기계 학습
Machine Learning

Chapter05. 딥러닝 최적화

순천향대학교 AI·빅데이터학과
20211464 민현식
minun001@gmail.com

Chapter05. 딥러닝 최적화 index

Chapter05. 딥러닝 최적화 preview

과학과 공학에서 최적화

예) 웨이퍼에 최대 집적하는 반도체공학, 목적지까지 최단 경로 찾기 등
목적함수를 정의하고 최적해를 구하면 됨

기계 학습의 최적화는 훨씬 복잡함

훈련집합으로 학습을 마친 후, 현장에서 발생하는 새로운 샘플을 잘 예측해야 함. 즉 일반화 능력이 뛰어나야 함
훈련집합이 테스트집합의 대리자 역할을 한다고 말할 수 있음
또한 MSE, 로그우도와 같은 목적함수가 궁극 목표인 정확률의 대리자 역할을 함

기계 학습의 최적화가 어려운 이유

대리자 관계
목적함수의 비볼록 성질, 고차원 특징 공간, 데이터의 희소성 등
긴 시간 소요 (예, GPU 4대 장착한 컴퓨터에서 VGGNet을 훈련하는데 2~3주 소요)

5장은 이런 어려움을 극복하는 여러 가지 효과적인 방안을 제시함

Chapter05. 딥러닝 최적화

5-1. 목적함수: 교차 엔트로피와 로그함수

5.1.1 평균제곱 오차를 다시 생각하기
5.1.2 교차 엔트로피 목적함수
5.1.3 softmax 활성함수와 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?

