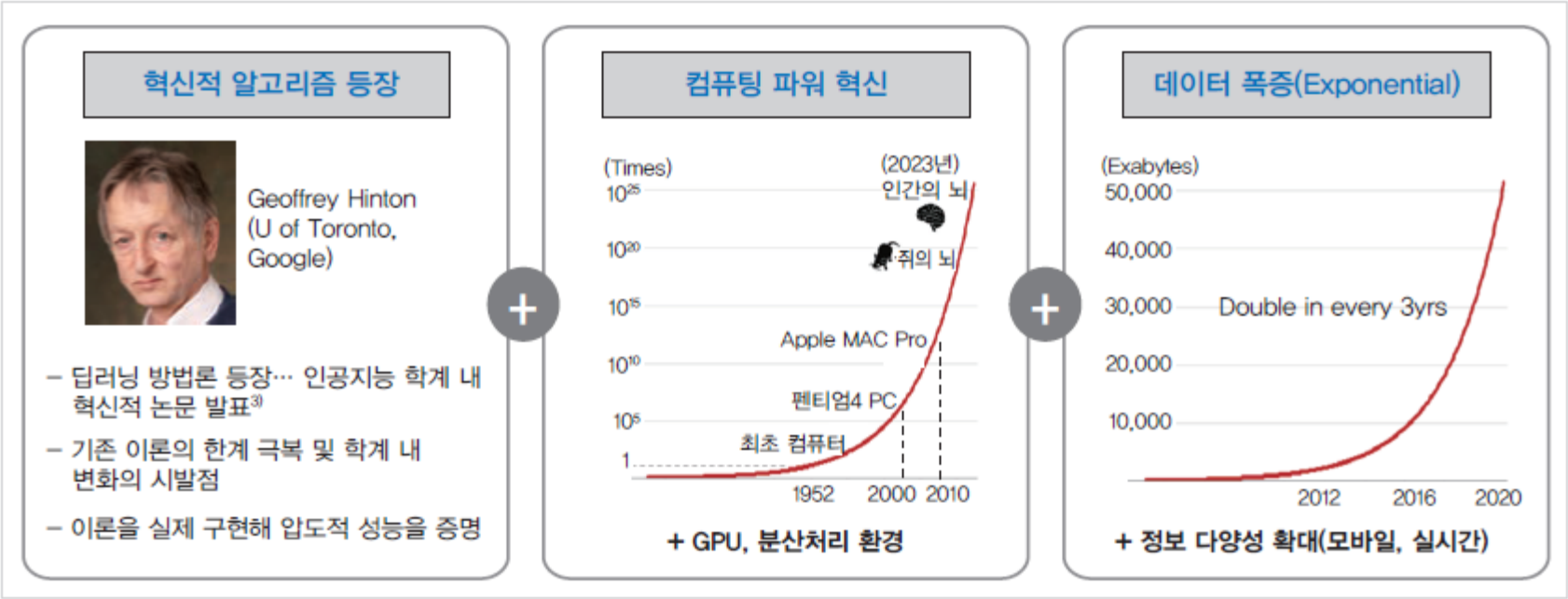
인 공 지 능

[5. 딥러닝 최적화]

소프트웨어융합대학
소프트웨어학부

지난 장에는 뭐했지?

4.1 딥러닝 성공 요인


혁신적 알고리즘 등장
Geoffrey Hinton
(U of Toronto,
Google)
– 딥러닝 방법론 등장… 인공지능 학계 내 혁신적 논문 발표[3]
– 기존 이론의 한계 극복 및 학계 내 변화의 시발점
– 이론을 실제 구현해 압도적 성능을 증명
+
컴퓨팅 파워 혁신
(Times)
(2023년)
인간의 뇌
쥐의 뇌
Apple MAC Pro
펜티엄4 PC
최초 컴퓨터
1952
2000
2010
+ GPU, 분산처리 환경
+
데이터 폭증(Exponential)
(Exabytes)
50,000
40,000
30,000
20,000
10,000
Double in every 3yrs
2012
2016
2020
+ 정보 다양성 확대(모바일, 실시간)

4.2 깊은 다층 퍼셉트론

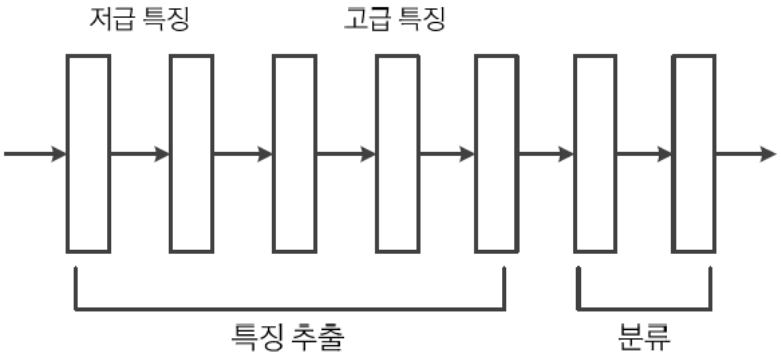


그림 4-2 깊은 신경망의 처리 절차

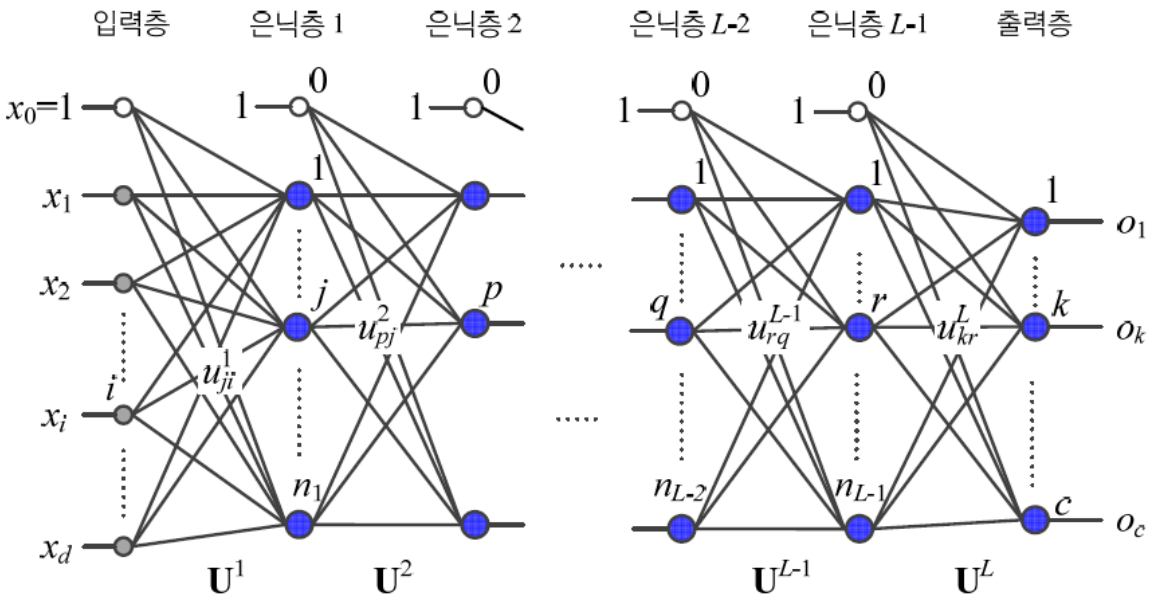
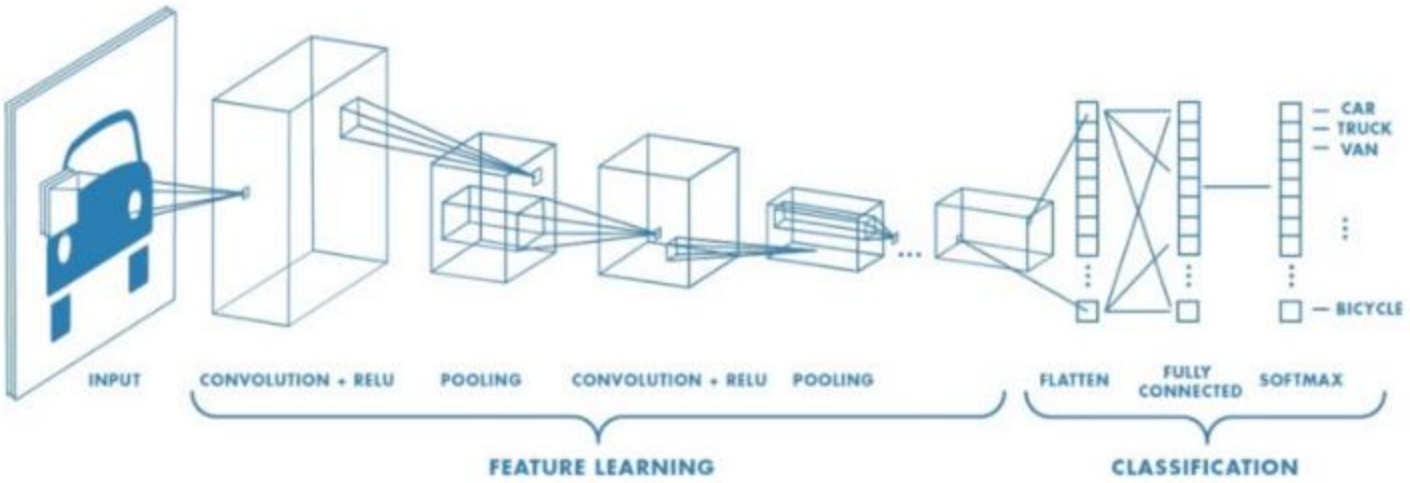


그림 4-3 깊은 MLP(DMLP)의 구조

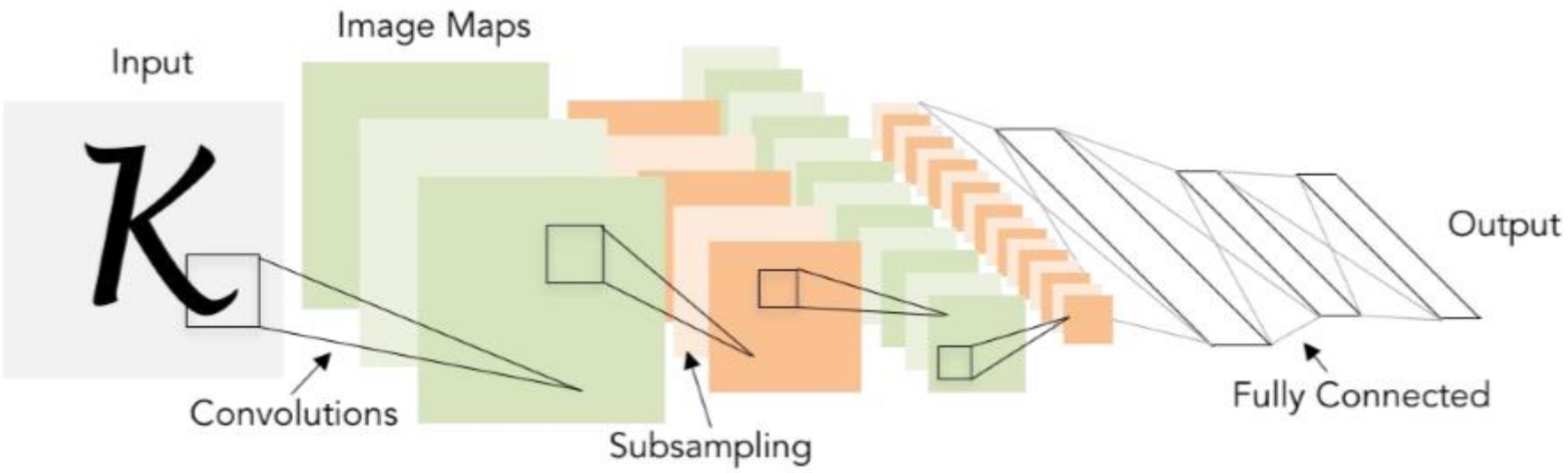
4.3 컨볼루션 신경망

- 전체 구조



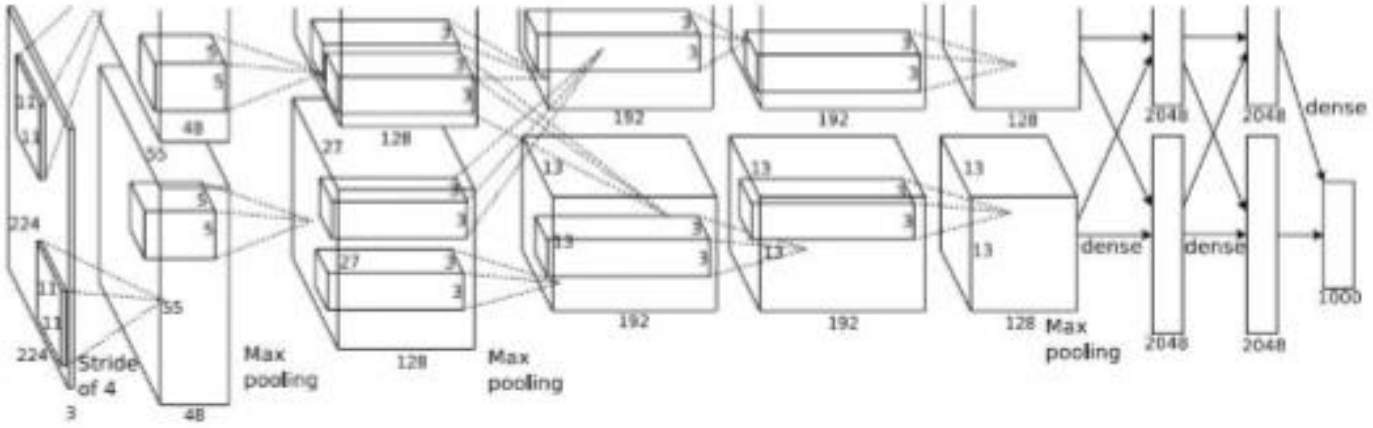
4.4 컨볼루션 신경망의 발전

- LeNet-5



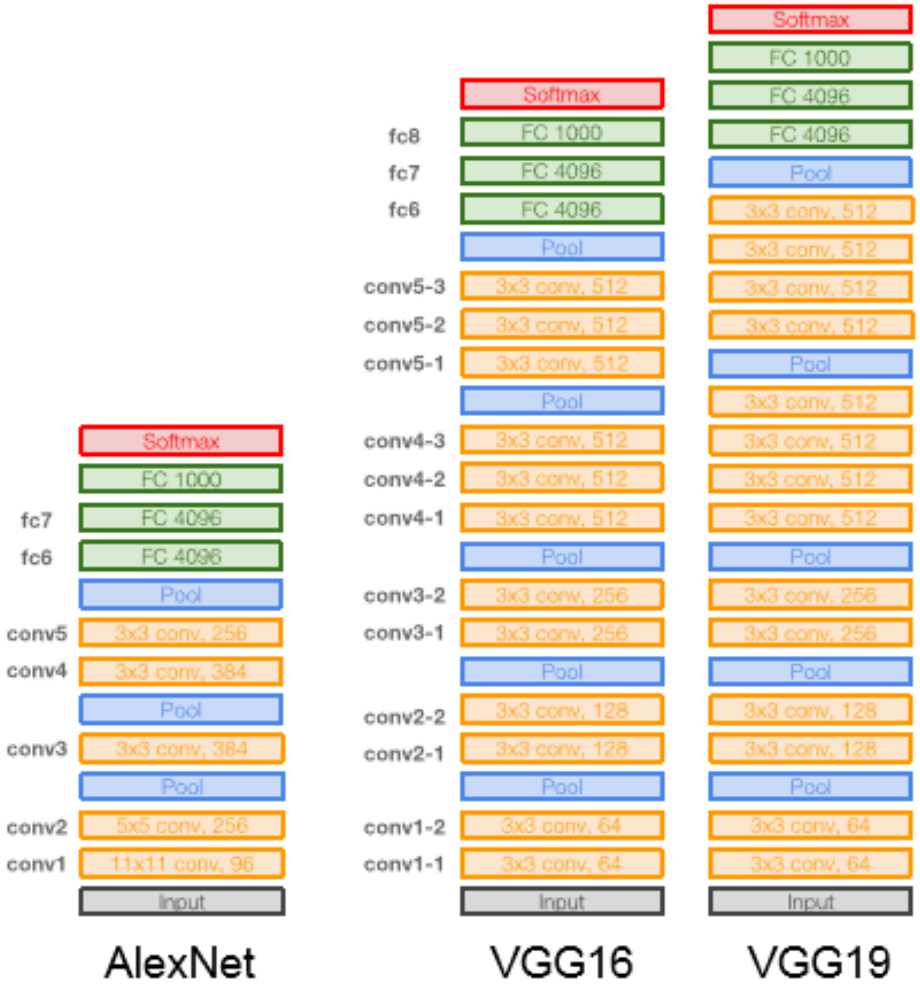
4.4 컨볼루션 신경망의 발전

- AlexNet



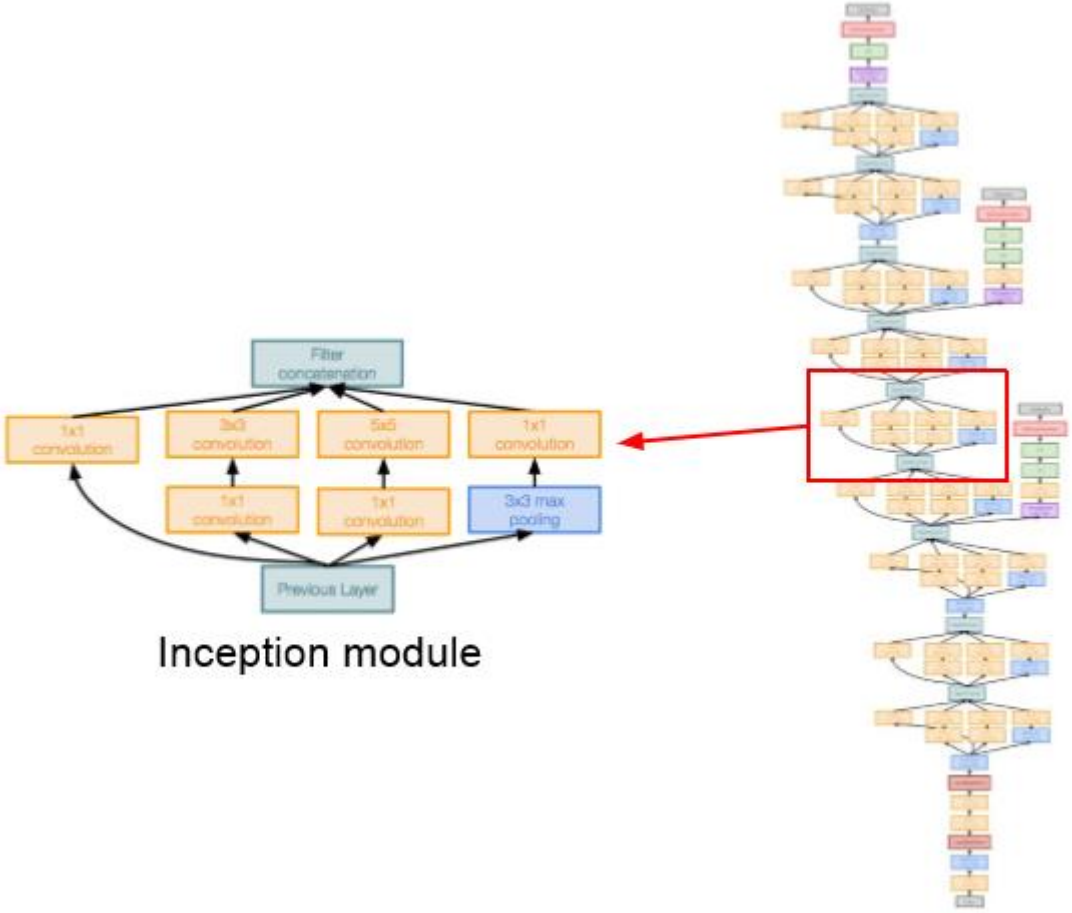
4.4 컨볼루션 신경망의 발전

- VGGNet



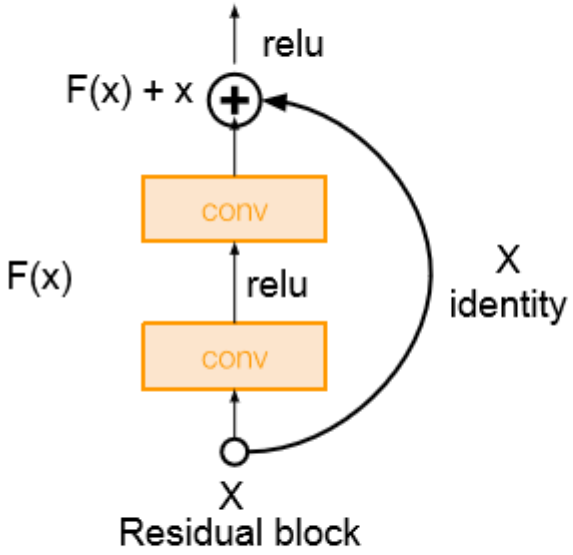
4.4 컨볼루션 신경망의 발전

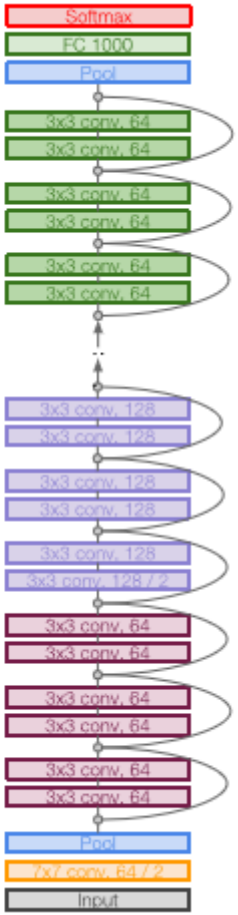
- GoogleNet



4.4 컨볼루션 신경망의 발전

- ResNet





4.6 딥러닝의 장점

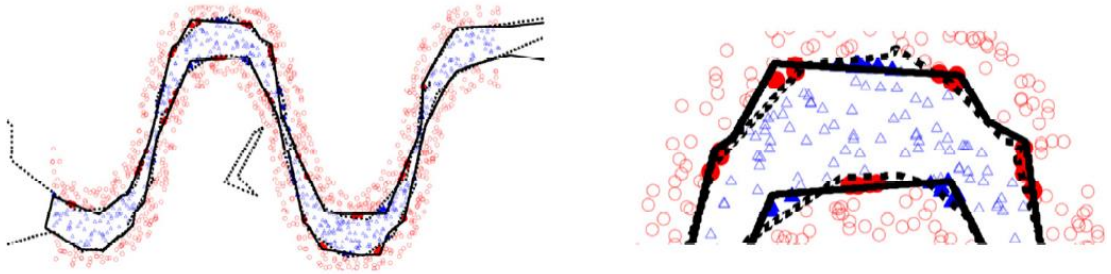


그림 4-38 은닉층의 개수가 늘어남에 따른 표현력 증가

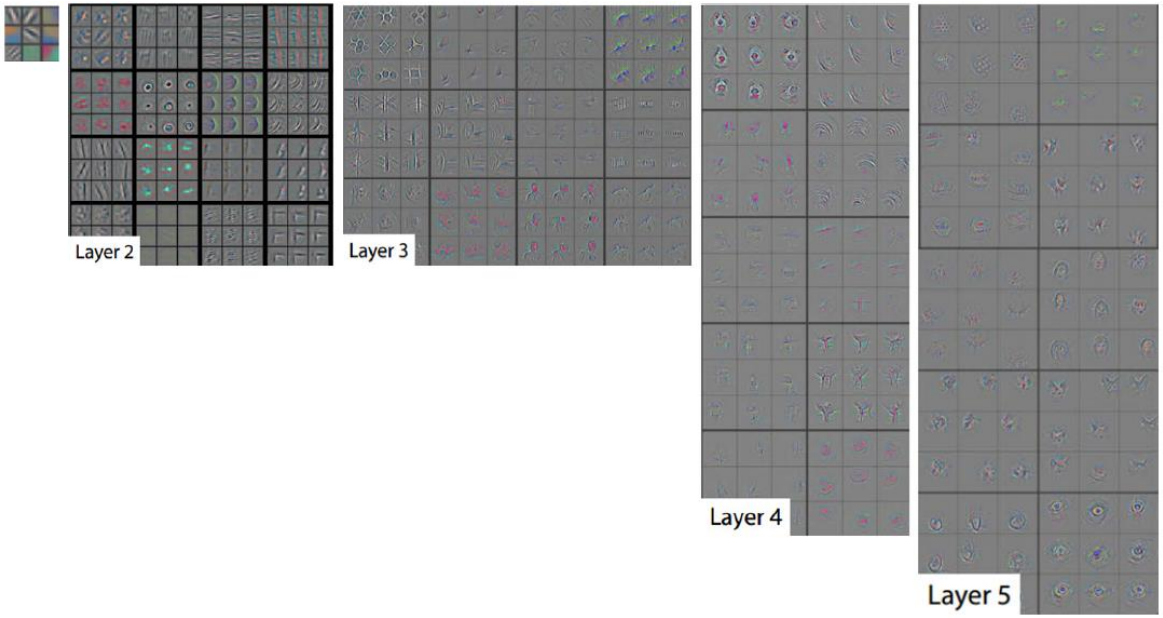


그림 4-40 CNN의 계층적 특징 추출

오늘 수업에는 뭐하지?

