차 엔트로피함수를 쓴다.

   O 출력과 쌍으로 음의 로그함수를 쓰는데 소프트맥스의 음의 로그는 다중클래스 교차 엔트로피 함수이다.

5. 신경망의 경우에도 일반적으로 해석적인 해를 찾아서 오류함수를 최소화할 수 있다.

   X 비선형성 때문에 해석적인 해를 찾기가 극히 어렵거나 불가능해진다.

6. 신경망의 경우, 매개변수를 근사하는 방법으로 주로 경사하강법을 쓴다.

O 해석적인 해를 찾기가 어려우므로 기울기정보를 사용하여 조금씩 근사해 나가는 방법을 쓴다.

7. 온라인배치방법을 쓰면 지역적 최솟값에서 탈출하기가 쉽다

   O 전체데이터의 오류함수 임계점과 개별 포인트에서 오류함수 임계점은 다르기 쉽기 때문에 탈출하기 쉽다.

8. 신경망의 경우, 정규화를 위해 조기종료나 dropout등의 방법을 쓴다.

   O 맞는 설명. 네트워크 구조 자체를 단순화해서 쓰기도 한다(예:CNN)

9. CNN은 이동불변성, 크기불변성이 다른 학습방법들에 비해 상대적으로 좋다.

   O 맞는 설명

정리하기

1. 앞선 회귀문제에서는 음의 로그최대화를 통해 매개변수를 결정했다. 뉴럴넷은 보통 오류함수를 최소화한다. 둘의 본질적인 차이는 없다.
2. 오류함수와 활성화 함수 사이에는 짝이 있다. 제곱합 오류함수의 경우에는 항등함수를 쓴다.
3. 이진분류문제에서 출력을 시그모이드 함수로 가정하면 결과값을 확률로 바로 해석할 수 있다.
4. 이진분류문제에서 조건부 분포에 음의 로그를 취한 오류함수는 교차 엔트로피 오류함수이다.
5. 교체 엔트로피오류함수는 제곱오류함수보다 훈련과정이 더 빠르고, 일반화도 개선된다고 알려져 있다.
6. 비선형함수의 경우 해석적으로 미분값이 0이 되는 지점을 찾기 어렵기 때문에, 보통 경사하강법을 이용한다.
7. 배치방식으로 를 계산하려면 데이터 집합 전체를 이용해야 하는 것이 원칙이다.
8. 전체집합을 이용하기 힘든 경우가 대부분이므로 stochastic gradient descent이용한다.
9. 오차역전파는 오류함수의 미분을 계산하고, 결과를 바탕으로 가중치 w를 조절한다.
10. 다층으로 이루어져 있을 때는, 출력단부터 차례로 계산해서 결국 모든 유닛의 w를 얻어낸다.
11. 뉴럴넷에서 입력과 출력의 차원은 데이터의 형태에 의해 정해지고, 중간 은닉유닛은 자유롭게 디자인할 수 있다.
12. 뉴럴넷도 과적합을 막기 위해 정규화 한다. 조기종료, 네트워크 구조 단순화등을 한다.
13. 구조를 단순화 하면서 효과적으로 과적합을 제어하고, 불변성까지 획득하는 대표적인 구조가 CNN이다.
14. 다루려고 하는 조건부 분포가 가우시안이 아닐 때는 혼합밀도 네트워크(가우시언이나 다른 분포들의 혼합)을 사용하여 근사한다.

참고하기

Bishop, C. M. "Bishop-Pattern Recognition and Machine Learning-Springer 2006." Antimicrob. Agents Chemother (2014): 03728-14.

다음 차시 예고

- 커널방법론
- SVM