Chapter05. 딥러닝 최적화

5-1. 목적함수: 교차 엔트로피와 로그함수

5.1.1 평균제곱 오차를 다시 생각하기

평균제곱 오차 목적함수

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

오차가 클수록 e값이 크므로 벌점으로 훌륭함

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

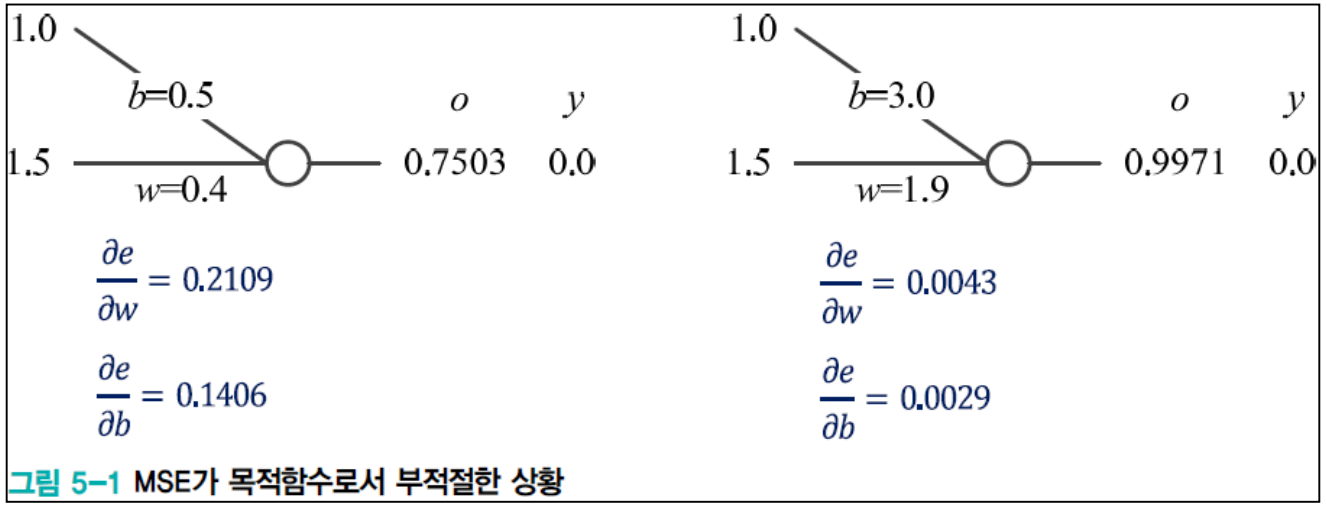


그림 5-1 MSE가 목적함수로서 부적절한 상황

Chapter05. 딥러닝 최적화

5-1. 목적함수: 교차 엔트로피와 로그함수

5.1.1 평균제곱 오차를 다시 생각하기

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

Chapter05. 딥러닝 최적화

5-1. 목적함수: 교차 엔트로피와 로그함수

5.1.1 평균제곱 오차를 다시 생각하기

이유

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

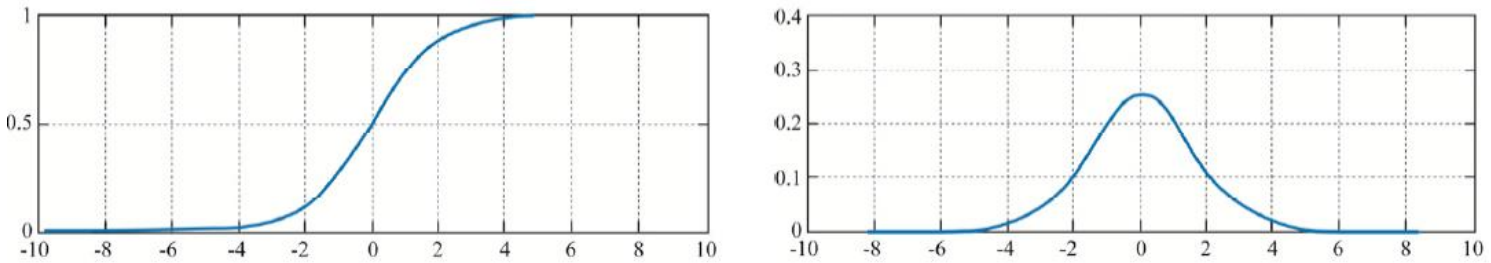


그림 5-2 로지스틱 시그모이드함수와 도함수

Chapter05. 딥러닝 최적화

5-1. 목적함수: 교차 엔트로피와 로그함수

5.1.2 교차 엔트로피 목적함수

교차 엔트로피

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

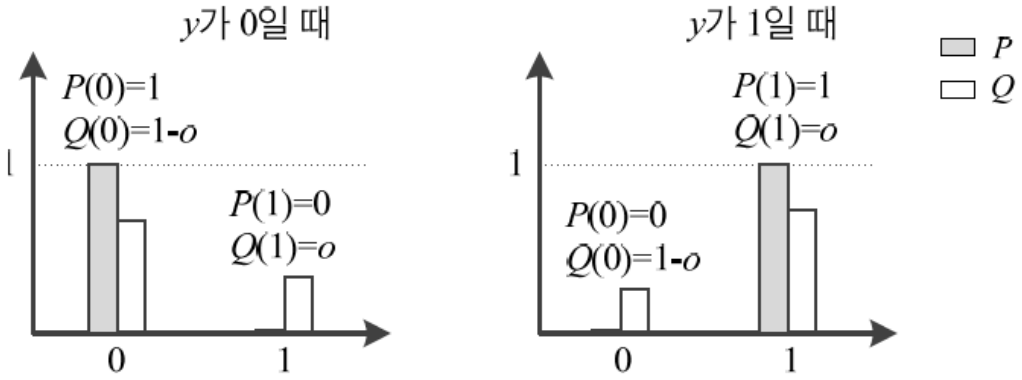


그림 5-3 레이블 y가 0일 때와 1일 때의 P와 Q의 확률분포

확률분포를 통일된 수식으로 쓰면,

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

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

Chapter05. 딥러닝 최적화

5-1. 목적함수: 교차 엔트로피와 로그함수

5.1.2 교차 엔트로피 목적함수

교차 엔트로피 목적함수

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

제구실 하는지 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 엉터리인 경우)

오류 $e = -(1 \log_2 0.0001 + (1-1)\log_2(1-0.0001)) = 13.2877$로서 높은 값

Chapter05. 딥러닝 최적화

5-1. 목적함수: 교차 엔트로피와 로그함수

5.1.2 교차 엔트로피 목적함수

공정한 벌점을 부여하는지 확인 (MSE의 느린 학습 문제를 해결하나?)
도함수를 구하면,

$$\begin{aligned}\frac{\partial e}{\partial w} &= -\left(\frac{y}{o}-\frac{1-y}{1-o}\right)\frac{\partial o}{\partial w}\\ &= -\left(\frac{y}{o}-\frac{1-y}{1-o}\right)x\sigma'(z)\\ &= -x\left(\frac{y}{o}-\frac{1-y}{1-o}\right)o(1-o)\\ &= x(o-y)\end{aligned} \longrightarrow \left.\begin{aligned}\frac{\partial e}{\partial w} &= x(o-y)\\ \frac{\partial e}{\partial b} &= (o-y)\end{aligned}\right\} \quad (5.5)$$

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

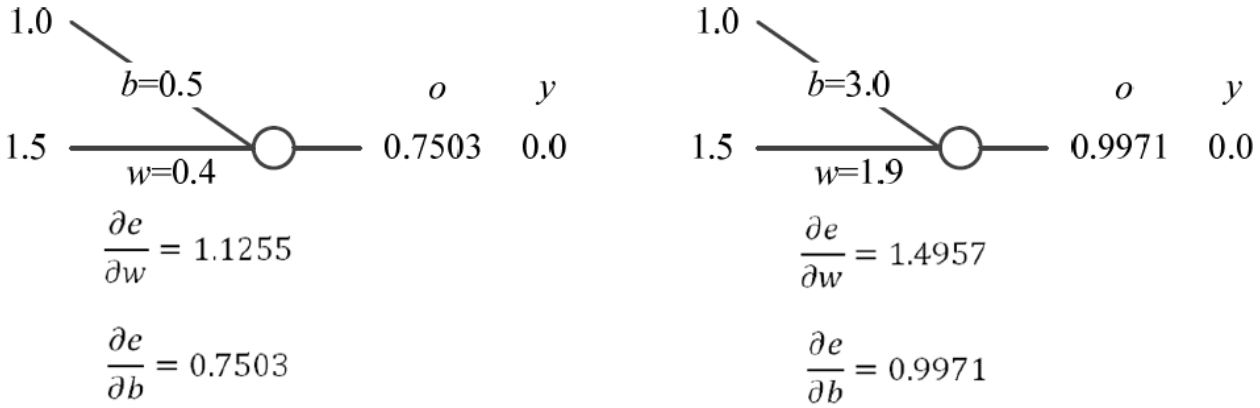


그림 5-4 교차 엔트로피를 목적함수로 사용하여 느린 학습 문제를 해결

Chapter05. 딥러닝 최적화

5-1. 목적함수: 교차 엔트로피와 로그함수

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)