PREVIEW

- 과학과 공학에서 최적화
  - 예) 우주선의 최적궤도, 운영체제의 작업 할당 계획 등
- 기계 학습의 최적화도 매우 복잡함
  - 훈련집합으로 학습을 마친 후, 현장에서 발생하는 새로운unknown 샘플을 잘 예측해야 함

    → 즉 일반화generalization 능력이 좋아야 함
    - 훈련집합은 전체 데이터 (실제, 알 수 없음)의 대리자 역할
    - 검증집합은 테스트집합의 대리자 역할
    - 오차 제곱의 평균mean square error(MSE), 로그우도log-likelihood와 같은 목적함수도 궁극 목표인 정확률 (=판단 기준)의 대리자 역할을 함
- 기계 학습의 최적화가 어려운 이유
  - 대리자 관계
  - 목적함수의 비볼록non-covex 성질, 고차원 특징 공간, 데이터의 희소성 등
  - 긴 시간 소요
- 5장은 최적화의 어려움을 극복하는 여러 가지 효과적인 방안을 제시함

각 절에서 다루는 내용

5.1절_ 목적함수로서 평균제곱 오차의 단점을 지적하고, 딥러닝이 많이 활용하는 교차 엔트로피와 로그우도를 소개한다.

5.2절_ 스토캐스틱 경사 하강법의 성능 향상에 효과적인 전처리, 가중치 초기화, 모멘텀, 적응적 학습률, 활성함수, 배치 정규화를 설명한다.

5.3절_ 딥러닝에서 규제의 필요성과 원리를 간략히 기술한다.

5.4절_ 많이 쓰는 규제 기법으로서 가중치 벌칙, 조기 멈춤, 데이터 확대, 드롭아웃, 앙상블을 설명한다.

5.5절_ 하이퍼 매개변수의 중요성과 최적값을 찾는 방법을 소개한다.

5.6절_ 2차 미분 정보를 활용하여 스토캐스틱 경사 하강법을 보완하는 기법을 기술한다.

5.1 목적함수: 교차 엔트로피와 로그우도

- 5.1.1 평균제곱 오차를 다시 생각하기
- 5.1.2 교차 엔트로피 목적함수
- 5.1.3 소프트맥스softmax 함수와 로그우도 목적함수

시험에서는 틀린 만큼 합당한 벌점을 받는 것이 중요하다. 그래야 다음 시험에서 심기일전으로 공부하여 틀리는 개수를 줄일 가능성이 크기 때문이다. 틀린 개수에 상관없이 비슷한 벌점을 받는다면 나태해져 성적을 올리는 데 지연이 발생할 것이다. 이러한 원리가 기계 학습에도 적용될까?

학습 과정의 판정기준 중요!!

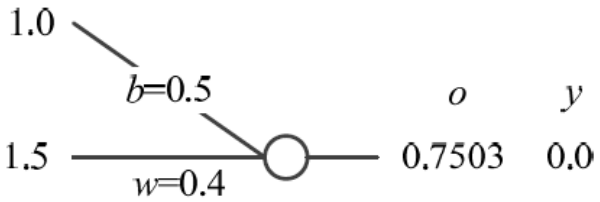
5.1.1 평균제곱 오차 다시 생각하기

- 평균제곱 오차(MSE) 목적함수

$$e = \frac{1}{2}\|\mathbf{y} - \mathbf{o}\|_2^2 \qquad (5.1)$$

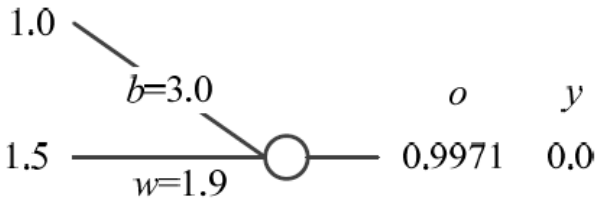
  - 오차가 클수록 e값이 크므로 벌점으로 훌륭함

- 하지만, 큰 허점이 존재
  - 왼쪽 상황은 $e = 0.2815$,
  - 오른쪽 상황은 $e = 0.4971$이므로 오른쪽이 더 큰 벌점을 받아야 마땅함



$$\frac{\partial e}{\partial w} = 0.2109$$

$$\frac{\partial e}{\partial b} = 0.1406$$



$$\frac{\partial e}{\partial w} = 0.0043$$

$$\frac{\partial e}{\partial b} = 0.0029$$

그림 5-1 MSE가 목적함수로서 부적절한 상황

신경망 학습 과정에서
학습은 오류를 줄이는 방향으로 가중치와 바이어스를 교정
← 큰 교정 (벌점)이 필요함에도 작은 그레이디언트가 적용됨

5.1.1 평균제곱 오차 다시 생각하기

큰 허점

- 식 (5.3)의 그레이디언트가 벌점에 해당

$$e = \frac{1}{2}(y - o)^2 = \frac{1}{2}(y - \sigma(wx + b))^2 \tag{5.2}$$

$$\left.\begin{aligned} \frac{\partial e}{\partial w} &= -(y - o)x\sigma'(wx + b) \\ \frac{\partial e}{\partial b} &= -(y - o)\sigma'(wx + b) \end{aligned}\right\} \tag{5.3}$$

- 그레이디언트를 계산해보면 왼쪽 상황의 그레이디언트가 더 큼

  → 더 많은 오류를 범한 상황이 더 낮은 벌점을 받은 꼴

  → 학습이 더딘 부정적 효과

이유

- $wx + b$ (아래 그래프의 가로축에 해당)가 커지면 그레이디언트가 작아짐

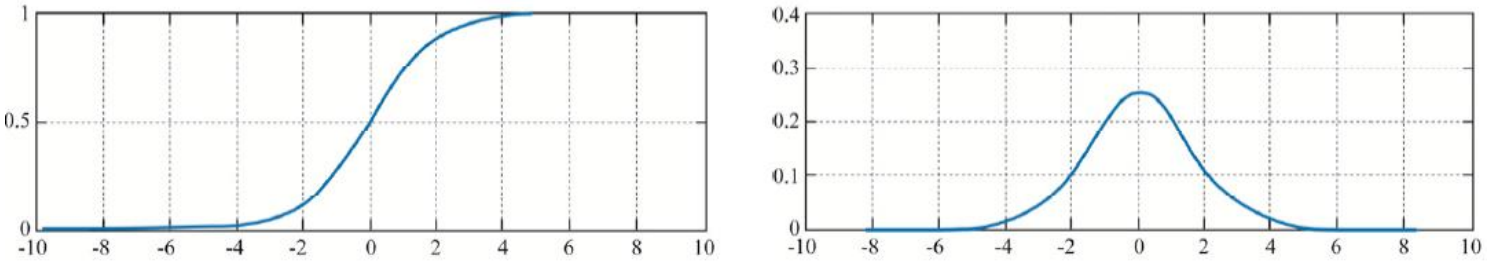


그림 5-2 로지스틱 시그모이드함수와 도함수

5.1.1 평균제곱 오차 다시 생각하기

(참고) 제곱 오차의 편향-분산 분해

- 훈련집합 $x_1, \ldots, x_n$와 각 x_i에 대응되는 y_i의 쌍으로 표현되는 데이터 집합
  - 여기에 잡음이 존재하는 함수적 관계가 있다고 가정 $y_i = f(x_i) + \epsilon$
  - 잡음 ϵ은 평균이 0이고, 분산이 σ^2인 정규분포
- 학습 알고리즘을 통해 실제 관계함수인 $y = f(x)$를 최대한 잘 근사한 근사함수 $\hat{f}(x)$ 찾음
  여기서 정량적인 판정기준으로 평균제곱 오차를 사용한 경우
- 선택된 모델 $\hat{f}$의 표본 x에 대한 기대 오차는 다음과 같이 편향-분산 분해할 수 있음

$$E\left[\left(y - \hat{f}(x)\right)^2\right] = Bias\left[\hat{f}(x)\right]^2 + Var\left[\hat{f}(x)\right] + \sigma^2$$

$Bias\left[\hat{f}(x)\right] = E[\hat{f}(x) - f(x)]$이고, $Var\left[\hat{f}(x)\right] = E[\left(\hat{f}(x) - E\left[\hat{f}(x)\right]\right)^2]$임

  - 편향 제곱값은 모델을 간소화 (낮은 용량, 비선형인데 선형 모델 적용하는 경우)에 의함
    즉, 편향값이 존재하면 $\hat{f}(x)$ 선택의 가정에 오류 존재함을 의미
  - 분산값은 $\hat{f}(x)$가 그 평균 근처에서 변동하는 폭의 정도
  - σ^2 줄일 수 없는 오차를 의미, 보이지 않는 표본들의 기대 오차의 하한값 역할

→ $\hat{f}(x)$가 복잡할수록

더 많은 데이터를 맞출 수 있어서 편향값이 작아지지만

모델이 각 점을 맞추기 위해 더 많이 변위함 때문에 분산값은 커짐

5.1.2 교차 엔트로피 목적함수

교차 엔트로피cross entropy

- 레이블label에 해당하는 y가 확률변수 (부류가 2개라고 가정하면 $y \in \{0,1\}$)
- 확률 분포: P는 정답 레이블, Q는 신경망 출력

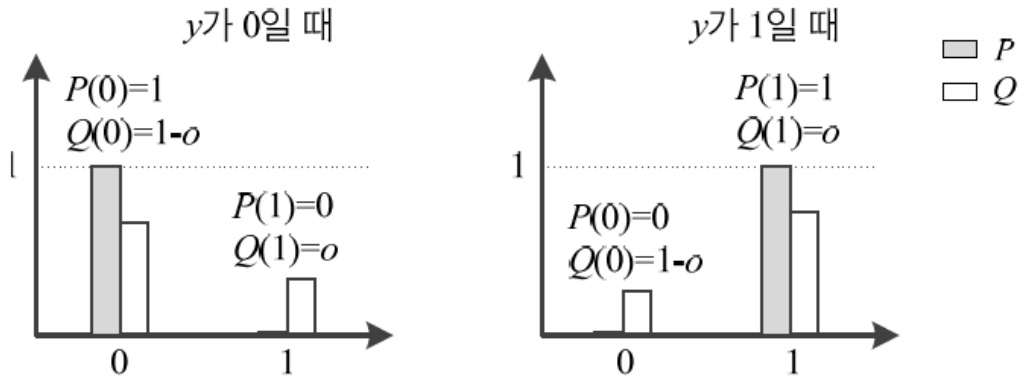


그림 5-3 레이블 y가 0일 때와 1일 때의 P와 Q의 확률분포

- 확률분포를 통일된 수식으로 쓰면,

$$P(0) = 1 - y \quad Q(0) = 1 - o$$
$$P(1) = y \quad Q(1) = o$$

신경망이기 때문에 $o = \sigma(z)$이고 $z = wx + b$

- 식 (2.47) 교차 엔트로피 $H(P,Q) = -\sum_x P(x)\log_2 Q(x) = -\sum_{i=1,k} P(e_i)\log_2 Q(e_i)$을 적용

$$\rightarrow H(P,Q) = -\sum_{y\in\{0,1\}} P(y)\log_2 Q(y)$$

5.1.2 교차 엔트로피 목적함수

- 손쉬운 예

$$P(0) = 1 - y \quad Q(0) = 1 - o$$
$$P(1) = y \qquad Q(1) = o$$

- $Q(o) = o$, $o \in \{0,1\}$으로 단순화하고
- 교차 엔트로피 $H(P,Q) = \sum P(x) log Q(x)$를 구하면

학습이 잘 안되어서 오분류된 손실은

$$-[1\ 0]\begin{bmatrix}\log 0\\ \log 1\end{bmatrix} = -(-\infty + 0) = \infty$$

학습이 잘 되어 제대로 분류된 손실은

$$-[1\ 0]\begin{bmatrix}\log 1\\ \log 0\end{bmatrix} = -(0 + 0) = 0$$

5.1.2 교차 엔트로피 목적함수

교차 엔트로피 목적함수

$$e = -(y \log_2 o + (1-y)\log_2(1-o)), \quad \text{이때, } o = \sigma(z) \text{이고} z = wx + b \qquad (5.4)$$

- 잘 역할을 수행하는지 확인
  - y가 1, o가 0.98일 때 (예측이 잘된 경우)
    - 오류 $e = -(1 \log_2 0.98 + (1-1)\log_2(1-0.98)) = 0.0291$로서 낮은 값
  - y가 1, o가 0.0001일 때 (예측이 잘못된 경우, 혹은 오분류된 경우)
    - 오류 $e = -(1 \log_2 0.0001 + (1-1)\log_2(1-0.0001)) = 13.2877$로서 높은 값

5.1.2 교차 엔트로피 목적함수

- 공정한 벌점을 부여하는지 확인 (MSE의 느린 학습 문제를 해결 확인)
  - 도함수를 구하면,

$$\frac{\partial e}{\partial w} = -\left(\frac{y}{o} - \frac{1-y}{1-o}\right)\frac{\partial o}{\partial w}$$

식 (5.4) 대입, 연쇄법칙 적용
(log f(x))′=f′(x)/f(x)

$$= -\left(\frac{y}{o} - \frac{1-y}{1-o}\right) x\sigma'(z)$$

$$\sigma'(z) = \sigma(z)\big(1-\sigma(z)\big) = o(1-o)$$

$$= -x\left(\frac{y}{o} - \frac{1-y}{1-o}\right) o(1-o)$$

$$= x(o-y)$$

$$\longrightarrow \quad \left.\begin{aligned}\frac{\partial e}{\partial w} &= x(o-y)\\ \frac{\partial e}{\partial b} &= (o-y)\end{aligned}\right\} \qquad (5.5)$$

  - 그레이디언트를 계산해 보면, 오류가 더 큰 오른쪽에 더 큰 벌점 (그레이디언트) 부과

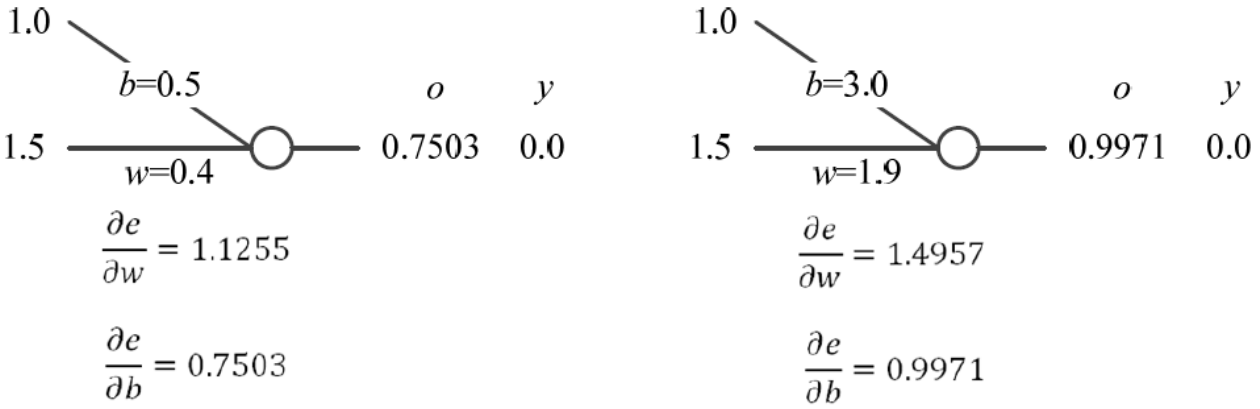


그림 5-4 교차 엔트로피를 목적함수로 사용하여 느린 학습 문제를 해결

5.1.2 교차 엔트로피 목적함수

- 식 (5.4)를 c개의 출력 노드를 가진 경우로 확장
  - 출력 벡터 $\mathbf{o} = (o_1, o_2, \cdots, o_c)^{\mathrm{T}}$인 상황으로 확장 ([그림 4-3]의 DMLP)

$$e = -\sum_{i=1,c} (y_i \log_2 o_i + (1 - y_i) \log_2 (1 - o_i)) \tag{5.6}$$

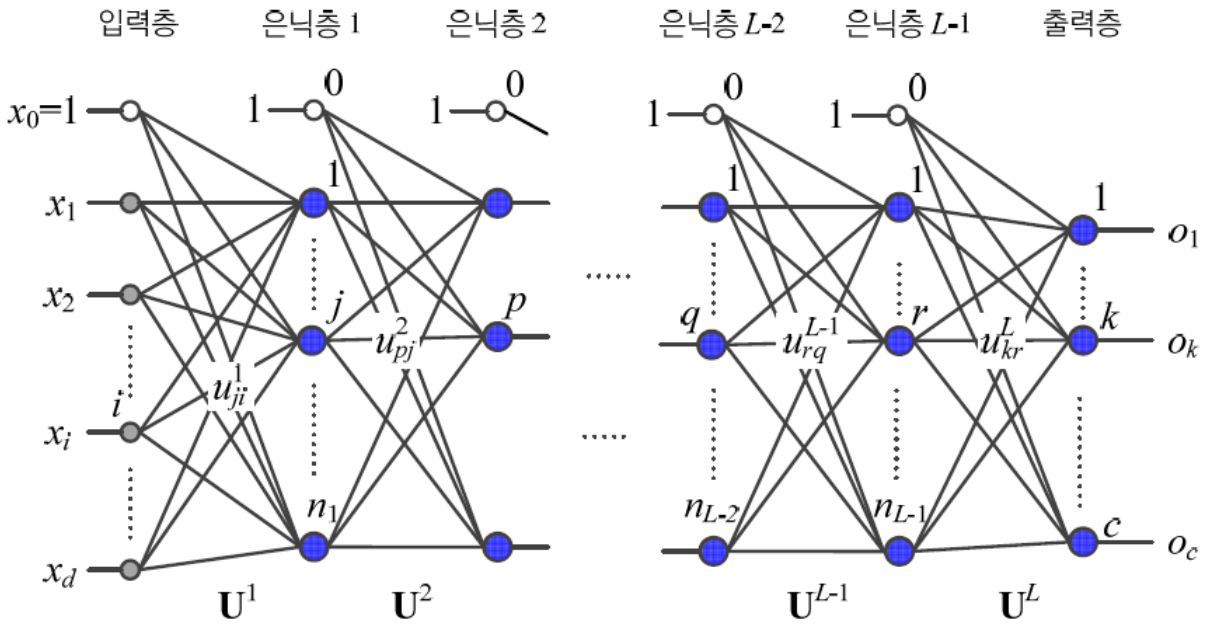


그림 4-3 깊은 MLP(DMLP)의 구조

5.1.3 소프트맥스 활성함수와 로그우도 목적함수

소프트맥스softmax 함수

$$o_j = \frac{e^{s_j}}{\sum_{i=1,c} e^{s_i}} \qquad (5.7)$$

- 동작 예시
  - 소프트맥스는 최대max를 모방

    출력 노드의 중간 계산 결과 s_i^L의 최댓값을 더욱 활성화하고 다른 작은 값들은 억제
  - 모두 더하면 1이 되어 확률 모방

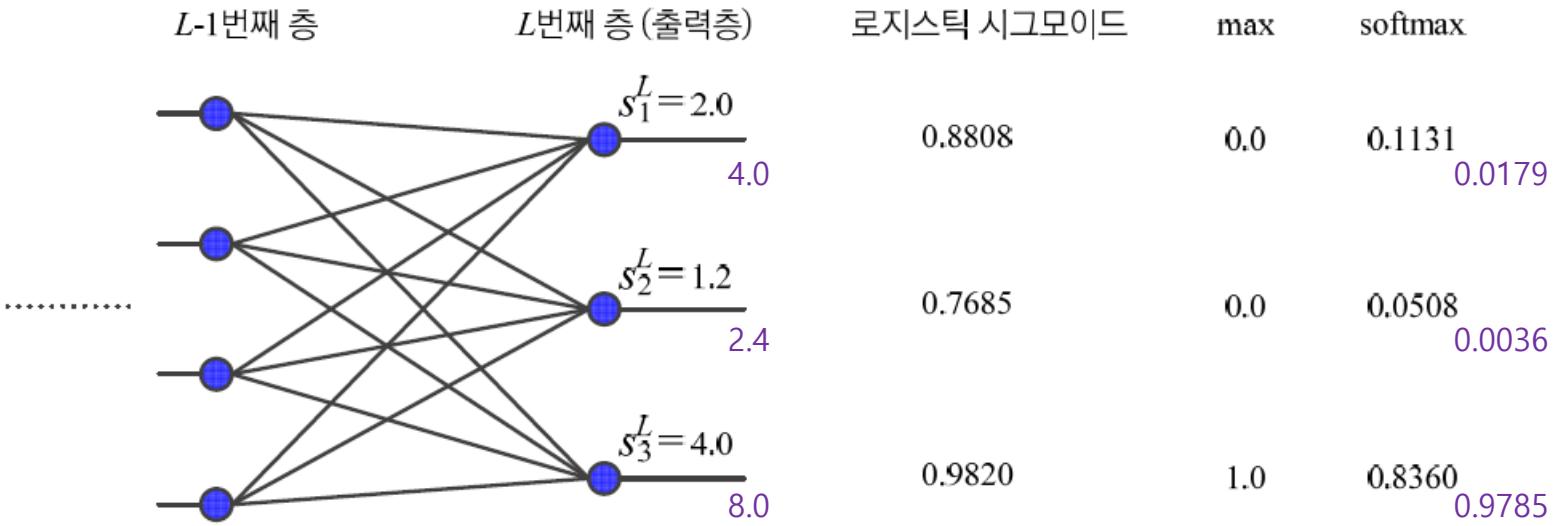


그림 5-5 출력층의 활성함수로 로지스틱 시그모이드와 softmax 비교

출력층의 변화에 따라
소프트맥스의 결과도 맥스 출력에 더 가까워짐
따라서 부드러운 맥스 함수로 불림

5.1.3 소프트맥스 활성함수와 로그우도 목적함수

로그우도log-likelihood 목적함수

$$e = -\log_2 o_y \qquad (5.8)$$

- 모든 출력 노드값을 사용하는 MSE나 교차 엔트로피와 달리 o_y라는 하나의 노드만 적용
- o_y는 샘플의 레이블에 해당하는 노드의 출력값
  - 동작 예시1) [그림 5-5]에서 현재 샘플이 두 번째 부류라면 o_y는 o_2
    $e = -\log_2 0.0508 = 4.2990$. 잘못 분류한 셈이므로 목적함수값이 큼
  - 동작 예시2) [그림 5-5]에서 현재 샘플이 세 번째 부류라면 o_y는 o_3
    $e = -\log_2 0.8360 = 0.2584$. 제대로 분류한 셈이므로 목적함수값이 작음

소프트맥스와 로그우도

- 소프트맥스는 최댓값이 아닌 값을 억제하여 0에 가깝게 만든다는 의도 내포
- 학습 샘플이 알려주는 부류에 해당하는 노드만 보겠다는 로그우도와 잘 어울림
- 따라서 둘을 결합하여 사용하는 경우가 많음

5.1.3 소프트맥스 활성함수와 로그우도 목적함수

- 소프트맥스 분류기softmax classifier
  - 다항 로지스틱 회귀 분석multinomial logistic regression의 예
  - 분류기의 최종 값을 확률로 표현
  - 소프트맥스와 로그우도 목적함수

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

$$L_i = -\log P(Y = y_i|X = x_i)$$

Probabilities must be >= 0

Probabilities must sum to 1

	Unnormalized log-probabilities / logits	exp	unnormalized probabilities	normalize	probabilities
cat	3.2		24.5		0.13
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

→ L_i = -log(0.13) = **2.04**

Maximum Likelihood Estimation
Choose probabilities to maximize the likelihood of the observed data

5.1.3 소프트맥스 활성함수와 로그우도 목적함수

- 소프트맥스와 교차 엔트로피 목적함수
  - 로그우도 손실함수 ~ 교차엔트로피 최소화

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i|X = x_i)$$

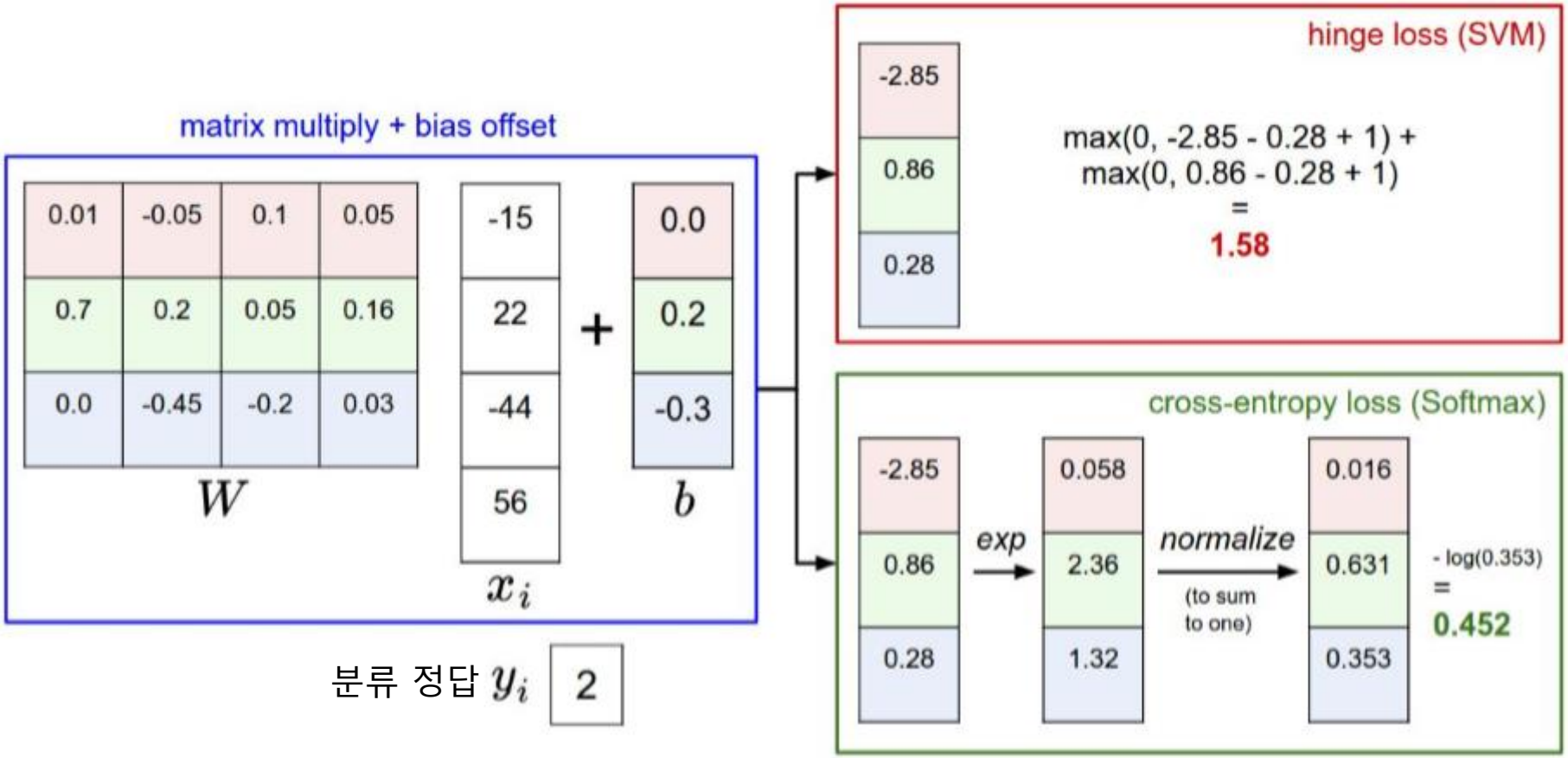
	Unnormalized log-probabilities / logits	exp	unnormalized probabilities	normalize	probabilities	compare	Correct probs
cat	3.2		24.5		0.13		1.00
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Cross Entropy

$$H(P, Q) = H(p) + D_{KL}(P\|Q)$$

5.1.3 소프트맥스 활성함수와 로그우도 목적함수

- 소프트맥스와 힌지로스 출력 비교



5.2 성능 향상을 위한 요령

- 5.2.1 데이터 전처리
- 5.2.2 가중치 초기화
- 5.2.3 모멘텀
- 5.2.4 적응적 학습률
- 5.2.5 활성함수
- 5.2.6 배치 정규화

5.2 성능 향상을 위한 요령

- 여러 연구결과들을 공유
  - 『Neural Networks: Tricks of the Trade』는 연구결과를 한데 묶은 대표적인 책(1998년에 1판, 2012년에 2판)
- 5.2절의 내용은 경험적 규칙
  - 주어진 데이터에 잘 들어맞을지는 실험을 통해 신중히 확인해야 함
  - Bengio의 권고 [Bengio2012]

"… the wisdom distilled here should be taken as a guideline, to be tried and challenged, not as a practice set in stone. … 이 논문이 제시한 정제된 기법들은 자신의 문제에 적용한 다음 변형하여 새로운 기법을 만드는 길잡이 역할 정도로 받아들여야지 만고불변의 법칙으로 여겨서는 안 된다."

5.2.1 데이터 전처리

규모scale 문제

- 예) 건강에 관련된 데이터 $(\text{키(m)}, \text{몸무게(kg)}, \text{혈압})^{\mathrm{T}}$
  - 1.885m와 1.525m는 33cm나 차이가 나지만 특징값 차이는 불과 0.33
  - 65.5kg과 45.0kg은 20.5라는 차이
  - 첫 번째와 두 번째 특징은 양수이며, 대략 100배의 규모 차이

- $-\delta_j z_i$가 그레이디언트이기 때문에 첫 번째 특징에 연결된 가중치는 두 번째 특징에 연결된 가중치에 비해 100여 배 느리게 학습됨

  → 느린 학습의 요인

5.2.1 데이터 전처리

- 모든 특징이 양수인 경우의 문제
  - $-\delta_j z_i$가 그레이디언트이므로

    [그림 5-6]의 경우 ↑ 표시된 가중치는 모두 증가, ↓ 표시된 가중치는 모두 감소
  - 이처럼 가중치가 뭉치로 증가 또는 감소하면

    최저점을 찾아가는 경로가 갈팡질팡하여 느린 수렴

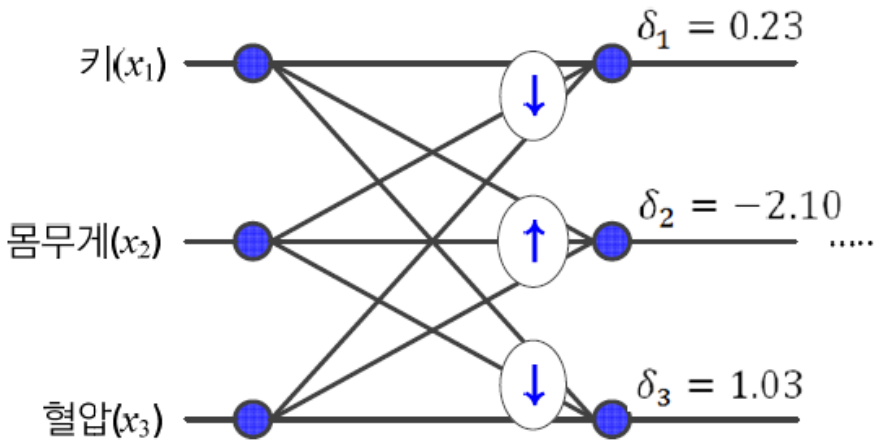


그림 5-6 특징이 모두 양수일 때 가중치가 뭉치로 갱신되는 효과

5.2.1 데이터 전처리

- 식 (5.9)의 정규화normalization는 규모 문제와 양수 문제를 해결해줌
  - 특징별로 독립적으로 적용
  - 통계학의 표준화 변환을 적용한 경우,

$$x_i^{new} = \frac{x_i^{old} - \mu_i}{\sigma_i} \qquad (5.9)$$

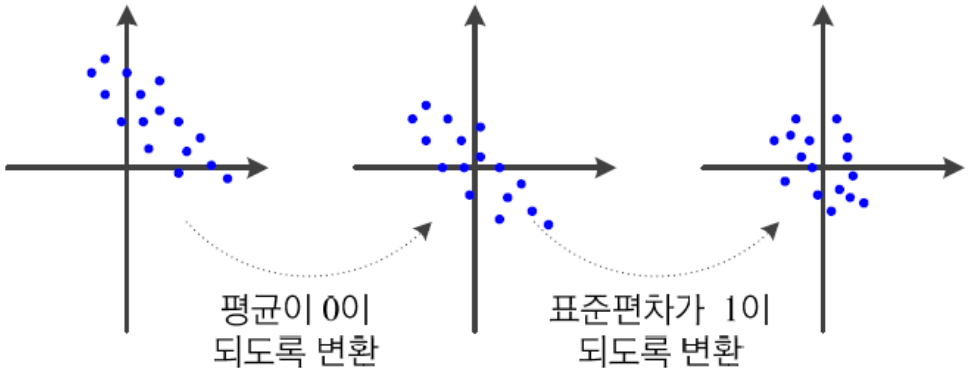


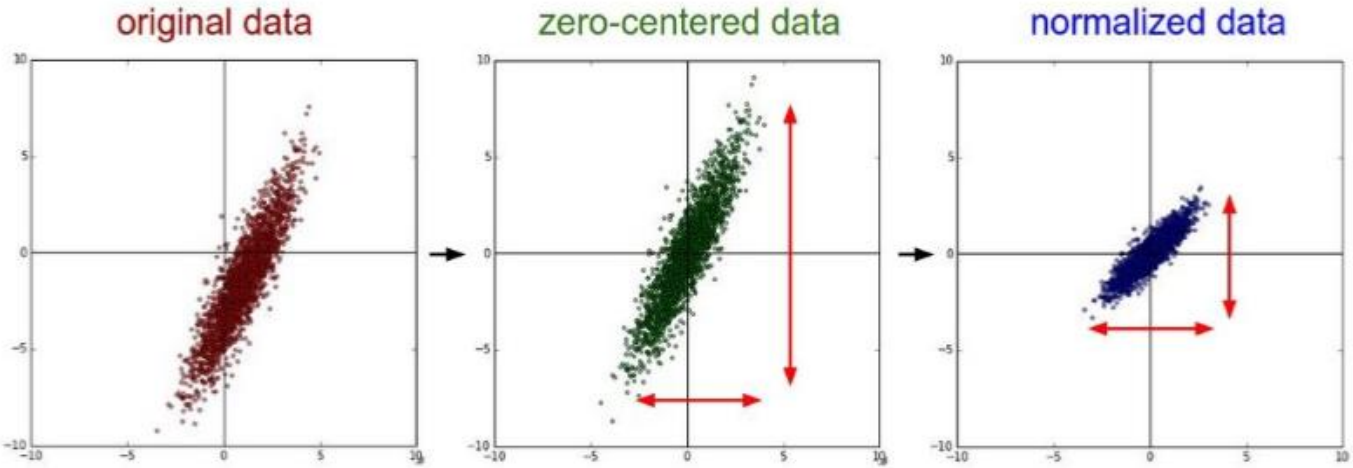
그림 5-7 표준점수로 변환

  - 일부의 경우, 최대 최소 변환을 적용한 경우

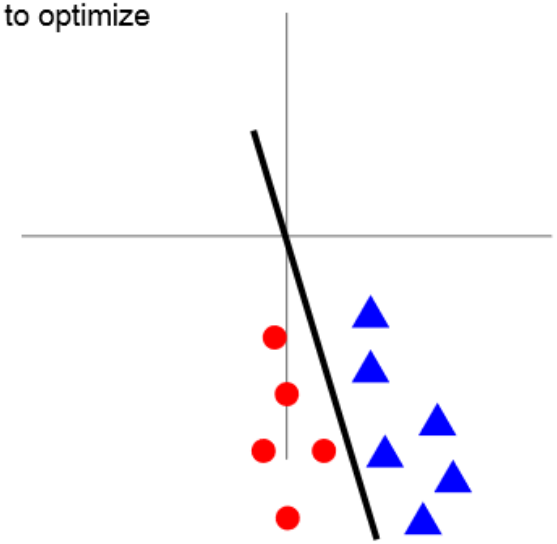
$$x_i^{new} = \frac{x_i^{old} - \min(x_i)}{\max(x_i) - \min(x_i)}$$

5.2.1 데이터 전처리

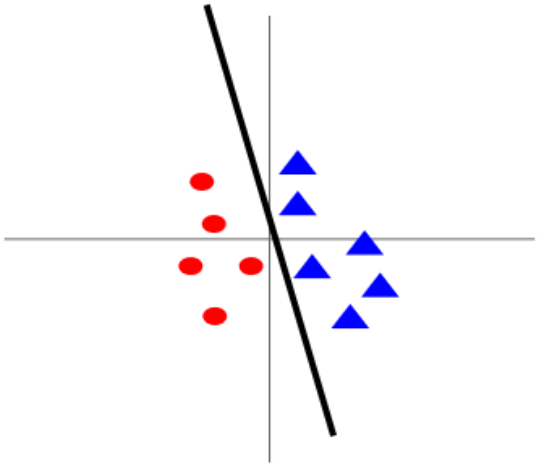
- 데이터 전처리 과정 예



Before normalization: classification loss very sensitive to changes in weight matrix; hard to optimize

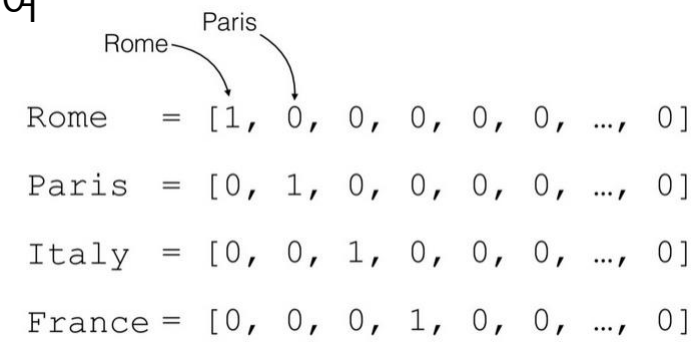


After normalization: less sensitive to small changes in weights; easier to optimize



5.2.1 데이터 전처리

- 명목 변수nominal value을 원핫one-hot 코드로 변환
  - 명목 변수: 객체간 서로 구분하기 위한 변수
    - 예) 성별의 남(1)과 여(2), 체질의 태양인(1), 태음인(2), 소양인(3), 소음인(4)
  - 명목 변수는 거리 개념이 없음
  - 원핫 코드는 값의 개수만큼 비트bit를 부여



    - 성별은 2비트, 체질은 4비트 부여
    - 예) 키 1.755m, 몸무게 65.5kg, 혈압 122, 남자, 소양인 샘플

$$(1.755, 65.5, 122, 1, 3) \rightarrow (1.755, 65.5, 122, \underbrace{[1, 0]}_{\text{성별}}, \underbrace{[0, 0, 1, 0]}_{\text{체질}})$$

5.2.2 가중치 초기화

대칭적 가중치 문제

- [그림 5-8]의 대칭적 가중치에서는 z_1^{l-1}과 z_2^{l-1}가 같은 값이 됨

  $-\delta_j z_i$가 그레이디언트이기 때문에 u_{11}^l과 u_{12}^l이 같은 값으로 갱신됨

  → 두 노드가 같은 일을 하는 중복성 발생

- 난수로 초기화함으로써 대칭 파괴symmetry break

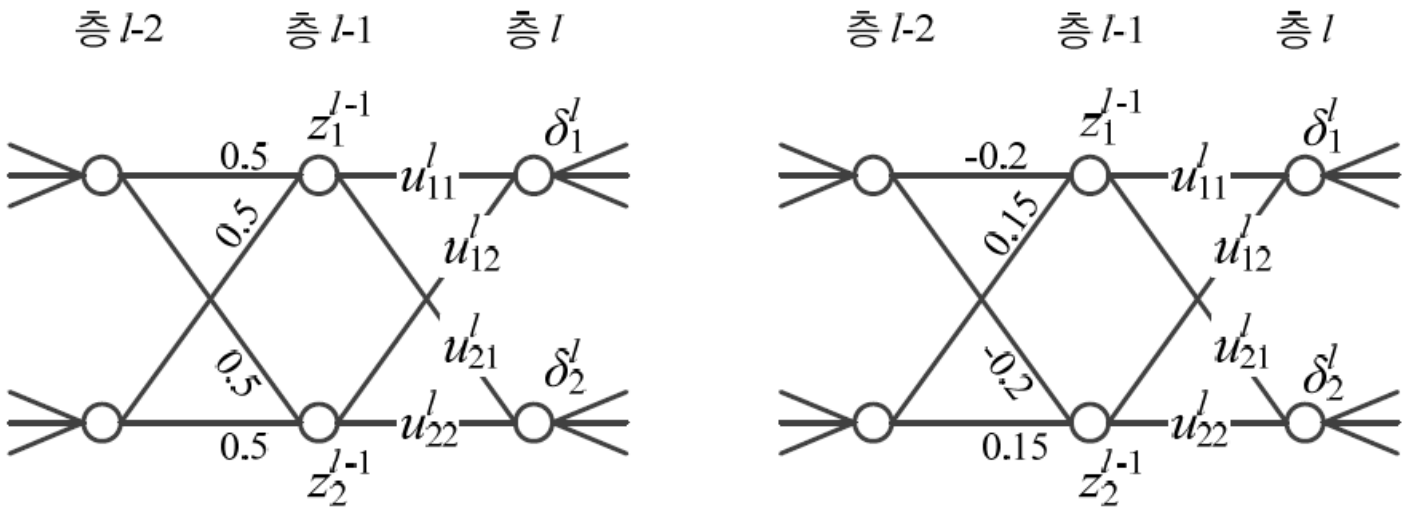


그림 5-8 대칭적 가중치로 초기화된 경우의 중복성 문제

5.2.2 가중치 초기화

- 난수로 가중치 초기화
  - 가우시안 분포 또는 균일 분포에서 난수 추출. 두 분포는 성능 차이 거의 없음
  - 난수 범위는 무척 중요함

    → 식 (5.10) 또는 식 (5.11)로 r을 결정한 후 $[-r, r]$ 사이에서 난수 발생

$$r = \frac{1}{\sqrt{n_{in}}} \qquad (5.10)$$

$$r = \frac{\sqrt{6}}{\sqrt{n_{in} + n_{out}}} \qquad (5.11)$$

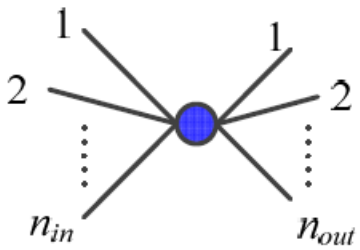
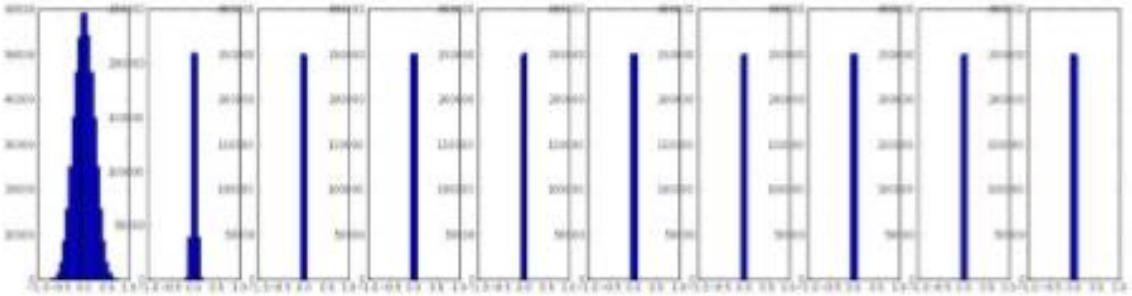