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

Chapter05. 딥러닝 최적화

5-1. 목적함수: 교차 엔트로피와 로그함수

5.1.3 softmax 활성함수와 로그우도 목적함수

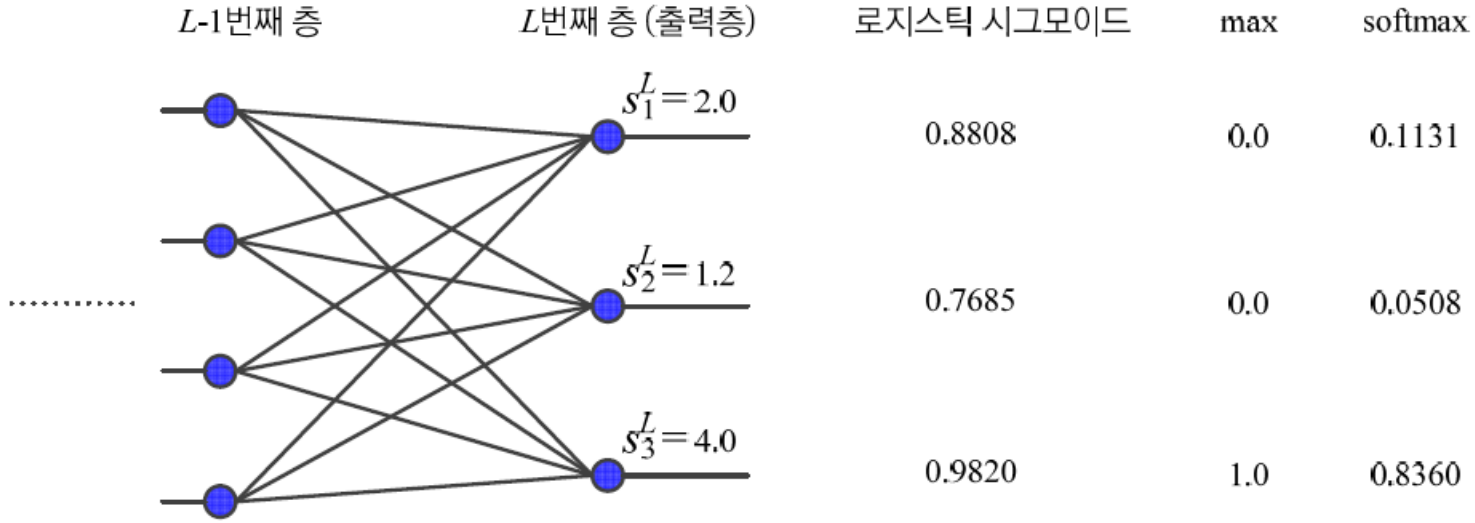


그림 5-5 출력층의 활성함수로 로지스틱 시그모이드와 softmax 비교

softmax 활성함수

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

동작 예시

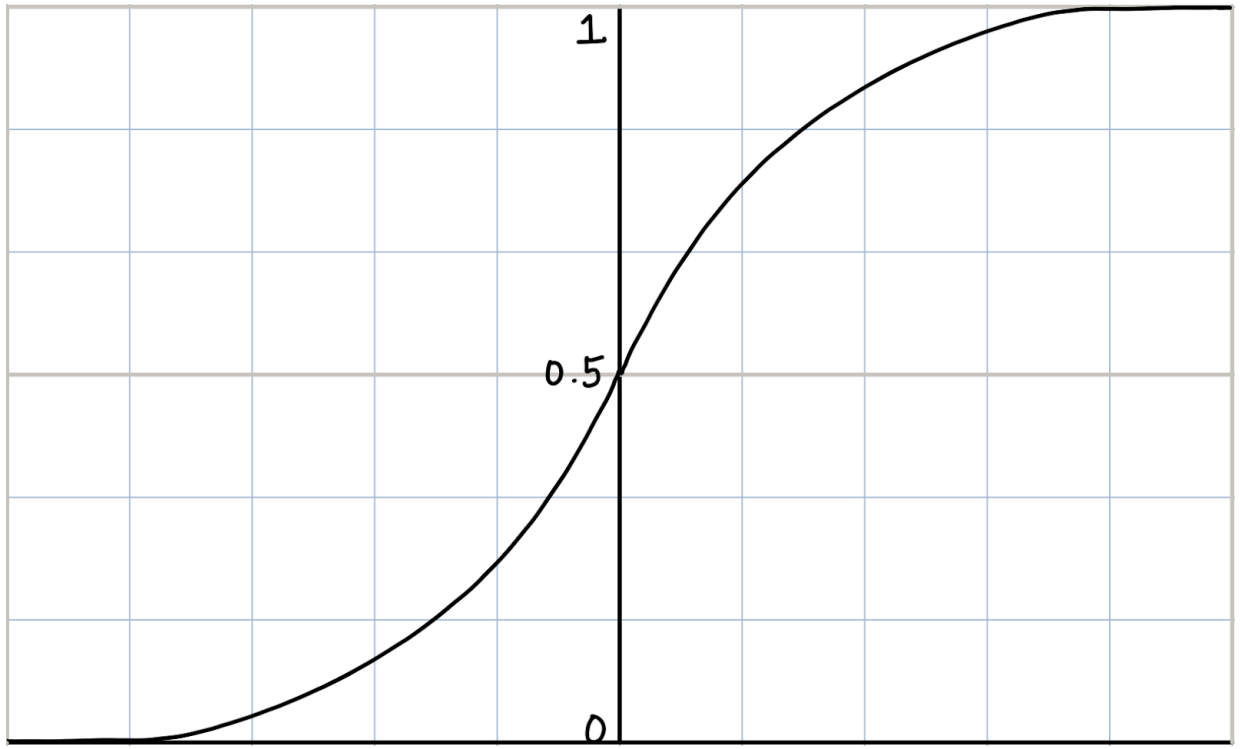
1. softmax는 max를 모방(출력 노드의 중간 계산 결과 s_i^L에서 최댓값은 더욱 활성화하고 작은 값은 억제
2. 모두 더하면 1이 되어 확률 모방