그림 5-9 노드로 들어 오는 에지 개수 n_{in}과 노드에서 나가는 에지 개수 n_{out}

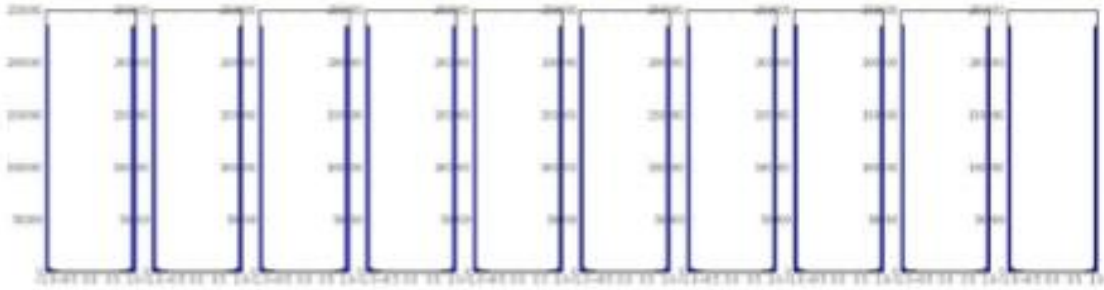
  - 바이어스는 보통 0으로 초기화
- 사례
  - AlexNet [Krizhevsky2012]: 평균 0, 표준편차 0.01인 가우시안에서 난수 생성
  - ResNet [He2016a]: 평균 0, 표준편차 $\sqrt{\frac{2}{n_{in}}}$인 가우시안에서 난수 생성, 바이어스 0 설정

5.2.2 가중치 초기화

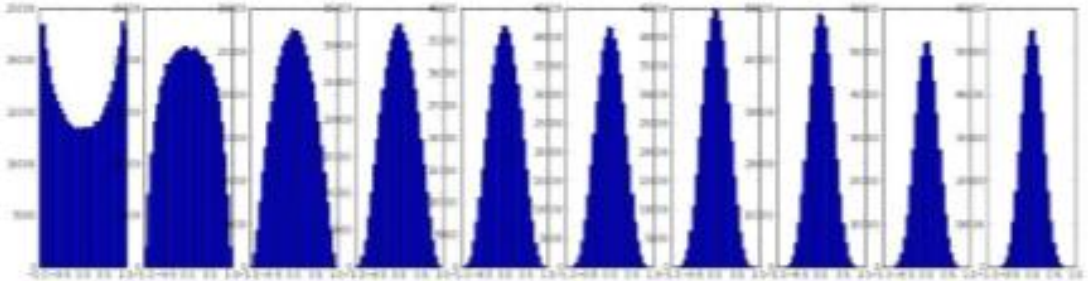
- 가중치 초기화에 따른 변화 예



초기화가 너무 작으면,
모든 활성값 영이 됨, 그레이디엔트도 역시 영
학습 안됨



초기화가 너무 크면,
활성값 포화, 그레이디엔트는 영
학습 안됨



초기화가 적당하면,
모든 층에서 활성값의 분포가 좋음
적절한 학습 수행

5.2.2 가중치 초기화

다른 방법들

- [Saxe2014]: 가중치 벡터가 수직이 되도록 설정
  - 분포로 난수로 가중치 설정하고 가중치 행렬을 특이SVD 분해로 분해한 후 가중치 재조정
- [Sussillo2014]: 임의 행로random walk 활용하여 설정
- [Sutskever2013]: 가중치 초기화와 모멘텀을 동시에 최적화
- [Mishkin2016]: 가중치 분포가 아니라 노드의 출력값 분포가 일정하도록 강제화
  - 1단계: 수직 규칙을 적용하여 가중치를 초기화
  - 2단계: 미니배치 1개를 가지고 전방 계산을 수행하면서 가중치 조정

    이 때, 각 층에 대해 노드 출력값의 분산이 1이 될 때까지 가중치 조정함

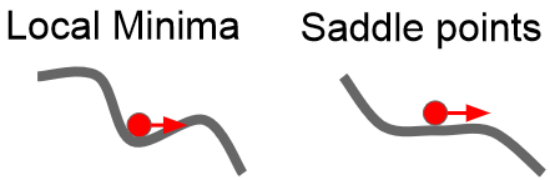
  → 훈련 집합에 맞는 가중치를 초기화

5.2.3 모멘텀

그레이디언트의 잡음 현상

- 기계 학습은 훈련집합을 이용하여 그레이디언트를 추정하므로 잡음 가능성 높음
- 모멘텀momentum은 그레이디언트에 부드럽게 함smoothing을 가하여 잡음 효과 줄임
  - 관성: 과거에 이동했던 방식을 기억하면서 기존 방향으로 일정 이상 추가 이동함

→ 수렴 속도 향상 (지역 최저에 빠지는 문제 해소)



모멘텀을 적용한 가중치 갱신 수식

$$\left.\begin{aligned} \mathbf{v} &= \alpha\mathbf{v} + \rho\frac{\partial J}{\partial \boldsymbol{\Theta}} \\ \boldsymbol{\Theta} &= \boldsymbol{\Theta} - \mathbf{v} \end{aligned}\right\} \quad (5.12)$$

- 속도 벡터 $\mathbf{v}$는 이전 그레이디언트를 누적한 것에 해당함 (처음에는 $\mathbf{v} = 0$로 출발)
- α의 효과 (관성의 정도)
  - $\alpha = 0$이면 모멘텀이 적용 안 된 이전 공식과 같음
  - α가 1에 가까울수록 이전 그레이디언트 정보에 큰 가중치를 주는 셈
    → $\boldsymbol{\Theta}$가 그리는 궤적이 매끄러움
  - 보통 0.5, 0.9, 또는 0.99 사용
    (또는 0.5로 시작하여 세대epoch가 지남에 따라 점점 키워 0.99에 도달하는 방법)

5.2.3 모멘텀

모멘텀의 효과

- 지나침overshooting 현상 누그러뜨림

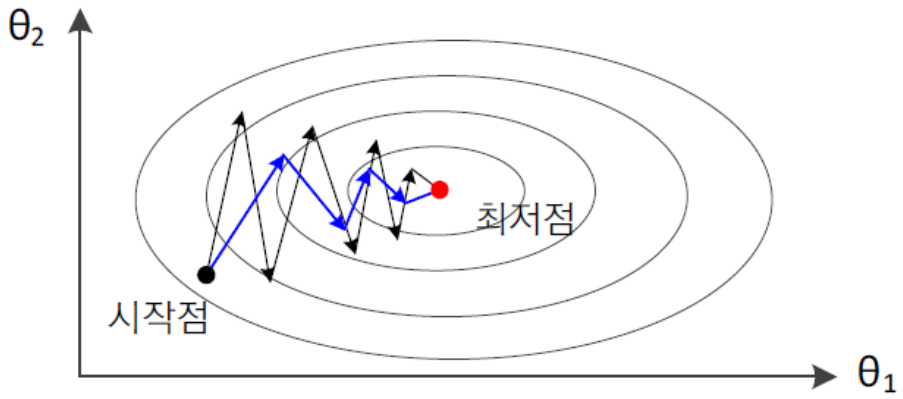


그림 5-10 모멘텀 효과

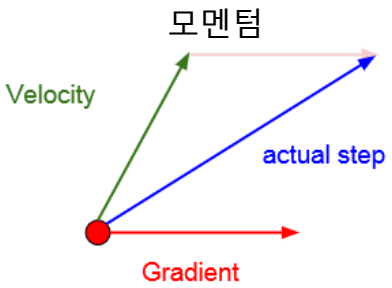
네스테로프 가속 그레이디언트nesterov accelerated gradient 모멘텀

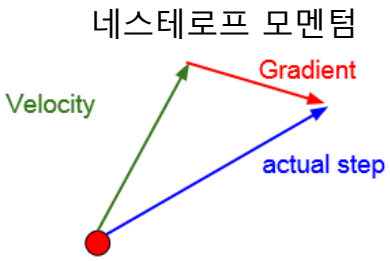
- 현재 $\mathbf{v}$ 값으로 다음 이동할 곳 $\tilde{\boldsymbol{\Theta}}$를 예견한 후, 예견한 곳의 그레이디언트 $\frac{\partial J}{\partial \boldsymbol{\Theta}}\Big|_{\tilde{\boldsymbol{\Theta}}}$를 사용

$$\tilde{\boldsymbol{\Theta}} = \boldsymbol{\Theta} + \alpha \mathbf{v}$$

$$\mathbf{v} = \alpha \mathbf{v} - \rho \frac{\partial J}{\partial \boldsymbol{\Theta}}\Big|_{\tilde{\boldsymbol{\Theta}}} \qquad (5.13)$$

$$\boldsymbol{\Theta} = \boldsymbol{\Theta} + \mathbf{v}$$





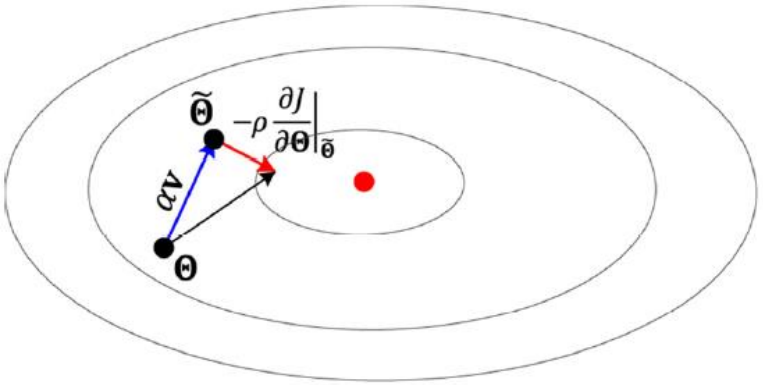


그림 5-11 네스테로프 모멘텀

5.2.3 모멘텀

- 일반적인 경사 하강 알고리즘

알고리즘 5-1 경사 하강법

입력: 훈련집합 $\mathbb{X}$, $\mathbb{Y}$, 학습률 ρ
출력: 최적의 매개변수 $\hat{\boldsymbol{\Theta}}$

1 난수를 생성하여 초기해 $\boldsymbol{\Theta}$를 설정한다.
2 repeat
3 $\quad \mathbf{g} = \frac{\partial J}{\partial \boldsymbol{\Theta}}\Big|_{\boldsymbol{\Theta}}$ // $\boldsymbol{\Theta}$에서 그레이디언트 $\frac{\partial J}{\partial \boldsymbol{\Theta}}\Big|_{\boldsymbol{\Theta}}$를 구해 $\mathbf{g}$라 한다.
4 $\quad \boldsymbol{\Theta} = \boldsymbol{\Theta} - \rho\mathbf{g}$ // 식 (2.58)
5 until (멈춤 조건)
6 $\hat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}$

- 네스테로프 모멘텀을 적용한 경사 하강 알고리즘

알고리즘 5-2 네스테로프 모멘텀을 적용한 경사 하강법

입력: 훈련집합 $\mathbb{X}$, $\mathbb{Y}$, 학습률 ρ, 속도 조절 계수 α
출력: 최적의 매개변수 $\hat{\boldsymbol{\Theta}}$

1 난수를 생성하여 초기해 $\boldsymbol{\Theta}$를 설정한다.
2 $\mathbf{v} = \mathbf{0}$ // 속도 항을 $\mathbf{0}$ 벡터로 초기화
3 repeat
4 $\quad \tilde{\boldsymbol{\Theta}} = \boldsymbol{\Theta} + \alpha\mathbf{v}$ // 예견
5 $\quad \mathbf{g} = \frac{\partial J}{\partial \boldsymbol{\Theta}}\Big|_{\tilde{\boldsymbol{\Theta}}}$ // 예견한 곳의 그레이디언트
6 $\quad \mathbf{v} = \alpha\mathbf{v} - \rho\mathbf{g}$
7 $\quad \boldsymbol{\Theta} = \boldsymbol{\Theta} + \mathbf{v}$
8 until (멈춤 조건)
9 $\hat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}$

5.2.4 적응적 학습률

- 학습률 ρ의 중요성
  - 너무 크면 지나침overshooting에 따른 진자 현상, 너무 작으면 수렴이 느림

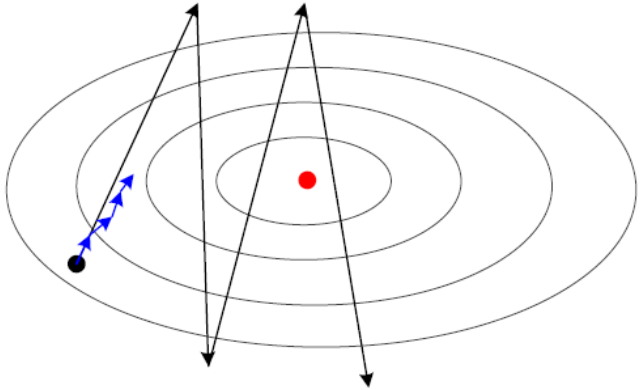
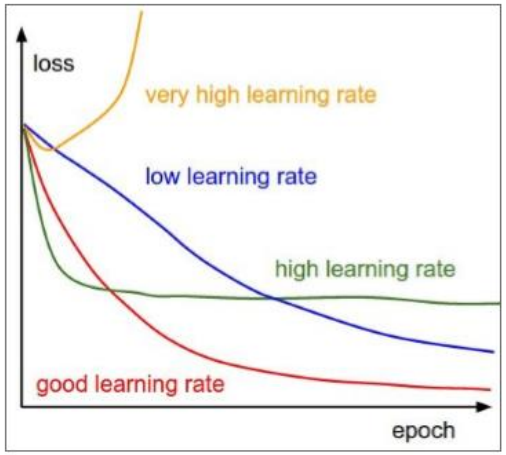


그림 5-12 학습률의 크기에 따른 최적화 알고리즘의 이동 궤적



- 적응적 학습률adaptive learning rates 혹은 per-parameter learning rates
  - 그레이디언트에 학습률 ρ를 곱하면, $\rho\frac{\partial J}{\partial \Theta} = \left(\rho\frac{\partial J}{\partial \theta_1}, \rho\frac{\partial J}{\partial \theta_2}, \cdots, \rho\frac{\partial J}{\partial \theta_k}\right)^{\mathrm{T}}$

    즉, 모든 매개변수가 같은 크기의 학습률을 사용하는 셈
  - 적응적 학습률은 매개변수마다 자신의 상황에 따라 학습률을 조절해 사용
    - 예) 학습률 담금질annealing
      - 이전 그레이디언트와 현재 그레이디언트의 부호가 같은 매개변수는 값을 키우고 다른 매개변수는 값을 줄이는 전략을 사용

5.2.4 적응적 학습률

- AdaGrad adaptive gradient
  - 라인 5~7을 자세히 쓰면

5. $(r_1, r_2, \cdots, r_k)^{\mathrm{T}} = (r_1 + g_1^2, r_2 + g_2^2, \cdots, r_k + g_k^2)^{\mathrm{T}}$
6. $(\Delta\theta_1, \Delta\theta_2, \cdots, \Delta\theta_k)^{\mathrm{T}} = \left(-\frac{\rho g_1}{\epsilon+\sqrt{r_1}}, -\frac{\rho g_2}{\epsilon+\sqrt{r_2}}, \cdots, -\frac{\rho g_k}{\epsilon+\sqrt{r_k}}\right)^{\mathrm{T}}$
7. $(\theta_1, \theta_2, \cdots, \theta_k)^{\mathrm{T}} = (\theta_1 + \Delta\theta_1, \theta_2 + \Delta\theta_2, \cdots, \theta_k + \Delta\theta_k)^{\mathrm{T}}$

  - $\mathbf{r}$은 이전 그레이디언트를 누적한 벡터
    - r_i가 크면 갱신값 $|\Delta\theta_i|$는 작아서 조금만 이동
    - r_i가 작으면 $|\Delta\theta_i|$는 커서 많이 이동
    - 예) [그림 5-10]에서 θ_1은 θ_2보다 보폭이 큼
  - 라인 6의 $\frac{\rho}{\epsilon+\sqrt{r_i}}$는 상황에 따라 보폭을 정해주는 적응적 학습률로 볼 수 있음
    - ϵ 분모가 0이 됨을 방지, 보통 10^{-5}~10^{-7} 범위 값으로 설정

알고리즘 5-3 AdaGrad

입력: 훈련집합 $\mathbb{X}$, $\mathbb{Y}$, 학습률 ρ
출력: 최적의 매개변수 $\hat{\boldsymbol{\Theta}}$

1 난수를 생성하여 초기해 $\boldsymbol{\Theta}$를 설정한다.
2 $\mathbf{r} = \mathbf{0}$ // 그레이디언트 누적 벡터 초기화
3 repeat
4 그레이디언트 $\mathbf{g} = \frac{\partial J}{\partial \boldsymbol{\Theta}}\Big|_{\boldsymbol{\Theta}}$를 구한다.
5 $\mathbf{r} = \mathbf{r} + \mathbf{g} \odot \mathbf{g}$ // $\odot$는 요소별 곱
6 $\Delta\boldsymbol{\Theta} = -\frac{\rho}{\epsilon+\sqrt{\mathbf{r}}} \odot \mathbf{g}$
7 $\boldsymbol{\Theta} = \boldsymbol{\Theta} + \Delta\boldsymbol{\Theta}$
8 until (멈춤 조건)
9 $\hat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}$

5.2.4 적응적 학습률

- RMSProp
  - AdaGrad의 단점
    - [알고리즘 5-3]의 라인 5는 단순히 제곱을 더함
    - 따라서, 오래된 그레이디언트와 최근 그레이디언트는 같은 비중의 역할
      → $\mathbf{r}$이 점점 커져 수렴 방해할 가능성
  - RMSProp은 가중 이동 평균weight moving average 기법 적용

$$\mathbf{r} = \alpha\mathbf{r} + (1-\alpha)\mathbf{g}\odot\mathbf{g} \qquad (5.14)$$

    - α가 작을수록 최근 것에 비중을 둠
    - 보통 α로 0.9, 0.99, 0.999를 사용

알고리즘 5-4 RMSProp

입력: 훈련집합 $\mathbb{X}$, $\mathbb{Y}$, 학습률 ρ, 가중 이동 평균 계수 α
출력: 최적의 매개변수 $\hat{\boldsymbol{\Theta}}$

1 난수를 생성하여 초기해 $\boldsymbol{\Theta}$를 설정한다.
2 $\mathbf{r} = \mathbf{0}$ // 그레이디언트 누적 벡터 초기화
3 repeat
4 그레이디언트 $\mathbf{g} = \frac{\partial J}{\partial \boldsymbol{\Theta}}\Big|_{\boldsymbol{\Theta}}$를 구한다.
5 $\mathbf{r} = \alpha\mathbf{r} + (1-\alpha)\mathbf{g}\odot\mathbf{g}$ // ⊙는 요소별 곱
6 $\Delta\boldsymbol{\Theta} = -\frac{\rho}{\epsilon+\sqrt{\mathbf{r}}}\odot\mathbf{g}$
7 $\boldsymbol{\Theta} = \boldsymbol{\Theta} + \Delta\boldsymbol{\Theta}$
8 until (멈춤 조건)
9 $\hat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}$

5.2.4 적응적 학습률

- Adam adaptive moment
  - RMSProp에 식 (5-12)의 모멘텀을 추가로 적용한 알고리즘

알고리즘 5-5 Adam

입력: 훈련집합 $\mathbb{X}$, $\mathbb{Y}$, 학습률 ρ, 모멘텀 계수 α_1, 가중 이동 평균 계수 α_2
출력: 최적의 매개변수 $\hat{\boldsymbol{\Theta}}$

1 난수를 생성하여 초기해 $\boldsymbol{\Theta}$를 설정한다.
2 $\mathbf{v} = \mathbf{0}, \mathbf{r} = \mathbf{0}$
3 t=1
4 repeat
5 그레이디언트 $\mathbf{g} = \frac{\partial J}{\partial \boldsymbol{\Theta}}\Big|_{\boldsymbol{\Theta}}$를 구한다.
6 $\mathbf{v} = \alpha_1 \mathbf{v} - (1-\alpha_1)\mathbf{g}$ // 속도 벡터
7 $\mathbf{v} = \frac{1}{1-(\alpha_1)^t}\mathbf{v}$
8 $\mathbf{r} = \alpha_2 \mathbf{r} + (1-\alpha_2)\mathbf{g} \odot \mathbf{g}$ // 그레이디언트 누적 벡터
9 $\mathbf{r} = \frac{1}{1-(\alpha_2)^t}\mathbf{r}$
10 $\Delta\boldsymbol{\Theta} = -\frac{\rho}{\epsilon+\sqrt{\mathbf{r}}}\mathbf{v}$
11 $\boldsymbol{\Theta} = \boldsymbol{\Theta} + \Delta\boldsymbol{\Theta}$
12 t++
13 until (멈춤 조건)
14 $\hat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}$

일반적으로 α_1=0.9, α_2=0.999, ρ=1e-3 혹은 5e-4 설정

5.2.4 적응적 학습률

- 동작 예시
  - [그림 5-13(a)]는 중앙으로 급강하하는 절벽 지형
  - [그림 5-13(b)]는 중앙 부근에 안장점이saddle point 있는 지형

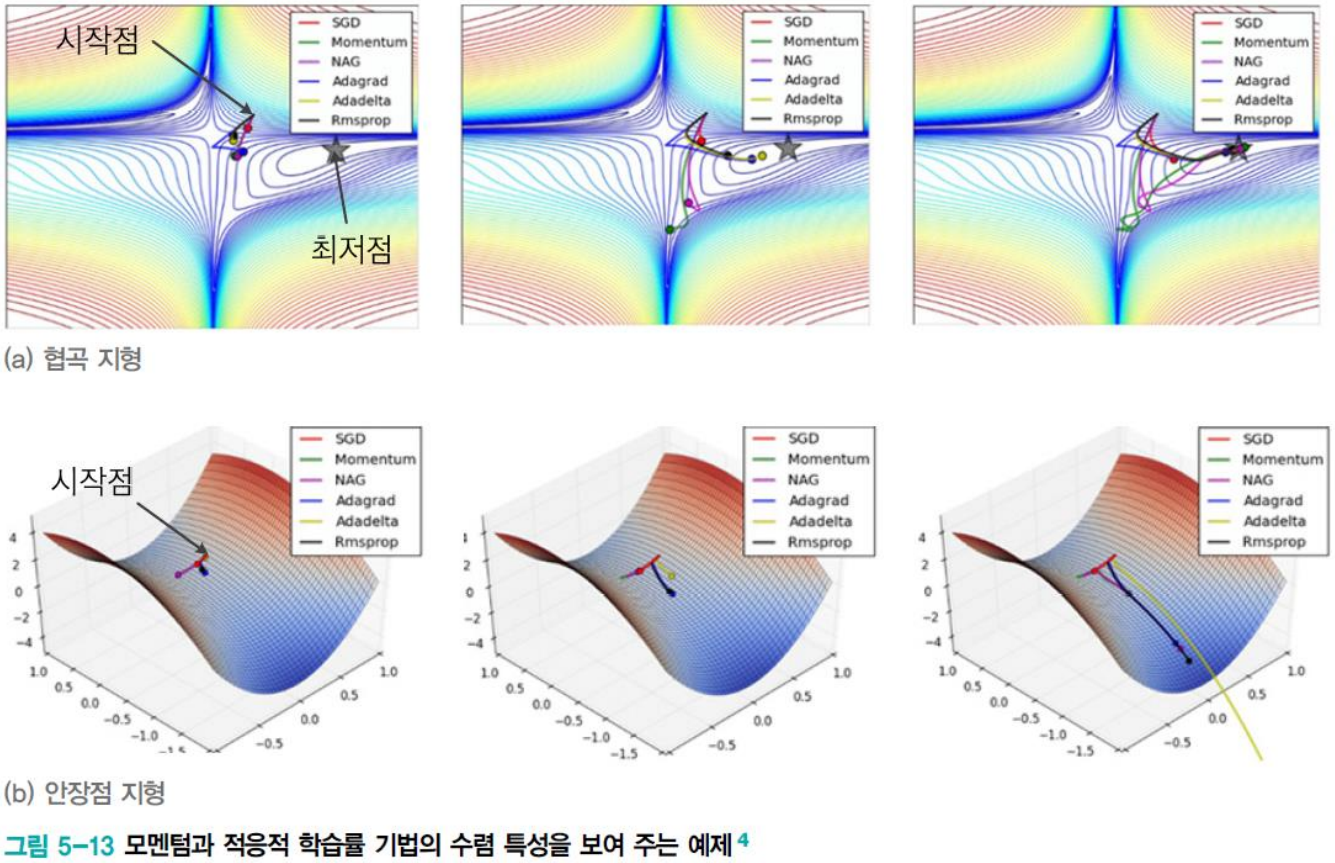


그림 5-13 모멘텀과 적응적 학습률 기법의 수렴 특성을 보여 주는 예제[4]

  - SGD, SGD+Momentum, Adagrad, RMSProp, Adam 모두 학습률을 하이퍼매개변수로 가짐

5.2.5 활성 함수

- 선형 연산 결과인 활성값 z를 계산하고 비선형 활성함수 τ를 적용하는 과정

$$z = \mathbf{w}^{\mathrm{T}}\tilde{\mathbf{x}} + b$$
$$y = \tau(z) \qquad (5.15)$$

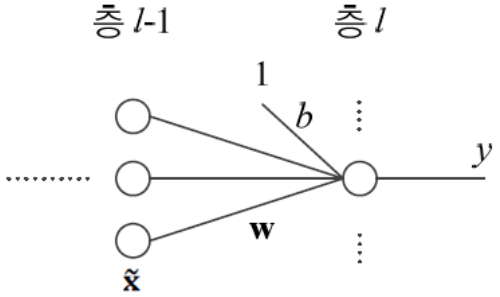


그림 5-14 신경망 노드의 연산

- 활성함수 변천사

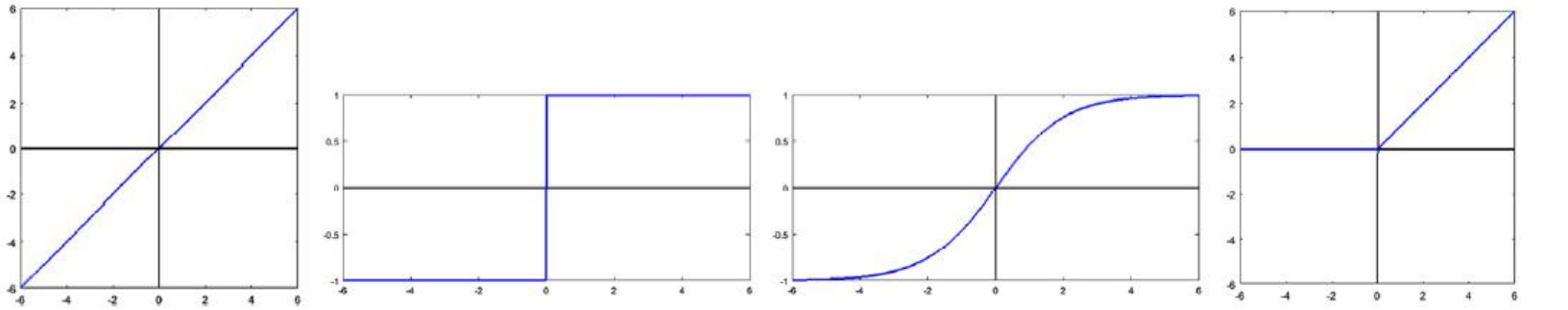


그림 5-15 활성함수 τ

  - tanh는 활성값이 커지면 포화 상태가 되고 그레이디언트는 0에 가까워짐

→ 매개변수 갱신 (학습)이 매우 느린 요인

5.2.5 활성 함수

- ReLU (Rectified Linear Unit) 활성함수
  - 그레이디언트 포화gradient saturation 문제 해소

$$z = \mathbf{w}^{\mathrm{T}}\tilde{\mathbf{x}} + b$$
$$y = \mathrm{ReLU}(z) = \max(0, z) \quad (5.16)$$

- ReLU의 변형
  - Leaky ReLU (보통 $\alpha = 0.01$을 사용)
  - PReLU (α를 학습으로 알아냄)

$$\mathrm{leakyReLU}(z) = \begin{cases} z, & z \geq 0 \\ az, & z < 0 \end{cases} \quad (5.17)$$

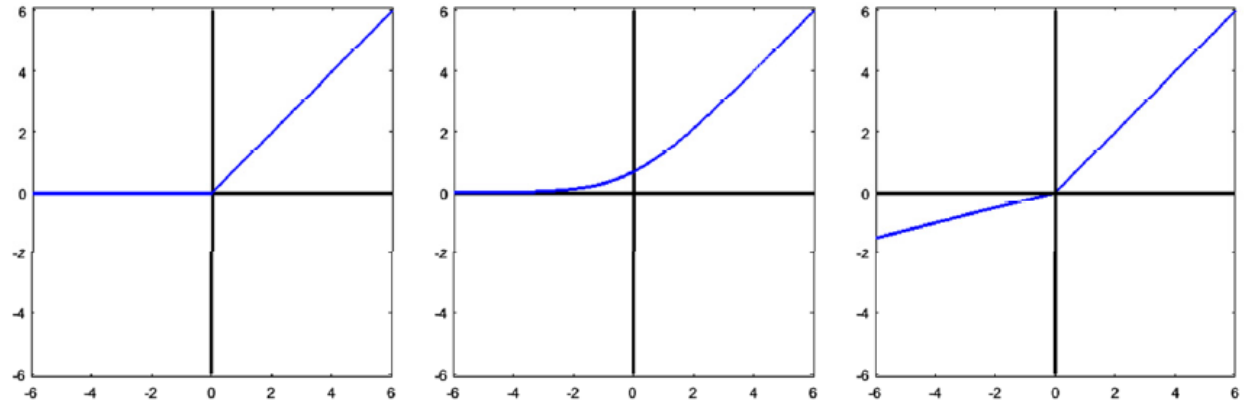


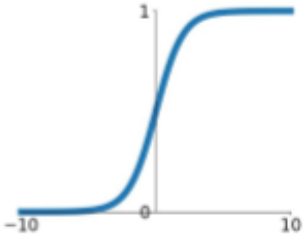
그림 5-16 ReLU의 변형

5.2.5 활성 함수

- 다양한 활성 함수들

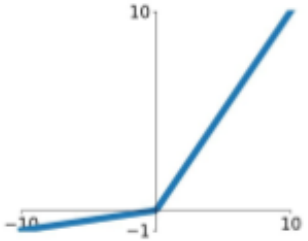
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



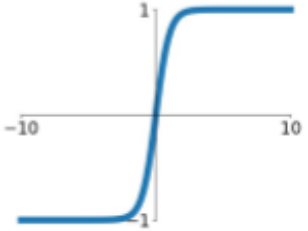
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$



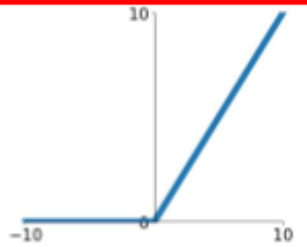
Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ReLU

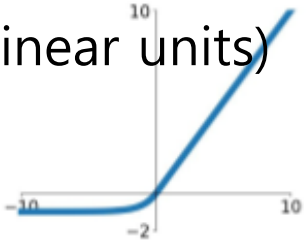
$$\max(0, x)$$

Good default choice



ELU (exponential linear units)

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



최근의 활성 함수들은 다음의 문제들을 해결하고자 함
- 포화된 영역이 그레이디언트를 제거함
- 출력이 영 중심 아님
- 다소 높은 Exp()의 연산량

5.2.6 배치 정규화

- 공변량 시프트covariate shift 현상
  - 학습이 진행되면서 층1의 매개변수가 바뀜에 따라 $\tilde{\mathbb{X}}^{(1)}$이 따라 바뀜

    → 층2 입장에서 보면 자신에게 입력되는 데이터의 분포가 수시로 바뀌는 셈
  - 층2, 층3, ...으로 깊어짐에 따라 더욱 심각
  - 학습을 방해하는 요인으로 작용

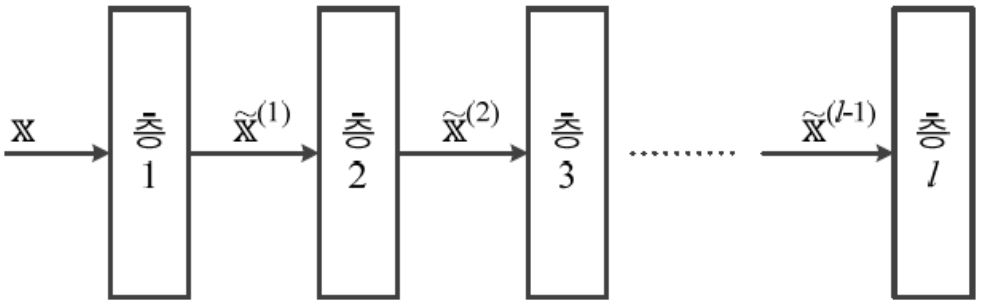


그림 5-17 공변량 시프트 현상

5.2.6 배치 정규화

배치 정규화batch normalization

- 공변량 시프트 현상을 누그러뜨리기 위해 식 (5.9)의 정규화를 모든 층에 적용하는 기법

$$x_i^{new} = \frac{x_i^{old} - \mu_i}{\sigma_i} \qquad (5.9)$$

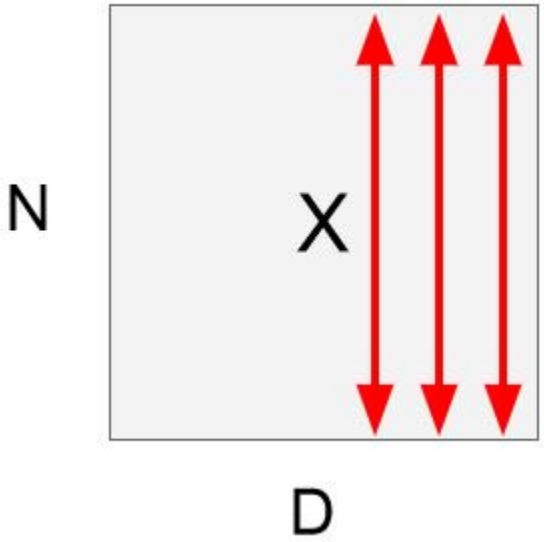
- 정규화를 적용하는 곳이 중요
  - 식 (5.15)의 연산 과정 중 식 (5.9)를 어디에 적용하나? (적용 위치)

$$z = \mathbf{w}^{\mathrm{T}}\tilde{\mathbf{x}} + b$$
$$y = \tau(z) \qquad (5.15)$$

  - 입력 $\tilde{\mathbf{x}}$ 또는 중간 결과 z 중 어느 것에 적용? → z에 적용하는 것이 유리
  - 일반적으로 완전 연결층, 컨볼루션 층 직후 혹은 비선형 함수 적용 직전 적용
- 훈련집합 전체 또는 미니배치 중 어느 것에 적용? (적용 단위)
  - 미니배치에 적용하는 것이 유리

5.2.6 배치 정규화

배치 정규화 과정



1. 통계적 평균과 분산을
각 차원에 대해서 독립적으로 계산

2. 정규화

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

배치 정규화 장점

- 신경망의 그레이디언트 흐름 개선
- 높은 학습률 허용
- 초기화에 대한 의존성 감소
- 의도하지 않았지만 규제와 유사한 행동을 하며, 드롭아웃의 필요성을 감소시킴

5.2.6 배치 정규화

- 정규화 변환을 수행하는 코드
  - 미니배치 $\mathbb{X}_B = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$에 식 (5.15)를 적용하여 $\tilde{\mathbb{X}}_B = \{z_1, z_2, \cdots, z_m\}$를 얻은 후, $\tilde{\mathbb{X}}_B$를 가지고 코드 1을 수행
  - 즉, 미니배치 단위로 노드마다 독립적으로 코드 1을 수행
  - γ와 β는 노드마다 고유한 매개변수로서 학습으로 알아냄

코드 1:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} z_i \qquad \text{\# 미니배치 평균}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (z_i - \mu_B)^2 \qquad \text{\# 미니배치 분산}$$

$$\tilde{z}_i = \frac{z_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}, \quad i = 1,2,\cdots,m \qquad \text{\# 정규화}$$

$$z_i' = \gamma \tilde{z}_i + \beta, \quad i = 1,2,\cdots,m \qquad \text{\# 비례scale와 변화shift}$$

(β가 바이어스 역할을 하므로 식 (5.15)의 바이어스 항 b는 제거해도 됨)

5.2.6 배치 정규화

- 최적화를 마친 후 추가적인 후처리 작업 필요
  - 각 노드는 전체 훈련집합을 가지고 독립적으로 코드2를 수행

코드 2:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} z_i$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (z_i - \mu)^2$$

노드에 μ, σ^2, γ, β를 저장한다. // 예측 단계에서 식 (5.18)로 변환을 수행하기 위함

- 예측 단계
  - 각 노드는 독립적으로 식 (5.18)을 적용하여 변환 (코드 1의 마지막 두 라인을 수행하는 셈)

$$z' = \frac{\gamma}{\sqrt{\sigma^2 + \varepsilon}} z + \left(\beta - \frac{\gamma\mu}{\sqrt{\sigma^2 + \varepsilon}}\right) \qquad (5.18)$$

5.2.6 배치 정규화

- CNN에서는
  - 노드 단위가 아니라 특징 맵 단위로 코드 1과 코드 2를 적용
    - 예를 들면,

      특징 맵의 크기가 p*q라면 미니배치에 있는 샘플마다 pq개의 값이 발생

      코드 1은 총 pqm개의 값을 가지고 μ_B와 σ_B^2를 계산

      γ와 β는 특징 맵마다 하나씩 존재

- 배치 정규화의 긍정적 효과를 측정한 실험사례 [Ioffe2015]
  - 가중치 초기화에 덜 민감함
  - 학습률을 크게 하여 수렴 속도 향상 가능
  - 시그모이드를 활성함수로 사용하는 깊은 신경망도 학습이 이루어짐

- 배치 정규화는 규제 효과를 제공
  - 드롭아웃이라는 규제 기법을 적용하지 않아도 높은 성능

5.3 규제의 필요성과 원리

- 5.3.1 과잉적합에 빠지는 이유와 과잉적합을 피하는 전략
- 5.3.2 규제의 정의

- 규제가 중요하기 때문에 1장에서 미리 소개한 내용
  - 1.5절의 과소적합과 과잉적합, 바이어스와 분산 ([그림 1-13], [그림 1-14])
  - 1.6절의 데이터 확대와 가중치 감쇠

5.3.1 과잉적합에 빠지는 이유와 과잉적합을 피하는 전략

- 학습 모델의 용량에 따른 일반화 능력

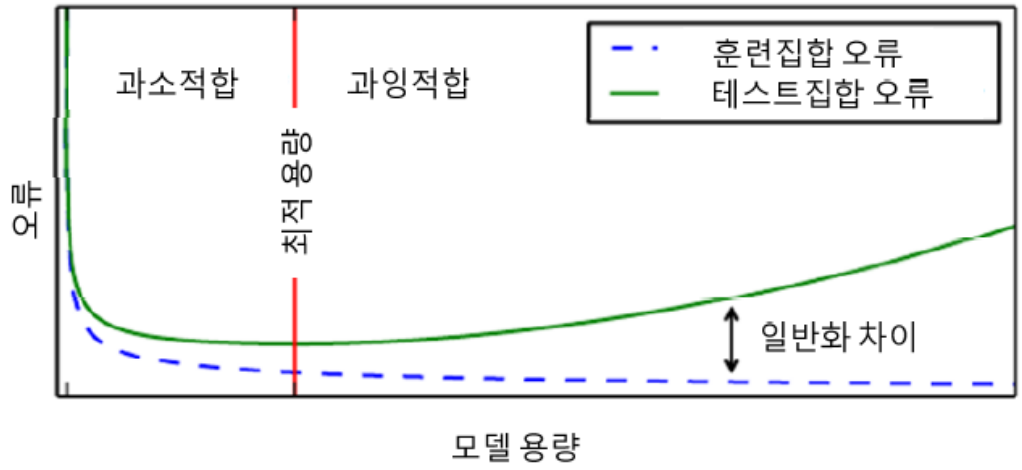


그림 5-18 학습 모델의 용량과 일반화 능력의 관계

- 대부분 가지고 있는 데이터에 비해 훨씬 큰 용량의 모델을 사용
  - 예) VGGNet은 분류층에 1억 2천 1백만 개의 매개변수
  - 훈련집합을 단순히 '암기'하는 과잉적합에 주의를 기울여야 함
- 현대 기계 학습의 전략
  - 충분히 큰 용량의 모델을 설계한 다음, 학습 과정에서 여러 규제 기법을 적용

5.3.2 규제의 정의

- 규제는 오래 전부터 수학과 통계학에서 연구해온 주제
  - 모델 용량에 비해 데이터가 부족한 경우의 불량조건문제를ill-posed problem 푸는 데 최초 제안
  - 적절한 가정을 투입하여 문제를 품 → '입력과 출력 사이의 변환은 매끄럽다'는 사전 지식
    - 데이터의 원천에 내재한 정보를 사전 지식이라고 하며, 이를 활용

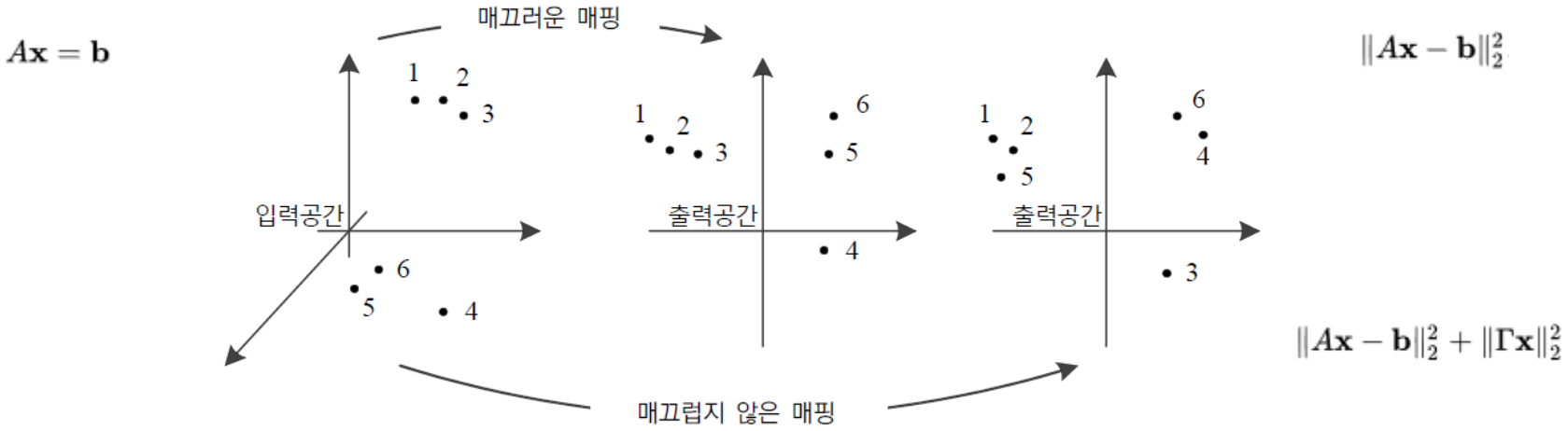


그림 5-20 사전 지식으로서 매끄러움의 특성

- 티호노프의 규제Tikhonov's regularization 기법는 매끄러움 가정에 기반을 둔 식 (5.19)를 사용
  - 통계에서는 릿지 회귀ridge regression, 기계학습에서는 가중치 감쇄weight decay 등이 대표적임

$$\underbrace{J_{regularized}(\Theta)}_{\text{규제를 적용한 목적함수}} = \underbrace{J(\Theta)}_{\text{목적함수}} + \lambda \underbrace{R(\Theta)}_{\text{규제 항}} \qquad (5.19)$$

5.3.2 규제의 정의

- 현대 기계 학습도 매끄러움 가정을 널리 사용함
  - 5.4.1절의 가중치 감쇠 기법
    - 모델의 구조적 용량을 충분히 크게 하고, '수치적 용량'을 제한하는 규제 기법
  - 6장의 비지도 학습 등

- 『Deep Learning』 책의 정의

"...any modification we make to a learning algorithm that is intended to reduce its generalization error ... 일반화 오류를 줄이려는 의도를 가지고 학습 알고리즘을 수정하는 방법 모두"

5.4 규제 기법

- 5.4.1 가중치 벌칙
- 5.4.2 조기 멈춤
- 5.4.3 데이터 확대
- 5.4.4 드롭아웃
- 5.4.5 앙상블 기법

- 명시적 규제와 암시적 규제
  - 명시적 규제: 가중치 감쇠나 드롭아웃처럼 목적함수나 신경망 구조를 직접 수정하는 방식
  - 암시적 규제: 조기 멈춤, 데이터 증대, 잡음 추가, 앙상블처럼 간접적으로 영향을 미치는 방식

5.4.1 가중치 벌칙

- 식 (5.19)를 관련 변수가 드러나도록 다시 쓰면,

$$\underbrace{J_{regularized}(\boldsymbol{\Theta};\mathbb{X},\mathbb{Y})}_{\text{규제를 적용한 목적함수}} = \underbrace{J(\boldsymbol{\Theta};\mathbb{X},\mathbb{Y})}_{\text{목적함수}} + \lambda \underbrace{R(\boldsymbol{\Theta})}_{\text{규제 항}} \qquad (5.20)$$

  - 규제항은 훈련집합과 무관하며, 데이터 생성 과정에 내재한 사전 지식에 해당
  - 규제항은 매개변수를 작은 값으로 유지하므로 모델의 용량을 제한하는 역할

    (수치적 용량을 제한함)

- 규제항 $R(\Theta)$로 무엇을 사용할 것인가?
  - 큰 가중치에 벌칙을 가해 작은 가중치를 유지하려고 주로 $L2$ 놈이나 $L1$ 놈을 사용