Chapter05. 딥러닝 최적화

5-1. 목적함수: 교차 엔트로피와 로그함수

5.1.3 softmax 활성함수와 로그우도 목적함수

softmax 활성함수



sigmoid function

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

로지스틱 함수

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

sigmoid 시그모이드

↓ 일반화 generalization

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{i=1}^{K} e^{z_i}}$$

Softmax 소프트맥스

Chapter05. 딥러닝 최적화

5-1. 목적함수: 교차 엔트로피와 로그함수

5.1.3 softmax 활성함수와 로그우도 목적함수

$$e = -\log_2 o_y \tag{5.8}$$

로그우도 목적함수

1. 모든 출력 노드값을 사용하는 MSE나 교차 엔트로피와 달리 o_y라는 하나의 노드만 사용

2. 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

Softmax와 로그우도

1.Softmax는 최댓값이 아닌 값을 억제하여 0에 가깝 게 만든다는 의도 내포
2. 학습 샘플이 알려주는 부류에 해당하는 노드만 보겠다는 로그우도와 잘 어울림
3. 따라서 둘을 결합하여 사용하는 경우가 많음

Chapter05. 딥러닝 최적화

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

"… 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."

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.1 데이터 전처리

규모 문제

예) 건강에 관련된 데이터 $(\text{키}(\text{m}), \text{몸무게}(\text{kg}), \text{혈압})^{\text{T}}$

1.885m와 1.525m는 33cm나 차이가 나지만 특징값 차이는 불과 0.33

65.5kg과 45.0kg은 20.5라는 차이

첫 번째와 두 번째 특징은 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

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.1 데이터 전처리

모든 특징이 양수인 경우의 문제

1. [그림 5-6]의 경우 ↑ 표시된 가중치는 모두 증가, ↓ 표시된 가중치는 모두 감소
2. 이처럼 뭉치로 증가 또는 감소하면 최저점을 찾아가는 경로가 갈팡질팡하여 느린 수렴

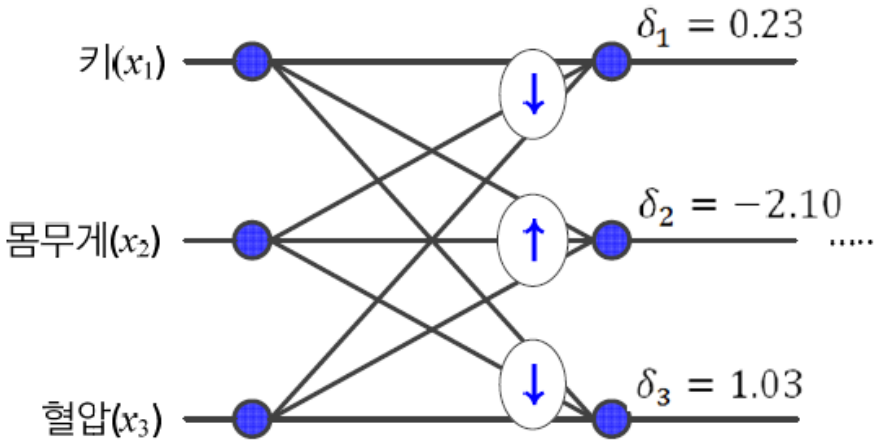


그림 5-6 특징이 모두 양수일 때 가중치가 뭉치로 갱신되는 효과

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.1 데이터 전처리

식 (5.9)의 정규화는 규모 문제와 양수 문제를 해결해줌

특징별로 독립적으로 적용

$$x_i^{new} = \frac{x_i^{old} - \mu_i}{\sigma_i} \tag{5.9}$$

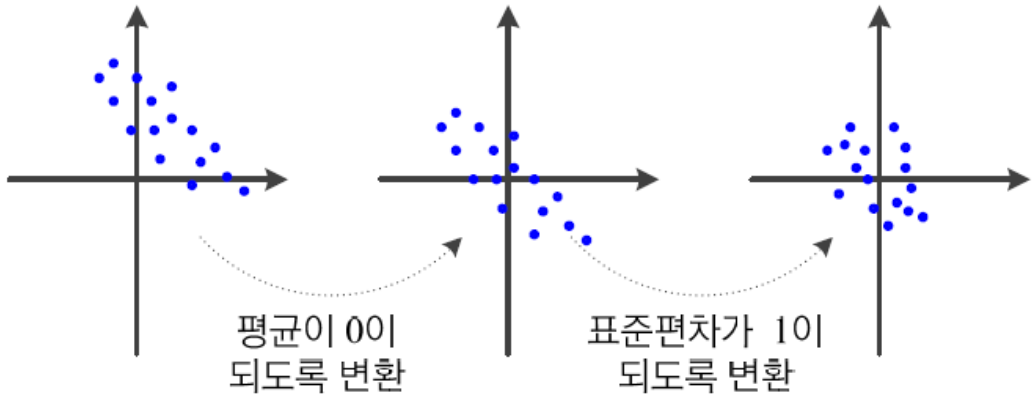


그림 5-7 표준점수로 변환

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.1 데이터 전처리

명칭값을norminal value 원핫one-hot 코드로 변환

1. 예) 성별의 남(1)과 여(2), 체질의 태양인(1), 태음인(2), 소양인(3), 소음인(4)