5.4.1 가중치 벌칙

- $L2$ 놈norm
  - 규제 항 R로 $L2$ 놈을 사용하는 규제 기법을 '가중치 감쇠'weight decay라 부름 → 식 (5.21)

$$\underbrace{J_{regularized}(\boldsymbol{\Theta};\mathbb{X},\mathbb{Y})}_{\text{규제를 적용한 목적함수}} = \underbrace{J(\boldsymbol{\Theta};\mathbb{X},\mathbb{Y})}_{\text{목적함수}} + \lambda \underbrace{\|\boldsymbol{\Theta}\|_2^2}_{\text{규제 항}} \qquad (5.21)$$

  - 식 (5.21)의 그레이디언트 계산

$$\nabla J_{regularized}(\boldsymbol{\Theta};\mathbb{X},\mathbb{Y}) = \nabla J(\boldsymbol{\Theta};\mathbb{X},\mathbb{Y}) + 2\lambda\boldsymbol{\Theta} \qquad (5.22)$$

5.4.1 가중치 벌칙

- 식 (5.22)를 이용하여 매개변수를 갱신하는 수식

$$\boldsymbol{\Theta} = \boldsymbol{\Theta} - \rho\nabla J_{regularized}(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y})$$

$$= \boldsymbol{\Theta} - \rho(\nabla J(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y}) + 2\lambda\boldsymbol{\Theta}) \longrightarrow \boldsymbol{\Theta} = (1 - 2\rho\lambda)\boldsymbol{\Theta} - \rho\nabla J \quad (5.23)$$

$$= (1 - 2\rho\lambda)\boldsymbol{\Theta} - \rho\nabla J(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y})$$

- $\lambda = 0$으로 두면 규제를 적용하지 않은 원래 식 $\boldsymbol{\Theta} = \boldsymbol{\Theta} - \rho\nabla J$가 됨
- 가중치 감쇠는 단지 $\boldsymbol{\Theta}$에 $(1 - 2\rho\lambda)$를 곱해주는 셈
  - 예를 들어, ρ=0.01, $\lambda = 2.0$이라면 $(1 - 2\rho\lambda)$=0.96
- 최종해를 원점 가까이 당기는 효과 (즉 가중치를 작게 유지함)

  = 가중치를 퍼트리는spread out 효과

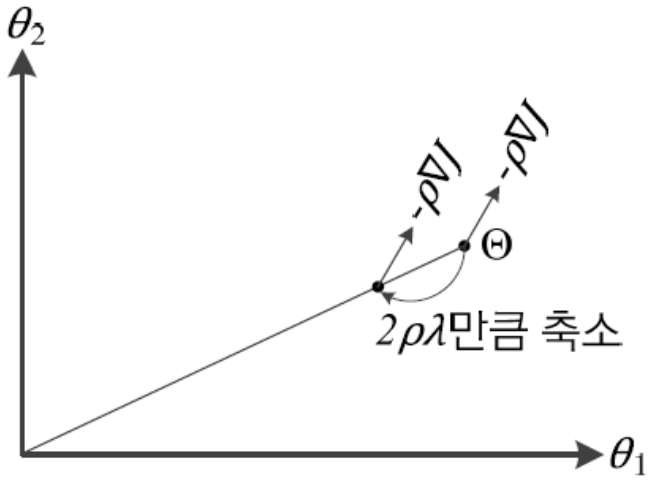


그림 5-21 L2 놈을 사용한 가중치 감쇠 기법의 효과

5.4.1 가중치 벌칙

- 선형 회귀linear regression에 적용
  - 선형 회귀는 훈련집합 $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, $\mathbb{Y} = \{y_1, y_2, \cdots, y_n\}$이 주어지면,

    식 (5.24)를 풀어 $\mathbf{w} = (w_1, w_2, \cdots, w_d)^{\mathrm{T}}$를 구하는 문제. 이때 $\mathbf{x_i} = (x_{i1}, x_{i2}, \cdots, x_{id})^{\mathrm{T}}$

$$w_1 x_{i1} + w_2 x_{i2} \cdots + w_d x_{id} = \mathbf{x}_i^{\mathrm{T}} \mathbf{w} = y_i, \qquad i = 1,2, \cdots, n \tag{5.24}$$

  - 식 (5.24)를 행렬식으로 바꿔 쓰면,

$$\mathbf{X}\mathbf{w} = \mathbf{y} \tag{5.25}$$

  - 가중치 감쇠를 적용한 목적함수

$$J_{regularized}(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 = (\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathrm{T}}(\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \|\mathbf{w}\|_2^2 \tag{5.27}$$

5.4.1 가중치 벌칙

- 식 (5.27)을 미분하여 0으로 놓으면,

$$\frac{\partial J_{regularized}}{\partial \mathbf{w}} = \mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} - \mathbf{X}^{\mathrm{T}}\mathbf{y} + 2\lambda\mathbf{w} = \mathbf{0} \implies (\mathbf{X}^{\mathrm{T}}\mathbf{X} + 2\lambda\mathbf{I})\mathbf{w} = \mathbf{X}^{\mathrm{T}}\mathbf{y} \quad (5.28)$$

- 식 (5.28)을 정리하면,

$$\hat{\mathbf{w}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + 2\lambda\mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \quad (5.29)$$

  - 공분산 행렬 $\mathbf{X}^{\mathrm{T}}\mathbf{X}$의 대각 요소가 2λ만큼씩 증가

→ 역행렬을 곱하므로 가중치를 축소하여 원점으로 당기는 효과 ([그림 5-21])

- 예측 단계에서는

$$y = \mathbf{x}^{\mathrm{T}}\hat{\mathbf{w}} \quad (5.30)$$

5.4.1 가중치 벌칙

예제 5-1 리지 회귀

훈련집합 $\mathbb{X} = \{\mathbf{x}_1 = \begin{pmatrix}1\\1\end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix}2\\3\end{pmatrix}, \mathbf{x}_3 = \begin{pmatrix}3\\3\end{pmatrix}\}$, $\mathbb{Y} = \{y_1 = 3.0, y_2 = 7.0, y_3 = 8.8\}$이 주어졌다고 가정하자. 특징 벡터가 2차원이므로 d=2이고 샘플이 3개이므로 n=3이다. 훈련집합으로 설계행렬 **X**와 레이블 행렬 **y**를 다음과 같이 쓸 수 있다.

$$\mathbf{X} = \begin{pmatrix}1 & 1\\2 & 3\\3 & 3\end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix}3.0\\7.0\\8.8\end{pmatrix}$$

이 값들을 식 (5.29)에 대입하여 다음과 같이 $\hat{\mathbf{w}}$을 구할 수 있다. 이때 $\lambda = 0.25$라 가정하자.

$$\hat{\mathbf{w}} = \left(\begin{pmatrix}1 & 2 & 3\\1 & 3 & 3\end{pmatrix}\begin{pmatrix}1 & 1\\2 & 3\\3 & 3\end{pmatrix} + \begin{pmatrix}0.5 & 0\\0 & 0.5\end{pmatrix}\right)^{-1}\begin{pmatrix}1 & 2 & 3\\1 & 3 & 3\end{pmatrix}\begin{pmatrix}3.0\\7.0\\8.8\end{pmatrix} = \begin{pmatrix}1.4916\\1.3607\end{pmatrix}$$

따라서 하이퍼 평면은 $y = 1.4916x_1 + 1.3607x_2$이다. 새로운 샘플로 $\mathbf{x} = (5 \quad 4)^{\mathrm{T}}$가 입력되면 식 (5.30)을 이용하여 12.9009를 예측한다.

5.4.1 가중치 벌칙

- MLP와 DMLP에 적용
  - 식 (3.21)에 식 (5.23)의 가중치 감쇠라는 규제 기법을 적용하면,

$$\left.\begin{aligned}\mathbf{U}^1 &= \mathbf{U}^1 - \rho\frac{\partial J}{\partial \mathbf{U}^1}\\ \mathbf{U}^2 &= \mathbf{U}^2 - \rho\frac{\partial J}{\partial \mathbf{U}^2}\end{aligned}\right\} \quad (3.21) \longrightarrow \left.\begin{aligned}\mathbf{U}^1 &= (1-2\rho\lambda)\mathbf{U}^1 - \rho\frac{\partial J}{\partial \mathbf{U}^1}\\ \mathbf{U}^2 &= (1-2\rho\lambda)\mathbf{U}^2 - \rho\frac{\partial J}{\partial \mathbf{U}^2}\end{aligned}\right\} \quad (5.31)$$

  - [알고리즘 3-4]에 적용하면,

13. for (k=1 to c) for (j=0 to p) $u_{kj}^2 = u_{kj}^2 - \rho\Delta u_{kj}^2$ // 가중치 감쇠 적용하지 않은 원래 알고리즘
14. for (j=1 to p) for (i=0 to d) $u_{ji}^1 = u_{ji}^1 - \rho\Delta u_{ji}^1$

⇓

13. for (k=1 to c) for (j=0 to p) $u_{kj}^2 = (1-2\rho\lambda)u_{kj}^2 - \rho\Delta u_{kj}^2$ // 가중치 감쇠 적용한 알고리즘
14. for (j=1 to p) for (i=0 to d) $u_{ji}^1 = (1-2\rho\lambda)u_{ji}^1 - \rho\Delta u_{ji}^1$

5.4.1 가중치 벌칙

- [알고리즘 3-6] 미니배치 버전에 적용하면,

14. $\mathbf{U}^2 = \mathbf{U}^2 - \rho\frac{\Delta\mathbf{U}^2}{t}$ // 가중치 감쇠 적용하지 않은 원래 알고리즘
15. $\mathbf{U}^1 = \mathbf{U}^1 - \rho\frac{\Delta\mathbf{U}^1}{t}$

⇓

14. $\mathbf{U}^2 = (1-2\rho\lambda)\mathbf{U}^2 - \rho\frac{\Delta\mathbf{U}^2}{t}$ // 가중치 감쇠 적용한 알고리즘
15. $\mathbf{U}^1 = (1-2\rho\lambda)\mathbf{U}^1 - \rho\frac{\Delta\mathbf{U}^1}{t}$

- DMLP를 위한 [알고리즘 4-1]에 적용하면,

16. for (l=L to 1) // 가중치 감쇠 적용하지 않은 원래 알고리즘
17. for (j=1 to n_l) for (i=0 to n_{l-1}) $u_{ji}^l = u_{ji}^l - \rho\left(\frac{1}{t}\right)\Delta u_{ji}^l$

⇓

16. for (l=L to 1) // 가중치 감쇠 적용한 알고리즘
17. for (j=1 to n_l) for (i=0 to n_{l-1}) $u_{ji}^l = (1-2\rho\lambda)u_{ji}^l - \rho\left(\frac{1}{t}\right)\Delta u_{ji}^l$

5.4.1 가중치 벌칙

$L1$ 놈

- 규제 항으로 $L1$ 놈을 적용하면, ($L1$ 놈은 $\|\boldsymbol{\Theta}\|_1 = |\theta_1| + |\theta_2| + \cdots$)

$$\underbrace{J_{regularized}(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y})}_{\text{규제를 적용한 목적함수}} = \underbrace{J(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y})}_{\text{목적함수}} + \lambda \underbrace{\|\boldsymbol{\Theta}\|_1}_{\text{규제 항}} \tag{5.32}$$

- 식 (5.32)를 미분하면,

$$\nabla J_{regularized}(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y}) = \nabla J(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y}) + \lambda \mathbf{sign}(\boldsymbol{\Theta}) \tag{5.33}$$

- 매개변수를 갱신하는 식에 대입하면,

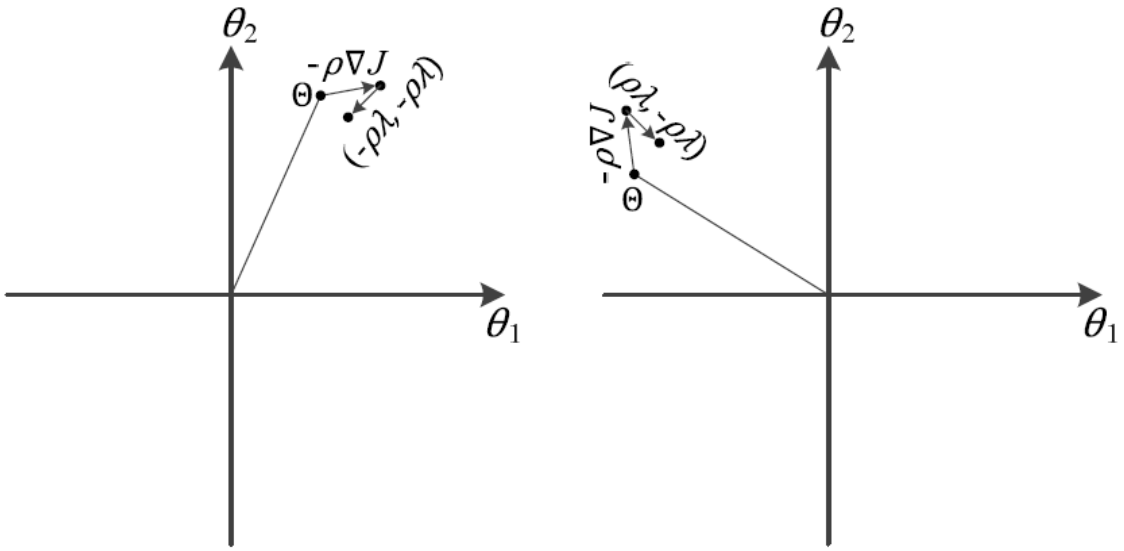
$$\begin{aligned}\boldsymbol{\Theta} &= \boldsymbol{\Theta} - \rho \nabla J_{regularized}(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y}) \\ &= \boldsymbol{\Theta} - \rho(\nabla J(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y}) + \lambda \mathbf{sign}(\boldsymbol{\Theta})) \\ &= \boldsymbol{\Theta} - \rho \nabla J(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y}) - \rho\lambda \mathbf{sign}(\boldsymbol{\Theta})\end{aligned}$$

5.4.1 가중치 벌칙

- 매개변수를 갱신하는 식

$$\boldsymbol{\Theta} = \boldsymbol{\Theta} - \rho\nabla J - \rho\lambda\mathbf{sign}(\boldsymbol{\Theta}) \qquad (5.34)$$

- 식 (5.34)의 가중치 감쇠 효과



(a) $\mathbf{sign}(\boldsymbol{\Theta}) = (1,1)^{\mathrm{T}}$인 경우 (b) $\mathbf{sign}(\boldsymbol{\Theta}) = (-1,1)^{\mathrm{T}}$인 경우

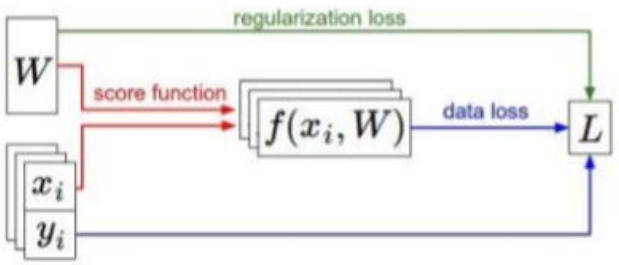
그림 5-22 *L1* 놈을 사용한 가중치 감쇠 기법의 효과

- $L1$ 놈의 희소성sparsity 효과 (0이 되는 매개변수가 많음)
  - 선형 회귀에 적용하면 특징 선택feature selection 효과

5.4.1 가중치 벌칙

규제

$$\underbrace{J_{regularized}(\mathbf{\Theta}; \mathbb{X}, \mathbb{Y})}_{\text{규제를 적용한 목적함수}} = \underbrace{J(\mathbf{\Theta}; \mathbb{X}, \mathbb{Y})}_{\text{목적함수}} + \lambda \underbrace{R(\mathbf{\Theta})}_{\text{규제 항}}$$



- 목적함수: 적용된 충분한 학습 모델로 훈련집합의 예측한 오차
- 규제: 학습 모델이 훈련집합의 예측을 너무 잘 수행하지 못하도록 방지

효과

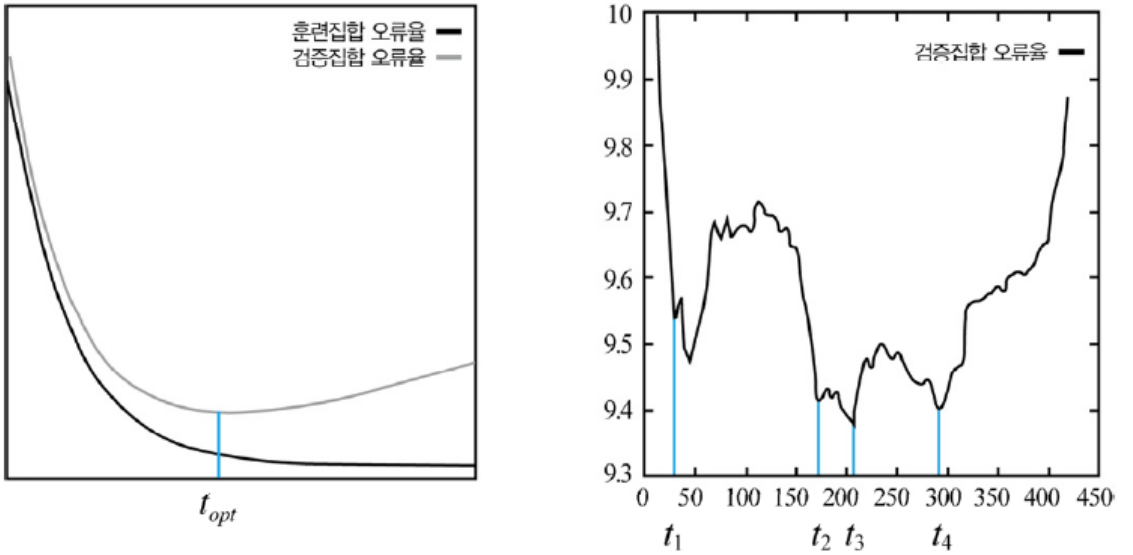
- 가중치에 대한 선호도 표현
- 학습 모델을 단순화시킴으로 테스트 집합에 대한 일반화 성능 향상 시킴
- 매끄럽게 하여 최적화 개선

대표적인 예

- L2 규제: $R(W) = \sum_k \sum_l W_{k,l}^2$
- L1 규제: $R(W) = \sum_k \sum_l |W_{k,l}|$
- 엘라스틱 넷elastic net: $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$ (L1+L2)

5.4.2 조기 멈춤

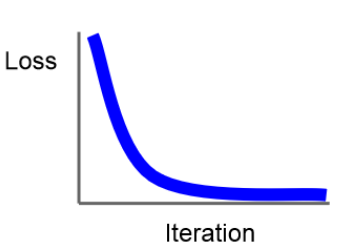
- 학습 시간에 따른 일반화 능력 [그림 5-23(a)]
  - 일정 시간 (t_{opt})이 지나면 과잉적합 현상이 나타남 → 일반화 능력 저하
  - 즉 훈련 데이터를 단순히 암기하기 시작

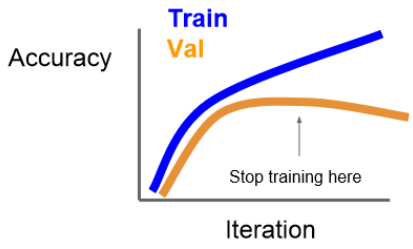


(a) 개념적인 도표 (b) 실제 데이터에 나타나는 지그재그 현상

그림 5-23 학습 시간에 따른 성능 추이

- 조기 멈춤early stopping이라는 규제 기법
  - 검증집합의 오류가 최저인 점 t_{opt}에서 학습을 멈춤





5.4.2 조기 멈춤

- [알고리즘 5-6]은 현실을 제대로 반영하지 않은 순진한 버전
  - [그림 5-23(a)] 상황에서 동작

알고리즘 5-6 조기 멈춤을 채택한 기계 학습 알고리즘(지그재그 현상을 고려하지 않은 순진한 버전)

입력 : 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$, 검증집합 $\mathbb{X}'$와 $\mathbb{Y}'$

출력 : 최적의 매개변수 $\hat{\Theta}$, 최적해가 발생한 세대 $\hat{t}$

```
1  난수를 생성하여 초기해 Θ0을 설정하고 오류율 e0 = 1.0으로 설정한다.   // 1.0은 오류율 최대치
2  t=0
3  while (true)
4     학습 알고리즘으로 Θt를 갱신하여 Θt+1을 얻는다.
5     Θt+1로 검증집합에 대한 오류율 et+1을 측정한다.
6     if(et+1 > et) break
7     t++
8  Θ̂ = Θt , t̂ = t
```

5.4.2 조기 멈춤

- 실제 현실은 [그림 5-23(b)]와 같은 상황
  - 순진한 버전을 적용하면 t_1에서 멈추므로 설익은 수렴
  - 이에 대처하는 여러 가지 방안 중에서 [알고리즘 5-7]은 참을성을 반영한 버전
    - 참을성: 연속적으로 성능 향상이 없으면 멈추는 정도

알고리즘 5-7 조기 멈춤을 채택한 기계 학습 알고리즘(참을성을 반영한 버전)

입력: 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$, 검증집합 $\mathbb{X}'$와 $\mathbb{Y}'$, 참을성 인자 p, 세대 반복 인자 q
출력: 최적의 매개변수 $\hat{\Theta}$, 최적해가 발생한 세대 $\hat{t}$

```
1  난수를 생성하여 초기해 Θ0을 설정한다.
2  Θ̂ = Θ0, t̂ = 0
3  t = 0, ê = 1.0, j = 0
4  while (j<p)
5    학습 알고리즘의 세대를 q번 반복하여 Θt+q를 얻는다.
6    Θt+q로 검증집합에 대한 오류율 et+q를 측정한다.
7    if (et+q < ê) // 새로운 최적을 발견한 상황
8      j = 0   // 참는 과정을 처음부터 새로 시작
9      Θ̂ = Θt+q, ê = et+q, t̂ = t + q
10   else
11     j=j+1
12   t=t+q
```