    1-1. 명칭값은 거리 개념이 없음

2. 원핫 코드는 값의 개수만큼 비트를 부여

    2-1.성별은 2비트, 체질은 4비트 부여

    2-2. 예) 키 1.755m, 몸무게 65.5kg, 혈압 122, 남자, 소양인 샘플

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

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.2 가중치 초기화

대칭적 가중치 문제

1. [그림 5-8]의 대칭적 가중치에서는 z_1^{l-1}과 z_2^{l-1}가 같은 값이 됨. $-\delta_j z_i$가 그레이디언트기 때문에 u_{11}^l과 u_{12}^l이 같은 값으로 갱신됨 → 두 노드가 같은 일을 하는 중복성 발생

2. 난수로 초기화함으로써 대칭 파괴

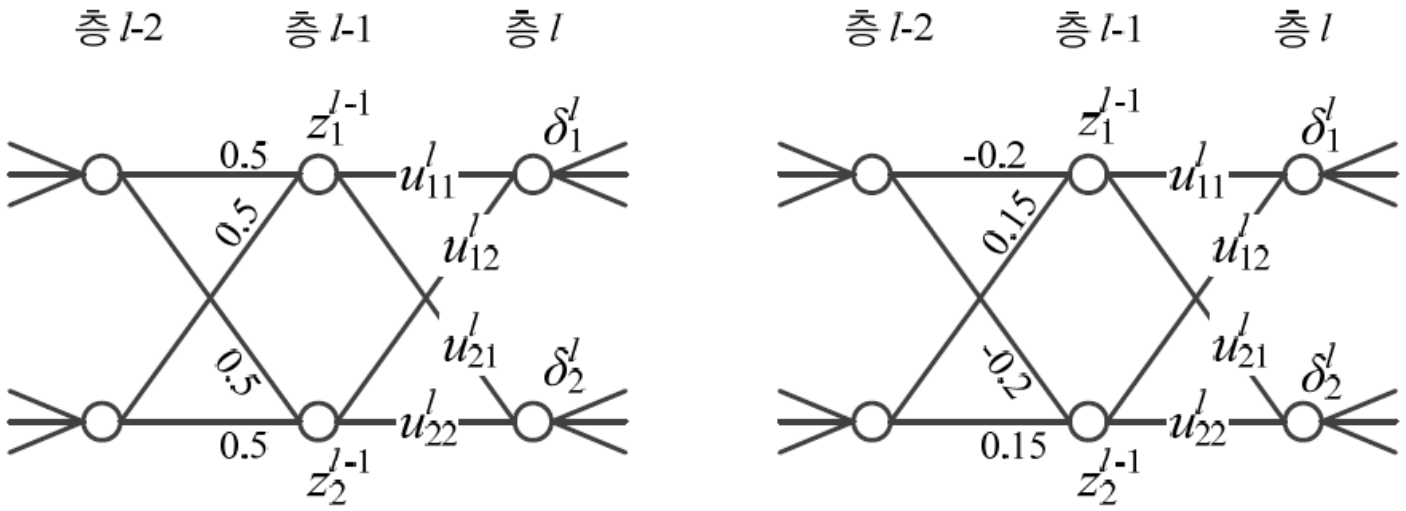


그림 5-8 대칭적 가중치로 초기화된 경우의 중복성 문제

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.2 가중치 초기화

난수로 가중치 초기화

1. 가우시언 분포 또는 균일 분포에서 난수 추출. 두 분포는 성능 차이 거의 없음
2. 난수 범위는 무척 중요함 → 식 (5.10) 또는 식 (5.11)로 r을 결정한 후 $[-r,r]$ 사이에서 난수 발생
3. 바이어스는 보통 0으로 초기화

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

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

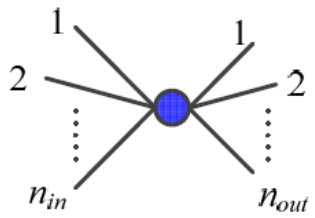


그림 5-9 노드로 들어 오는 에지 개수 n_{in}과 노드에서 나가는 에지 개수 n_{out}

사례

AlexNet [Krizhevsky2012]: 평균 0, 표준편차 0.001인 가우시언에서 난수 생성

ResNet [He2016a]: 평균 0, 표준편차 $\sqrt{\frac{2}{n_{in}}}$인 가우시언에서 난수 생성

또 다른 방법들

1. [Saxe2014]: 가중치 벡터가 수직이 되도록 설정
2. [Sussillo2014]: 임의 행로random walk 활용하여 설정
3. [Sutskever2013]: 가중치 초기화와 모멘텀을 동시에 최적화
4. [Mishkin2016]: 가중치 분포가 아니라 노드의 출력값 분포가 일정하도록 강제화

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.3 모멘텀

그레이디언트의 잡음 현상

1. 기계 학습은 훈련집합을 이용하여 그레이디언트를 추정하므로 잡음 가능성 높음
2. 모멘텀은 그레이디언트에 스무딩을 가하여 잡음 효과 줄임 → 수렴 속도 향상

모멘텀을 적용한 가중치 갱신 수식

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

1. 속도 벡터 $\mathbf{v}$는 이전 그레이디언트를 누적한 것에 해당함(처음에는 $\mathbf{v} = 0$로 출발)
2. α의 효과
   - 2-1. $\alpha = 0$이면 모멘텀이 적용 안 된 이전 공식과 같음
   - 2-2. α가 1에 가까울수록 이전 그레이디언트 정보에 큰 가중치를 주는 셈 → $\boldsymbol{\Theta}$가 그리는 궤적이 매끄러움
   - 2-3. 보통 0.5, 0.9, 또는 0.99 사용 (또는 0.5로 시작하여 세대가 지남에 따라 점점 키워 0.99에 도달하는 방법)

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.3 모멘텀

모멘텀의 효과

오버슈팅 현상 누그러뜨림

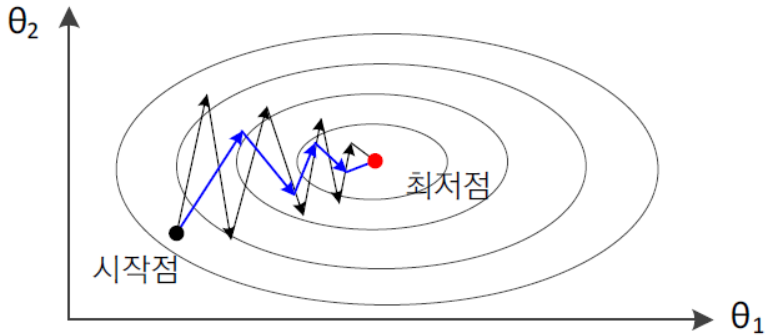


그림 5-10 모멘텀 효과

네스테로프 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

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

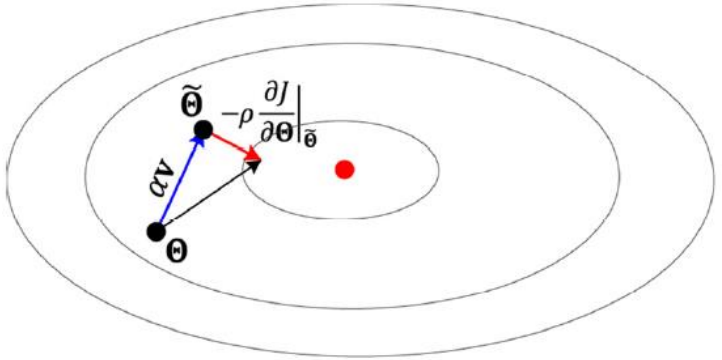


그림 5-11 네스테로프 모멘텀

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.3 모멘텀

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 $\hat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}$

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 $\hat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}$

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.4 적응적 학습률

학습률 ρ의 중요성

너무 크면 오버슈팅에 따른 진자 현상, 너무 작으면 수렴이 느림

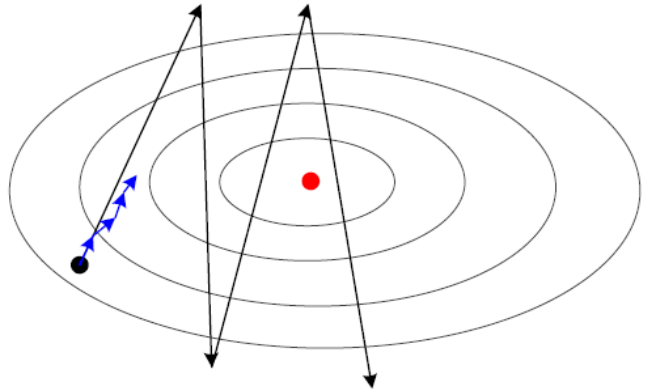


그림 5-12 학습률의 크기에 따른 최적화 알고리즘의 이동 궤적

적응적 학습률

1. 그레이디언트에 학습률 ρ를 곱하면, $\rho\frac{\partial J}{\partial \boldsymbol{\Theta}} = \left(\rho\frac{\partial J}{\partial \theta_1}, \rho\frac{\partial J}{\partial \theta_2}, \cdots, \rho\frac{\partial J}{\partial \theta_k}\right)^{\mathrm{T}}$. 즉 모든 매개변수가 같은 크기의 학습률을 사용하는 셈
2. 적응적 학습률은 매개변수마다 자신의 상황에 따라 학습률을 조절해 사용

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.4 적응적 학습률

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 $\mathbf{r} = \mathbf{r} + \mathbf{g} \odot \mathbf{g}$ // ⊙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 $\hat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}$

AdaGrad

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. $(\theta_1, \theta_2, \cdots, \theta_k)^{\mathrm{T}} = (\theta_1 + \Delta\theta_1, \theta_2 + \Delta\theta_2, \cdots, \theta_k + \Delta\theta_k)^{\mathrm{T}}$

1. $\mathbf{r}$은 이전 그레이디언트를 누적한 벡터
1-1. r_i가 크면 $|\Delta\theta_i|$는 작아서 조금만 이동
1-2. r_i가 작으면 $|\Delta\theta_i|$는 커서 많이 이동
예) [그림 5-10]에서 θ_1은 θ_2보다 보폭이 큼

2. 라인 6의 $\frac{\rho}{\epsilon + \sqrt{r_i}}$는 상황에 따라 보폭을 정해주는 적응적 학습률로 볼 수 있음

https://dodonam.tistory.com/134

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.4 적응적 학습률

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 $\hat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}$