5.4.3 데이터 확대

- 과잉적합 방지하는 가장 확실한 방법은 큰 훈련집합 사용

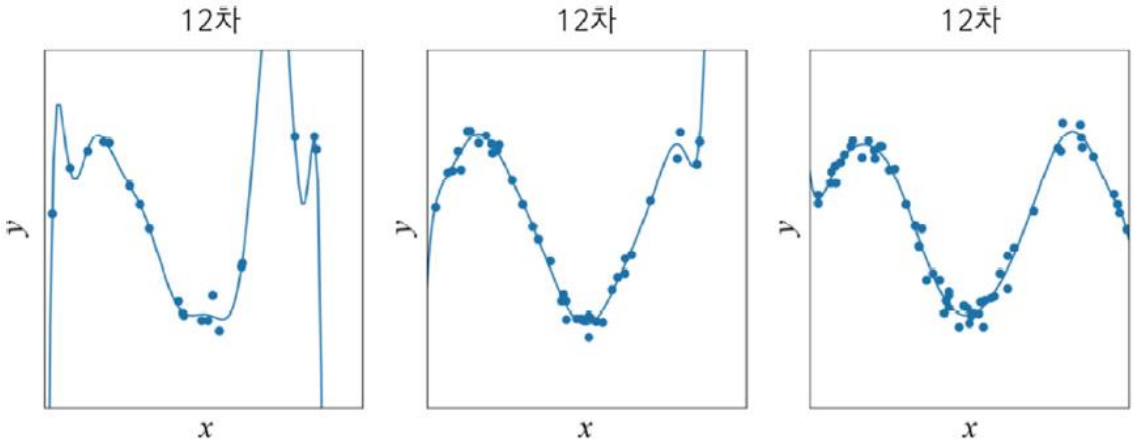


그림 1-17 데이터를 확대하여 일반화 능력을 향상함

  - 하지만 데이터 수집은 비용이 많이 드는 작업

- 데이터 확대라는 규제 기법
  - 데이터를 인위적으로 변형하여 확대함
  - 자연계에서 벌어지는 잠재적인 변형을 프로그램으로 흉내 내는 셈

5.4.3 데이터 확대

- 예) MNIST에 아핀affine 변환(이동translation, 회전rotation, 반전reflection)을 적용

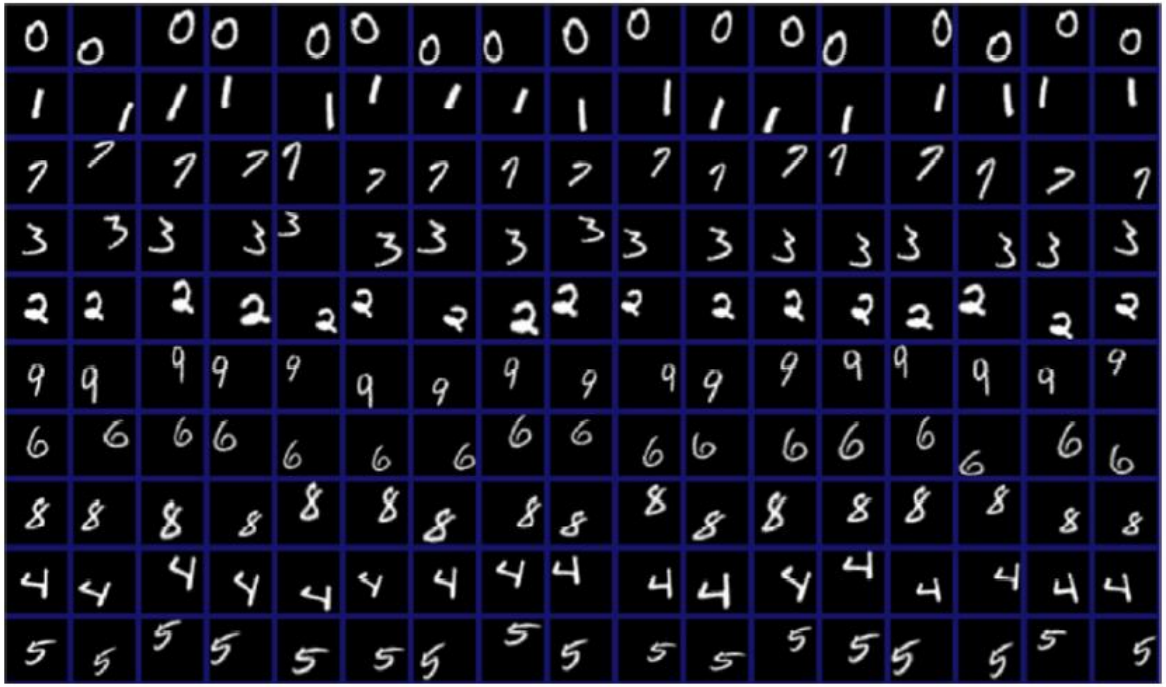


그림 5-24 필기 숫자 데이터의 다양한 변형[8]

  - 한계
    - 수작업 변형
    - 모든 부류가 같은 변형 사용

5.4.3 데이터 확대

- 예) 모핑morphing을 이용한 변형 [Hauberg2016]

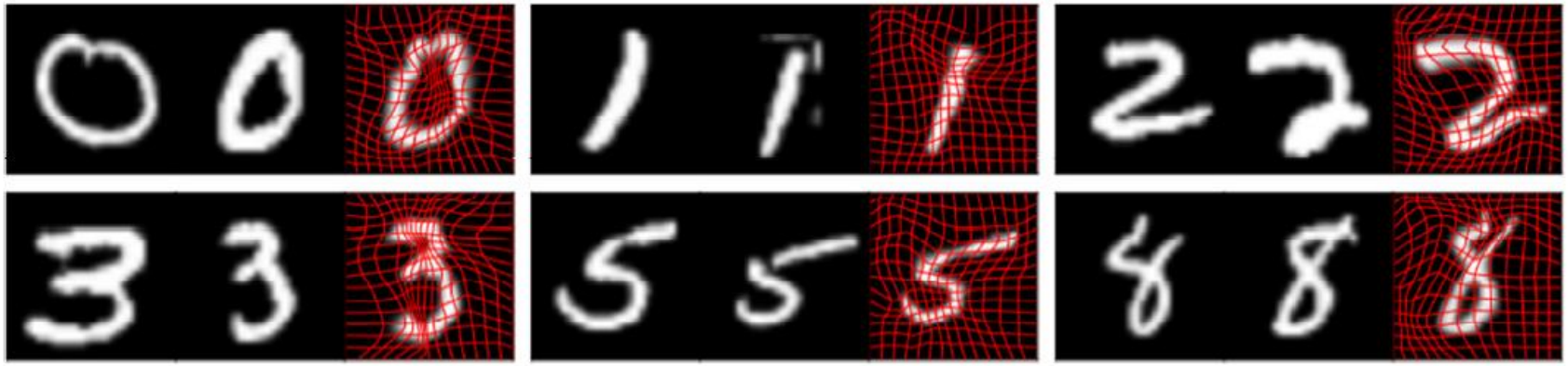
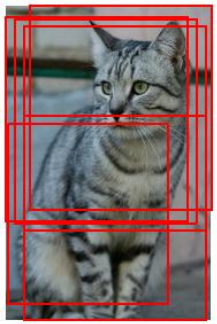


그림 5-25 비선형 변환 학습

- 비선형 변환으로서 아핀 변환에 비해 훨씬 다양한 형태의 확대
- 학습 기반: 데이터에 맞는 '비선형 변환 규칙을 학습'하는 셈

5.4.3 데이터 확대



- 예) 자연영상 확대 [Krizhevsky2012]
  - 256*256 영상에서 224*224 영상을 1024장 잘라내어 이동 효과
    좌우 반전까지 시도하여 2048배로 확대
  - PCA를 이용한 색상 변환color jitter으로 추가 확대
  - 예측 단계에서는 [그림 5-26]과 같이 5장 잘라내고 좌우 반전하여 10장을 만든 후, 앙상블 적용하여 정확도 향상

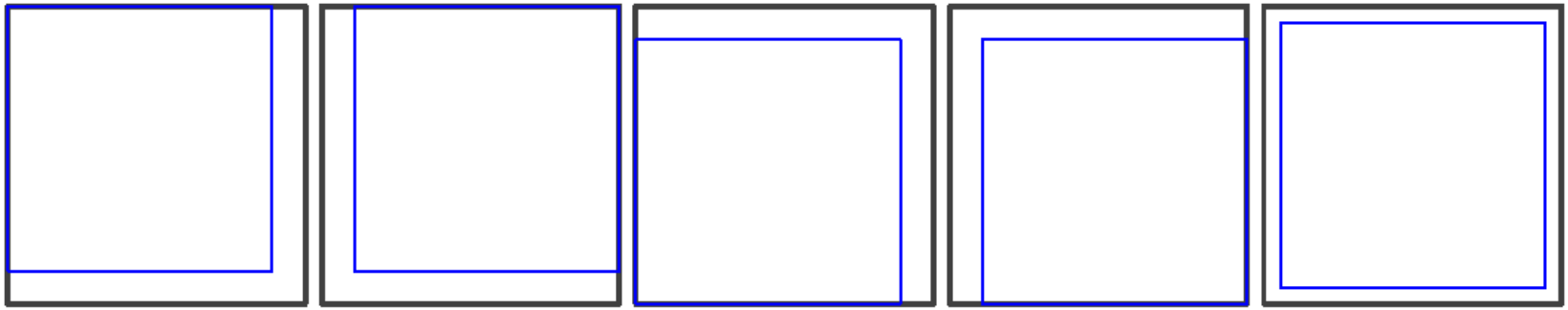
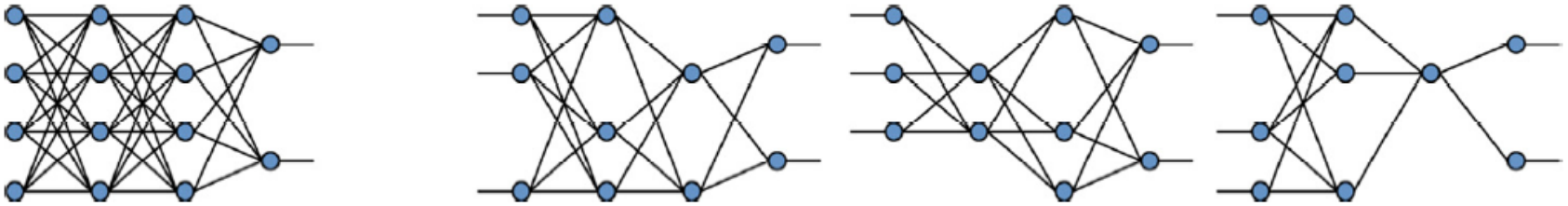


그림 5-26 예측 단계에서 영상 잘라내기

- 예) 잡음을 섞어 확대하는 기법
  - 입력 데이터에 잡음을 섞는 기법
  - 은닉 노드에 잡음을 섞는 기법 (고급 특징 수준에서 데이터를 확대하는 셈)

5.4.4 드롭아웃

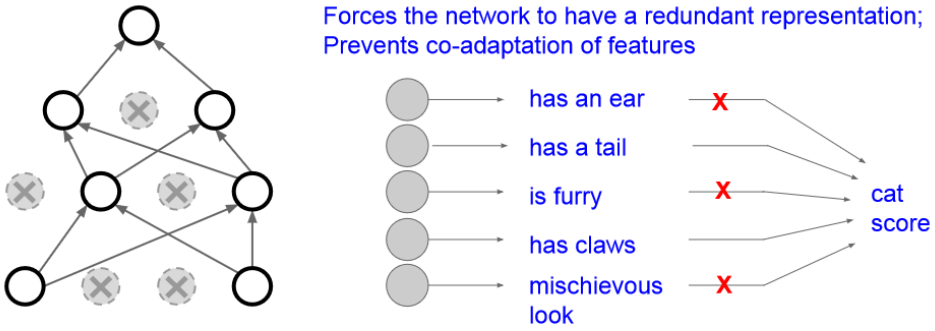
- 드롭아웃dropout 규제 기법
  - 입력층과 은닉층의 노드 중 일정 비율을 임의로 선택하여 제거 (일반적으로 p=0.5 적용)
  - 남은 부분 신경망을 학습



(a) 원래 신경망(4-4-4-2 구조) (b) 드롭아웃된 3개의 신경망 예시

그림 5-27 드롭아웃된 신경망

  - 많은 부분 신경망을 만들고, 예측 단계에서 앙상블 결합하는 기법으로 볼 수 있음



  - 많은 부분 신경망을 학습하고, 저장하고, 앙상블 결합하는데 요구되는 계산 시간과 메모리 공간 측면의 부담

5.4.4 드롭아웃

- 실제로는 가중치 공유 사용
  - 하나의 신경망 (하나의 가중치 집합)에 드롭아웃을 적용함 ([알고리즘 5-8])

알고리즘 5–8 드롭아웃을 채택한 기계 학습 알고리즘

입력: 드롭아웃 비율 p_{input}, p_{hidden}
출력: 최적해 $\hat{\Theta}$

```
1  난수를 생성하여 초기해 Θ를 설정한다.
2  while (! 멈춤 조건) // 수렴 조건
3     미니배치 𝔹를 샘플링한다.
4     for (i=1 to |𝔹|)  // 𝔹의 샘플 각각에 대해
5        입력층은 p_input, 은닉층은 p_hidden 비율로 드롭아웃을 수행한다.
6        드롭아웃된 부분 신경망 Θ_i^dropout 로 전방 계산을 한다.
7        오류 역전파를 이용하여 Θ_i^dropout 를 위한 그레이디언트 ∇_i^dropout 를 구한다.
8     ∇_1^dropout, ∇_2^dropout, ..., ∇_|𝔹|^dropout 의 평균 ∇_ave^dropout 를 계산한다.
9     Θ = Θ − ρ∇_ave^dropout  // 가중치 갱신
10 Θ̂ = Θ
```

5.4.4 드롭아웃

- 라인 6의 전방 계산

l번째 은닉층의 j번째 노드의 연산:

$$z_j^l = \tau_l(s_j^l)$$

이때 $s_j^l = \mathbf{u}_j^l \mathbf{z}^{l-1}$

$\Rightarrow$

드롭아웃 적용:

$$z_j^l = \tau_l(s_j^l)$$

이때 $\begin{cases} \tilde{\mathbf{z}}^{l-1} = \mathbf{z}^{l-1} \odot \boldsymbol{\pi}^{l-1} \\ s_j^l = \mathbf{u}_j^l \tilde{\mathbf{z}}^{l-1} \end{cases}$ (5.35)

- 불린boolean 배열 $\boldsymbol{\pi}$에 노드 제거 여부를 표시
- $\boldsymbol{\pi}$는 샘플마다 독립적으로 정하는데, 난수로 설정함
- 보통 입력층 제거 비율 $P_{input} = 0.2$, 은닉층 제거 비율 $P_{hidden} = 0.5$로 설정

5.4.4 드롭아웃

예측 단계

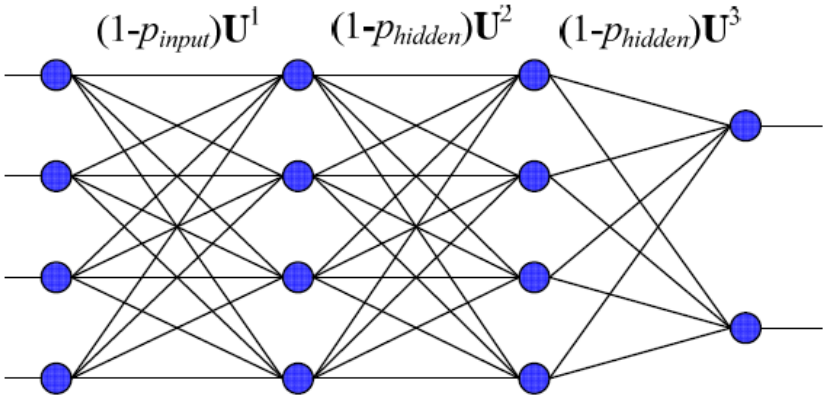


그림 5-28 드롭아웃의 예측 단계

- 앙상블 효과 모방
  - 가중치에 생존 비율 (1-드롭아웃 비율)을 곱하여 전방 계산
  - 학습 과정에서 가중치가 (1-드롭아웃 비율)만큼만 참여했기 때문

메모리와 계산 효율

- 추가 메모리는 불린 배열 **π**, 추가 계산은 작음
- 실제 부담은 신경망의 크기에서 옴: 보통 은닉 노드 수를 $\frac{1}{P_{hidden}}$만큼 늘림

5.4.5 앙상블 기법

- 앙상블ensembles
  - 서로 다른 여러 개의 모델을 결합하여 일반화 오류를 줄이는 기법
  - 현대 기계 학습은 앙상블도 규제로 여김
- 두 가지 일

1. 서로 다른 예측기를 학습하는 일
   - 예, 서로 다른 구조의 신경망 여러 개를 학습 또는 같은 구조를 사용하지만

     서로 다른 초깃값과 하이퍼 매개변수를 설정하고 학습
   - 예, 배깅bagging (bootstrap aggregating) (훈련집합을 여러 번 샘플링하여 서로 다른 훈련집합을 구성)
   - 예, 부스팅boosting (i번째 예측기가 틀린 샘플을 i+1번째 예측기가 잘 인식하도록 연계성을 고려)
2. 학습된 예측기를 결합하는 일 – 모델 평균model averaging
   - 여러 모델의 출력으로부터 평균을 구하거나 투표하여 최종 결과 결정
   - 주로 투표 방식을 사용

- 자세한 내용은 12장

5.5 하이퍼 매개변수 최적화

- 학습 모델에는 두 종류의 매개변수가 있음
  - 내부 매개변수parameter
    - 신경망의 경우 에지 가중치로서 이 책은 $\boldsymbol{\theta}$로 표기 (예, 식 (4.1)의 가중치 행렬 $\mathbf{U}^{(l)}$)
    - 학습 알고리즘이 최적화함
      - 주어진 데이터로부터 결정됨
  - 하이퍼 매개변수hyper-parameter
    - 모델의 외부에서 모델의 동작을 조정함
      - 사람에 의해서 결정됨
    - 예, 은닉층의 개수, CNN의 필터 크기와 보폭, 학습률, 모멘텀과 관련된 매개변수 등

5.5 하이퍼 매개변수 최적화

하이퍼 매개변수 선택

- 표준 참고 문헌이 제시하는 기본값을 사용하면 됨
  - 보통 여러 후보 값 또는 범위를 제시
- 후보 값 중에서 주어진 데이터에 최적인 값 선택 ← 하이퍼 매개변수 최적화

알고리즘 5-9 하이퍼 매개변수 최적화

입력: 훈련집합 $\mathbb{X}$, 검증집합 $\mathbb{X}_{validate}$, 하이퍼 매개변수집합 $\mathcal{H}$
출력: 최적 하이퍼 매개변숫값 $\hat{\mathrm{H}}$

```
1  q_max = 0
2  repeat
3     하이퍼 매개변수 조합 H̃을 생성한다.
4     H̃으로 설정된 모델을 𝕏로 학습한다.
5     학습된 모델을 𝕏_validate로 성능 q를 측정한다.
6     if (q > q_max) q_max = q, Ĥ = H̃
7  until(멈춤 조건)
```

- 라인 3을 구현하는 방법에 따라 수동 탐색, 격자 탐색, 임의 탐색

→ 최근 학습을 통한 자동 탐색 방법들이 연구됨 (예, 자동화된 기계학습automated machine learning)

5.5 하이퍼 매개변수 최적화

- 격자 탐색grid search과 임의 탐색random search

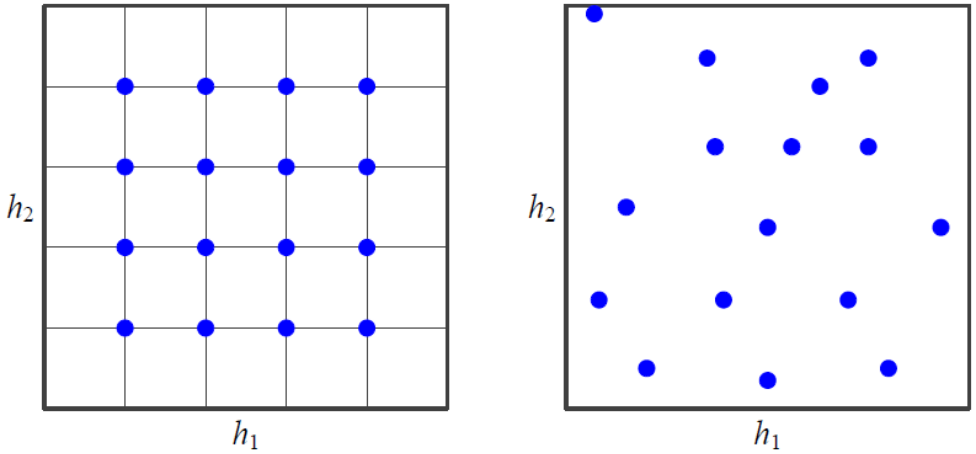


그림 5-29 하이퍼 매개변수 탐색 방법

  - 임의 탐색은 난수로 하이퍼 매개변수 조합을 생성함

- 로그 공간log space 간격으로 탐색
  - 어떤 하이퍼 매개변수는 로그 공간를 사용해야 함
  - 예, 학습률 범위가 [0.0001~0.1]일 때
    - 등 간격은 0.0001, 0.0002, 0.0003, ..., 0.0998, 0.0999로 총 1000개 값을 조사
    - 로그 공간 간격은 2배씩 증가시킴

      즉 0.0001, 0.0002, 0.0004, 0.0008, ..., 0.0256, 0.0512, ...를 조사함

5.5 하이퍼 매개변수 최적화

- 차원의 저주 문제 발생
  - 매개변수가 m개이고 각각이 q개 구간이라면 q^m개의 점을 조사해야 함
- 임의 탐색이 우월함
  - [Bergstra2012]의 실험 (32개의 매개변수) → 임의 탐색이 유리함
  - [그림 5-30]은 임의 탐색이 유리한 이유를 설명