RMSProp

1. AdaGrad의 단점
   1-1. [알고리즘 5-3]의 라인 5는 단순히 제곱을 더함
   1-2. 따라서 오래된 그레이디언트와 최근 그레이디언트는 같은 비중의 역할 → $\mathbf{r}$이 점점 커져 수렴 방해할 가능성

RMSProp은 가중 이동 평균 기법 적용

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

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

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.4 적응적 학습률

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 $\hat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}$

Adam

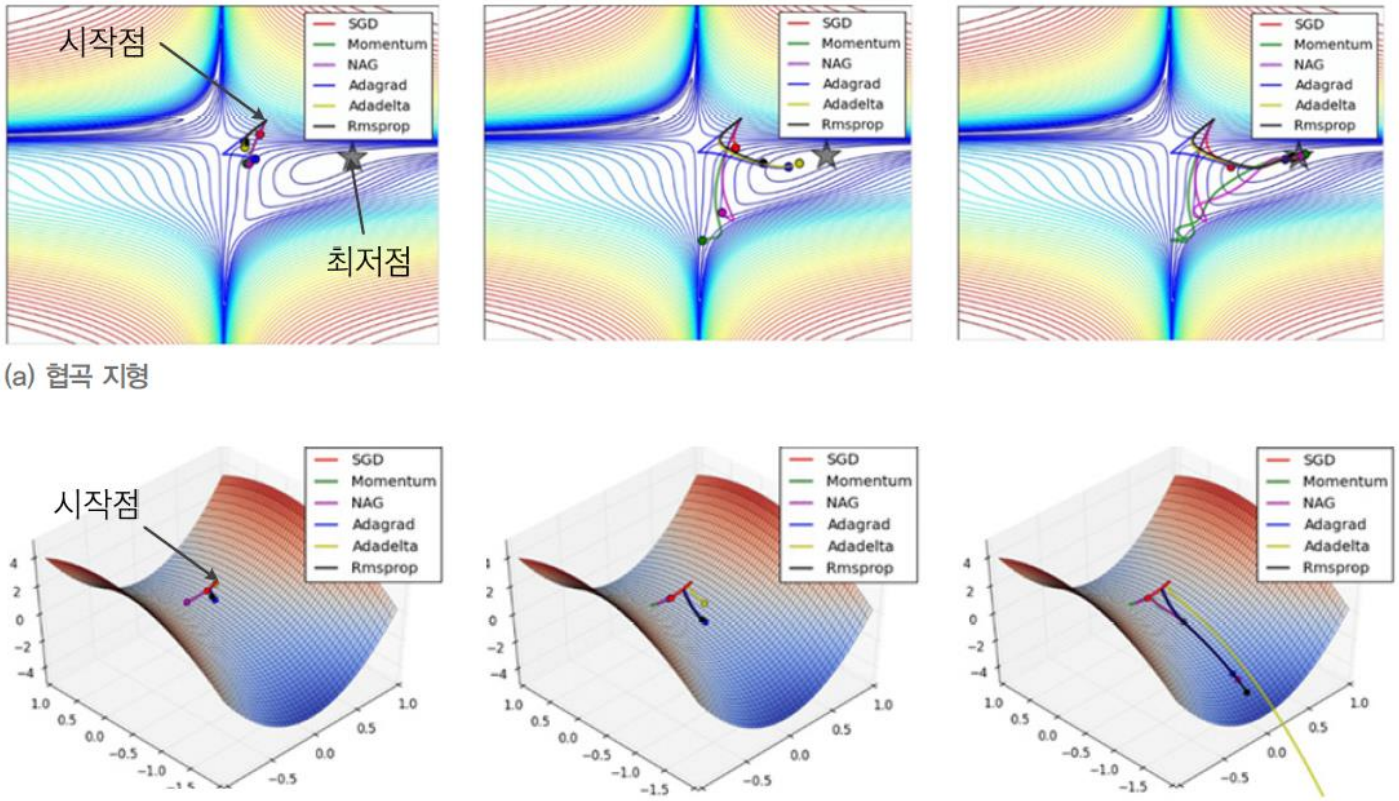
RMSProp에 식 (5-12)의 모멘텀을 추가로 적용한 알고리즘

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.4 적응적 학습률

실시간 애니메이션 http://cs231n.github.io/neural-networks-3



(a) 협곡 지형

(b) 안장점 지형

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

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

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.5 활성 함수