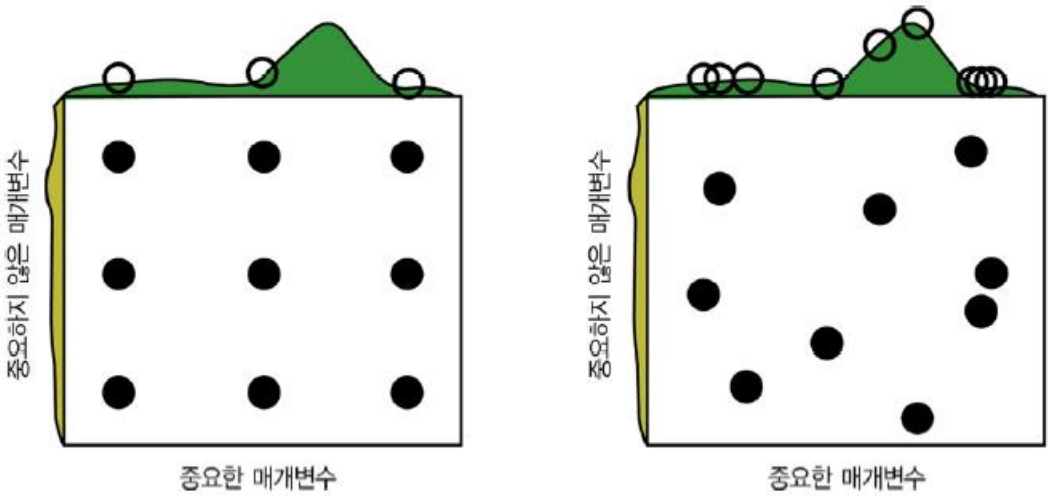


그림 5-30 격자 탐색과 임의 탐색의 성능 비교

Coarse to fine search

```
val_acc: 0.412000, lr: 1.405206e-04, reg: 4.793564e-01, (1 / 100)
val_acc: 0.214000, lr: 7.231888e-06, reg: 2.321281e-04, (2 / 100)
val_acc: 0.208000, lr: 2.119571e-06, reg: 8.011857e+01, (3 / 100)
val_acc: 0.196000, lr: 1.551131e-05, reg: 4.374936e-05, (4 / 100)
val_acc: 0.079000, lr: 1.753300e-05, reg: 1.200424e+03, (5 / 100)
val_acc: 0.223000, lr: 4.215128e-05, reg: 4.196174e+01, (6 / 100)
val_acc: 0.441000, lr: 1.750259e-04, reg: 2.110807e-04, (7 / 100)
val_acc: 0.241000, lr: 6.749231e-05, reg: 4.226413e+01, (8 / 100)
val_acc: 0.482000, lr: 4.296863e-04, reg: 6.642555e-01, (9 / 100)
val_acc: 0.079000, lr: 5.401602e-06, reg: 1.599828e+04, (10 / 100)
val_acc: 0.154000, lr: 1.618508e-06, reg: 4.925252e-01, (11 / 100)
```

```
val_acc: 0.527000, lr: 5.340517e-04, reg: 4.097824e-01, (0 / 100)
val_acc: 0.492000, lr: 2.279484e-04, reg: 9.991345e-04, (1 / 100)
val_acc: 0.512000, lr: 8.680827e-04, reg: 1.349727e-02, (2 / 100)
val_acc: 0.461000, lr: 1.828377e-04, reg: 1.220193e-02, (3 / 100)
val_acc: 0.460000, lr: 1.113730e-04, reg: 5.244389e-02, (4 / 100)
val_acc: 0.498000, lr: 9.477776e-04, reg: 2.001293e-03, (5 / 100)
val_acc: 0.469000, lr: 1.484369e-04, reg: 4.328313e-01, (6 / 100)
val_acc: 0.522000, lr: 5.586261e-04, reg: 2.312685e-04, (7 / 100)
val_acc: 0.530000, lr: 5.000183e-04, reg: 8.259964e-02, (8 / 100)
val_acc: 0.489000, lr: 1.979168e-04, reg: 1.010889e-04, (9 / 100)
val_acc: 0.490000, lr: 2.836931e-04, reg: 2.486271e-03, (10 / 100)
val_acc: 0.475000, lr: 2.021162e-04, reg: 2.287887e-01, (11 / 100)
val_acc: 0.460000, lr: 1.135527e-04, reg: 3.985040e-02, (12 / 100)
val_acc: 0.515000, lr: 6.947668e-04, reg: 1.562888e-02, (13 / 100)
val_acc: 0.531000, lr: 9.471549e-04, reg: 1.433895e-03, (14 / 100)
val_acc: 0.509000, lr: 3.140888e-04, reg: 2.057518e-01, (15 / 100)
val_acc: 0.514000, lr: 6.438349e-04, reg: 3.033781e-01, (16 / 100)
val_acc: 0.502000, lr: 3.921784e-04, reg: 2.787126e-04, (17 / 100)
val_acc: 0.509000, lr: 9.752279e-04, reg: 2.050865e-03, (18 / 100)
val_acc: 0.500000, lr: 2.412048e-04, reg: 4.997821e-04, (19 / 100)
val_acc: 0.466000, lr: 1.319314e-04, reg: 1.189915e-02, (20 / 100)
val_acc: 0.516000, lr: 8.039527e-04, reg: 1.520291e-02, (21 / 100)
```

5.6 2차 미분을 이용한 방법

- 5.6.1 뉴턴 방법
- 5.6.2 켤레 그레이디언트 방법
- 5.6.3 유사 뉴턴 방법

- 그레이디언트 (1차 미분)를 사용하는 경사 하강법
  - 현재 기계 학습의 주류 알고리즘
  - 두 가지 방향의 개선책 [Bottou2017]
    - 그레이디언트의 잡음을 줄임 (예, 미니배치 사용 등)
    - 2차 미분 정보를 활용 → 5.6절의 주제

5.6 2차 미분을 이용한 최적화

경사 하강법 다시 보기

예제 5-2 **경사 하강법의 동작**

변수가 2개인 아래 함수 J의 최저점을 구하는 문제를 생각하자. 먼저 1차 도함수인 그레이디언트 ∇J를 구한다.

$$J(\mathbf{w}) = J(w_1, w_2) = (w_1 - 2)^2 + 4(w_2 - 1)^2$$
$$\nabla J(\mathbf{w}) = \left(\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}\right)^{\mathrm{T}} = (2(w_1 - 2),\ 8(w_2 - 1))^{\mathrm{T}}$$

시작점이 $\mathbf{w}_1 = (4,2)^{\mathrm{T}}$라고 가정하면 $\mathbf{w}_1$에서의 그레이디언트는 $\nabla J(\mathbf{w}_1) = (4,8)^{\mathrm{T}}$이다. 식 (2.58)에 따라 새로운 점 $\mathbf{w}_2$를 계산하면, $\mathbf{w}_2 = (3.2,0.4)^{\mathrm{T}}$가 된다. 이때 학습률 ρ는 0.2라고 가정하자.

$$\mathbf{w}_2 = \mathbf{w}_1 - \rho\nabla J(\mathbf{w}_1) = \begin{pmatrix}4\\2\end{pmatrix} - 0.2\begin{pmatrix}4\\8\end{pmatrix} = \begin{pmatrix}3.2\\0.4\end{pmatrix}$$

$\mathbf{w}_2$에서 그레이디언트는 $\nabla J(\mathbf{w}_2) = (2.4, -4.8)^{\mathrm{T}}$이고, 새로운 점은 $\mathbf{w}_3 = (3.2,0.4)^{\mathrm{T}} - 0.2(2.4, -4.8)^{\mathrm{T}} = (2.72,1.36)^{\mathrm{T}}$가 된다. 비슷한 계산을 반복하면 다음과 같은 궤적을 그리며 최저점 $(2,1)^{\mathrm{T}}$로 접근하는 것을 알 수 있다.

$$\mathbf{w}_1 = \begin{pmatrix}4\\2\end{pmatrix} \rightarrow \mathbf{w}_2 = \begin{pmatrix}3.2\\0.4\end{pmatrix} \rightarrow \mathbf{w}_3 = \begin{pmatrix}2.72\\1.36\end{pmatrix} \rightarrow \mathbf{w}_4 = \begin{pmatrix}2.432\\0.784\end{pmatrix} \rightarrow \mathbf{w}_5 = \begin{pmatrix}2.2592\\1.1296\end{pmatrix} \rightarrow \cdots$$

5.6 2차 미분을 이용한 최적화

- 경사 하강법을 더 빠르게 할 수 있나?
  - [그림 5-31]에서 파란 경로는 경사 하강법이 해를 찾아가는 과정
  - 1차 미분 정보로는 빨간 경로를 알 수 없음

    ∵ 1차 미분은 현재 위치에서 지역적인 기울기 정보만 알려주기 때문

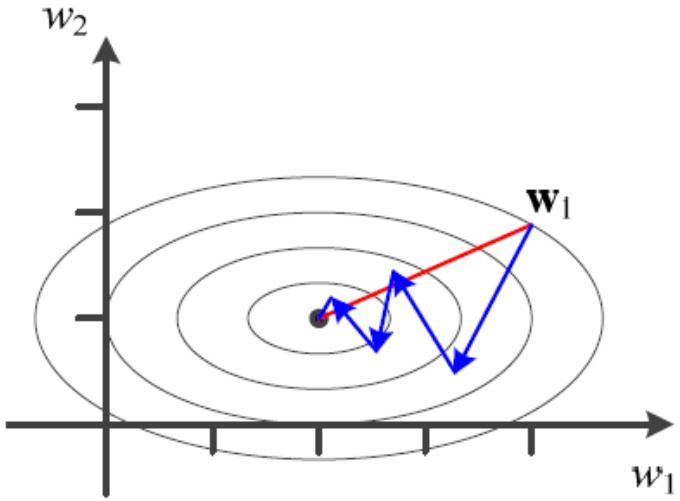


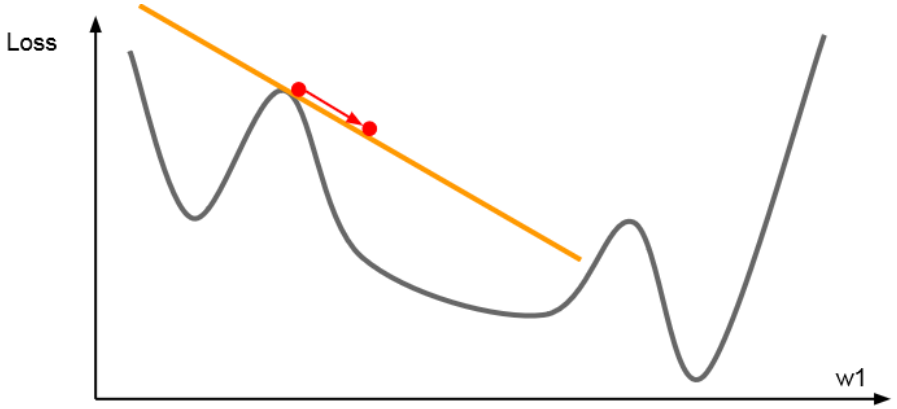
그림 5-31 1차 미분을 사용하는 경사 하강법을 더 빠르게 할 수 있는가

- 뉴턴 방법Newton methods은 2차 미분 정보를 활용하여 빨간 경로를 알아냄

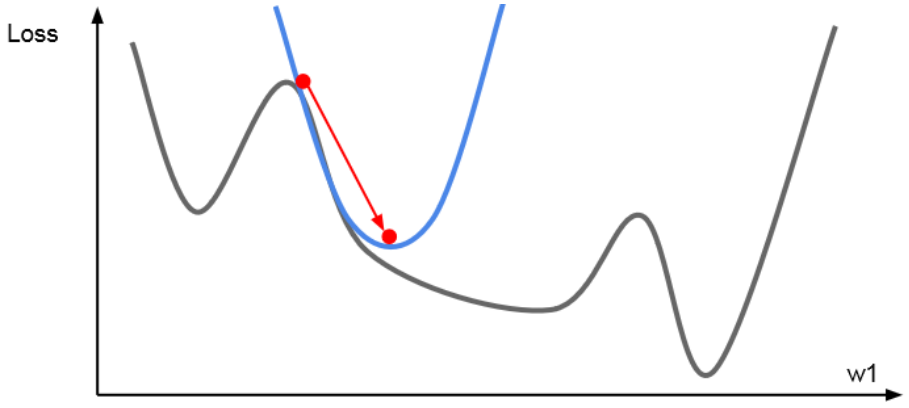
5.6 2차 미분을 이용한 최적화

- 1차 미분 최적화와 2차 미분 최적화 비교

1차 미분의 최적화
- 그레이디언트 사용하여 선형 근사 사용
- 근사치 최소화



2차 미분의 최적화
- 그레이디언트와 헤시안을 사용하여 2차 근사 사용
- 근사치의 최소값



5.6.1 뉴턴 방법

- 테일러 급수를 적용하면,

$$J(w+\delta) \approx J(w) + J'(w)\delta + \frac{J''(w)}{2}\delta^2 \tag{5.36}$$

  - 식 (5.37)은 변수가 여러 개일 때 ($\mathbf{H}$는 헤시언Hessian 행렬)

$$J(\mathbf{w}+\boldsymbol{\delta}) \approx J(\mathbf{w}) + \nabla J^{\mathrm{T}}\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^{\mathrm{T}}\mathbf{H}\boldsymbol{\delta} \tag{5.37}$$

  - δ로 미분하면,

$$\frac{\partial J(w+\delta)}{\partial \delta} \approx J'(w) + \delta J''(w)$$

  - [그림 5-32]처럼 $w+\delta$를 최소점이라 가정하면,

$$\frac{\partial J(w+\delta)}{\partial \delta} \approx J'(w) + \delta J''(w) = 0$$

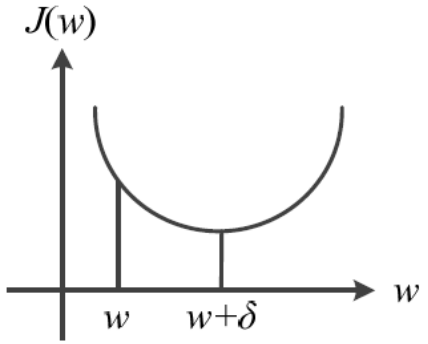


그림 5-32 변수가 하나인 상황의 뉴턴 방법

5.6.1 뉴턴 방법

- 식을 조금 정리하면,

$$\delta = -\frac{J'(w)}{J''(w)} = -\left(J''(w)\right)^{-1} J'(w) \qquad (5.38)$$

- 변수가 여러 개인 경우로 확장하면,

$$\boldsymbol{\delta} = -\mathbf{H}^{-1}\nabla J \qquad (5.39)$$

5.6.1 뉴턴 방법

예제 5-3 뉴턴 방법

[예제 5-2]의 함수 $J(\mathbf{w}) = J(w_1, w_2) = (w_1 - 2)^2 + 4(w_2 - 1)^2$을 다시 활용한다. 그레이디언트와 헤시언 행렬을 구하면 아래와 같다.

$$\nabla J(\mathbf{w}) = \begin{pmatrix} 2w_1 - 4 \\ 8w_2 - 8 \end{pmatrix}$$

$$\mathbf{H} = \begin{pmatrix} 2 & 0 \\ 0 & 8 \end{pmatrix}$$

시작점 $(4,2)^T$를 현재 점 $\mathbf{w}_1$이라고 하면, $\mathbf{w}_1$에서 그레이디언트는 $\nabla J(\mathbf{w}_1) = \begin{pmatrix} 4 \\ 8 \end{pmatrix}$이다. 이것을 식 (5.39)에 대입하면 아래와 같다.

$$\boldsymbol{\delta} = -\begin{pmatrix} 2 & 0 \\ 0 & 8 \end{pmatrix}^{-1} \begin{pmatrix} 4 \\ 8 \end{pmatrix} = -\begin{pmatrix} 1/2 & 0 \\ 0 & 1/8 \end{pmatrix} \begin{pmatrix} 4 \\ 8 \end{pmatrix} = -\begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

다음 점 $\mathbf{w}_2$는 아래 식으로 계산할 수 있고, 이 점은 최저점임을 알 수 있다.

$$\mathbf{w}_2 = \mathbf{w}_1 + \boldsymbol{\delta} = \begin{pmatrix} 4 \\ 2 \end{pmatrix} - \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

5.6.1 뉴턴 방법

뉴턴 방법의 적용

- [예제 5.3]은 2차 함수에 뉴턴 방법을 적용했으므로,
  3차 항 이상을 무시한 식 (5.36)을 사용했음에도 최적 경우 제시
- 기계 학습이 사용하는 목적함수는 2차 함수보다 복잡한 함수이므로
  한 번에 최적해에 도달 불가능
  → [알고리즘 5-10]과 같이 반복하는 뉴턴 방법을 사용해야 함

알고리즘 5-10 뉴턴 방법

입력 : 훈련집합 $\mathbb{X}$, $\mathbb{Y}$
출력 : 최적해 $\hat{\boldsymbol{\Theta}}$

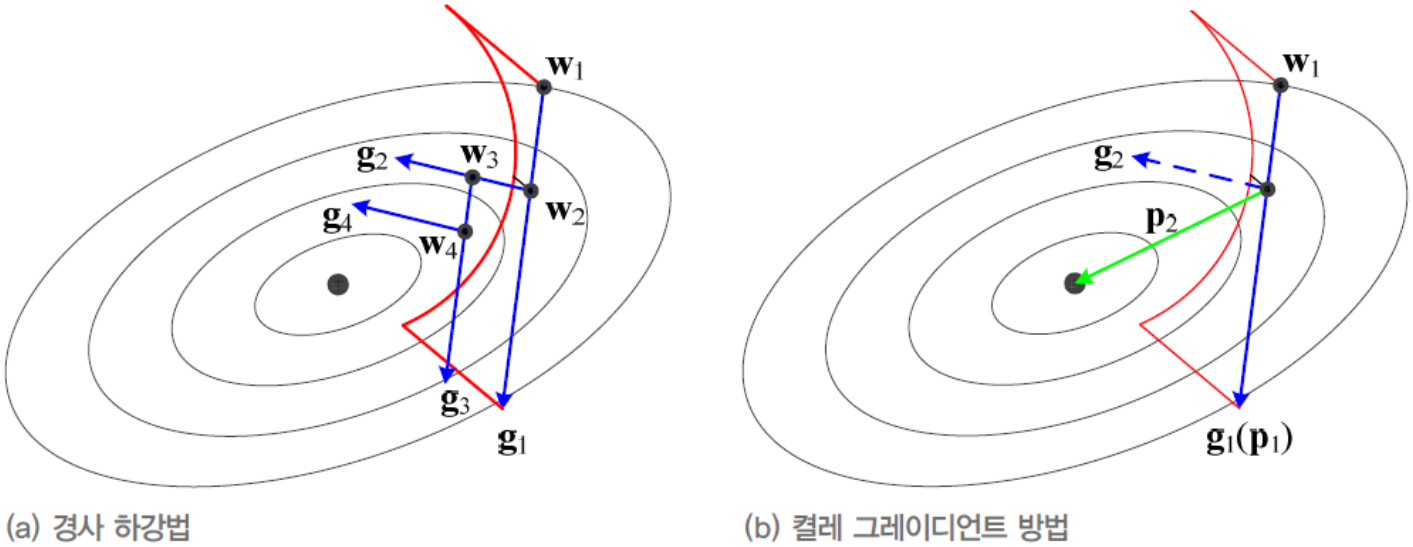
```
1  난수를 생성하여 초기해 Θ를 설정한다.
2  repeat
3     δ = −H^{-1}∇J를 계산한다.    // 식 (5.39)
4     Θ = Θ + δ
5  until (멈춤 조건)
6  Θ̂ = Θ
```

- 라인 3에서 헤시언 $\mathbf{H}$를 구해야 함
  → 매개변수의 개수를 m이라 할 때 $O(m^3)$이라는 과다한 연산량 필요
  → 켤레 그레이디언트 방법이 대안 제시

5.6.2 켤레 그레이디언트 방법

직선 탐색line search

- [그림 5-33(a)]는 경사 하강법이 학습률을 직선 탐색으로 찾는 상황을 예시



(a) 경사 하강법 (b) 켤레 그레이디언트 방법

그림 5-33 경사 하강법과 켤레 그레이디언트 방법의 비교

- 켤레 그레이디언트 방법conjugate gradient method
  - [그림 5-33(a)]의 경사 하강법은 근본적으로 이전 경사 하강법과 같음
    - $\mathbf{g}_2$로 직선 탐색할 때 직전에 사용한 $\mathbf{g}_1$ 정보를 전혀 고려하지 않음
  - [그림 5-33(b)]의 켤레 그레이디언트는 직전 정보를 사용하여 해에 빨리 접근

5.6.2 켤레 그레이디언트 방법

켤레 그레이디언트

- 직전에 사용한 직선이 $\mathbf{p}_{t-1}$이라면
  다음 순간에는 식 (5.40)을 만족하는 $\mathbf{p}_t$를 사용 ($\mathbf{p}_{t-1}$과 $\mathbf{p}_t$를 켤레라 부름)
  그런데 식 (5.40)은 여전히 헤시언 행렬을 포함

$$\mathbf{p}_t^{\mathrm{T}}\mathbf{H}\mathbf{p}_{t-1} = 0 \qquad (5.40)$$

- 켤레 그레이디언트 방법에서는 식 (5.41)로 대치하여 계산
  - $\mathbf{g}_{t-1}$과 $\mathbf{g}_t$는 그레이디언트로서, 그레이디언트를 이용하여 2차 미분 정보를 근사하는 셈

$$\mathbf{p}_t = -\mathbf{g}_t + \beta_t \mathbf{p}_{t-1} \qquad (5.41)$$

$$\text{Polak} - \text{Ribiere: } \beta_t = \frac{(\mathbf{g}_t - \mathbf{g}_{t-1})^{\mathrm{T}}\mathbf{g}_t}{\mathbf{g}_{t-1}^{\mathrm{T}}\mathbf{g}_{t-1}} \qquad (5.42)$$

$$\text{Fletcher} - \text{Reeves: } \beta_t = \frac{\mathbf{g}_t^{\mathrm{T}}\mathbf{g}_t}{\mathbf{g}_{t-1}^{\mathrm{T}}\mathbf{g}_{t-1}} \qquad (5.43)$$

5.6.2 켤레 그레이디언트 방법

알고리즘 5-11 켤레 그레이디언트

입력: 훈련집합 $\mathbb{X}$, $\mathbb{Y}$, β를 리셋하는 주기 q
출력: 최적의 매개변수 $\hat{\boldsymbol{\Theta}}$

1 $\mathbf{p}_0 = \mathbf{0}$ // 라인 탐색에 사용할 벡터를 $\mathbf{0}$ 벡터로 초기화
2 t=1
3 난수를 생성하여 초기해 $\boldsymbol{\Theta}_1$을 설정한다.
4 repeat
5 $\boldsymbol{\Theta}_t$에서 그레이디언트 $\mathbf{g}_t = \frac{\partial J}{\partial \boldsymbol{\Theta}}\Big|_{\boldsymbol{\Theta}_t}$를 구한다.
6 if (t mod q=1) $\beta_t = 0$ // q번에 한 번씩 β를 리셋
7 else $\beta_t = \frac{(\mathbf{g}_t - \mathbf{g}_{t-1})^{\mathrm{T}}\mathbf{g}_t}{\mathbf{g}_{t-1}^{\mathrm{T}}\mathbf{g}_{t-1}}$ // 식 (5.42)
8 $\mathbf{p}_t = -\mathbf{g}_t + \beta_t \mathbf{p}_{t-1}$ // 식 (5.41)
9 $\boldsymbol{\Theta}_t$와 $\boldsymbol{\Theta}_t + \varepsilon \mathbf{p}_t$를 잇는 직선에서 직선 탐색을 수행하여 최소에 해당하는 $\hat{\varepsilon}$을 구한다.
10 $\boldsymbol{\Theta}_{t+1} = \boldsymbol{\Theta}_t + \hat{\varepsilon}\mathbf{p}_t$
11 t++
12 until (멈춤 조건)
13 $\hat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}_t$

식 (5.42)와 식 (5.43)은 2차 함수를 가정하고 유도한 식임
라인 6은 이 때문에 발생하는 오차를 보정하는 역할

5.6.3 유사 뉴턴 방법

- 유사 뉴턴 방법quasi-Newton methods의 기본 아이디어
  - 헤시언 $\mathbf{H}$의 역행렬을 근사하는 행렬 $\mathbf{M}$을 사용
  - 처음에 단위 행렬 $\mathbf{I}$로 시작하여 그레이디언트 정보를 이용하여 점점 개선
  - LFGS가 많이 사용됨
  - 기계 학습에서는 $\mathbf{M}$을 저장하는 메모리를 적게 쓰는limited memory L-BFGS를 주로 사용함
    - 전체 배치를 통한 갱신을 할 수 있다면, L-BFGS 사용을 고려함
- 기계 학습에서 2차 미분 정보의 활용
  - 현재 널리 활용되지는 않지만 연구 계속되고 있음

"… These methods, such as those built on noise reduction and second-order techniques, offer the ability to attain improved convergence rates, overcome the adverse effects of high nonlinearity and ill-conditioning, and exploit parallelism and distributed architectures in new way. … 잡음 감소와 2차 미분과 같은 방법은 수렴 속도를 향상하고, 심한 비선형과 불량 조건 같은 부정적 효과를 극복하며, 새로운 방식으로 병렬분산 계산 구조를 이용하는 길을 제시한다." [Bottou2017]