활성값 z를 계산하고 활성함수 τ를 적용하는 과정

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

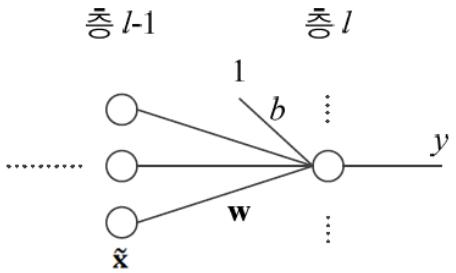


그림 5-14 신경망 노드의 연산

시대별 활성함수

tanh는 활성값이 커지면 포화 상태가 되고 그레이디언트는 0에 가까워짐 →

매개변수 갱신(학습)이 매우 느린 요인

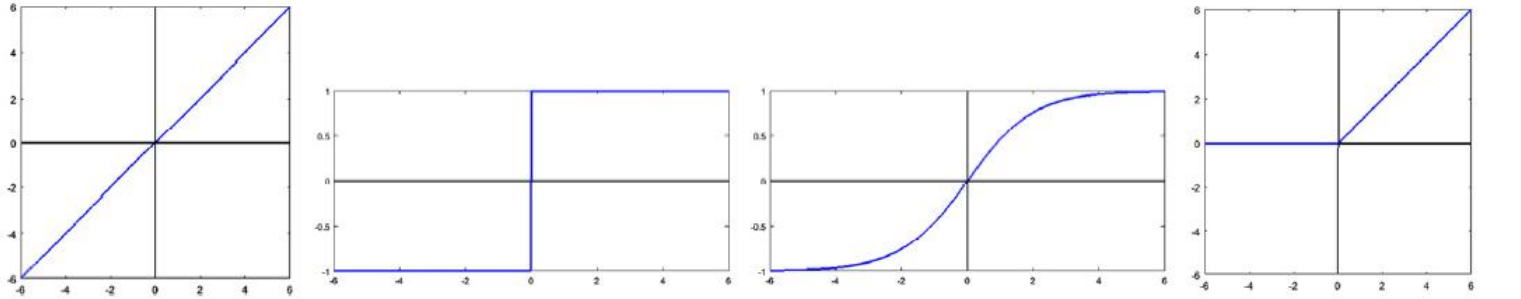


그림 5-15 활성함수 τ

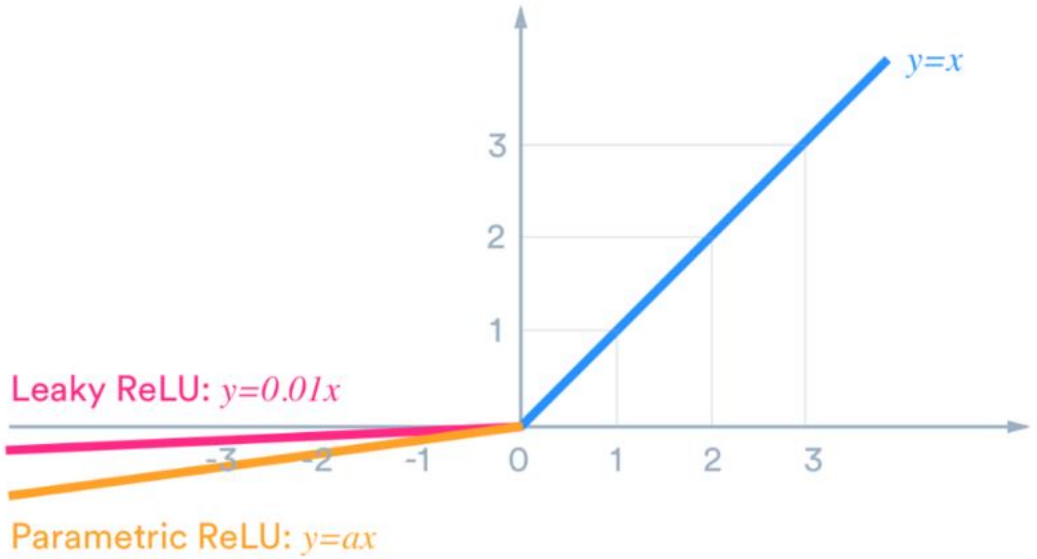
Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.5 활성 함수

ReLU(Rectified Linear Unit) 활성함수
포화 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)

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

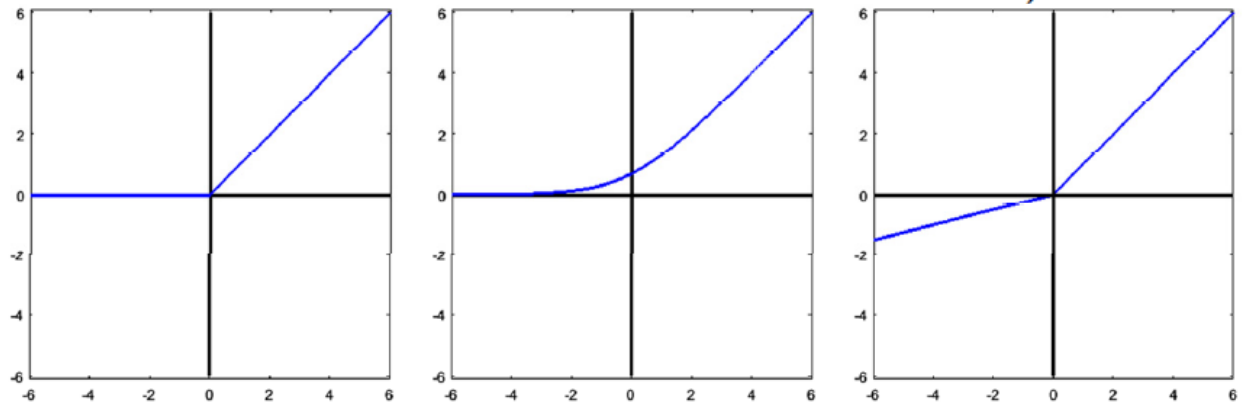


그림 5-16 ReLU의 변형

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.6 배치 정규화

공변량 시프트covariate shift 현상

1. 학습이 진행되면서 층1의 매개변수가 바뀜에 따라 $\tilde{\mathbb{x}}^{(1)}$이 따라 바뀜 → 층2 입장에서 보면 자신에게 입력되는 데이터의 분포가 수시로 바뀌는 셈
2. 층2, 층3, ...으로 깊어짐에 따라 더욱 심각
3. 학습을 방해하는 요인으로 작용

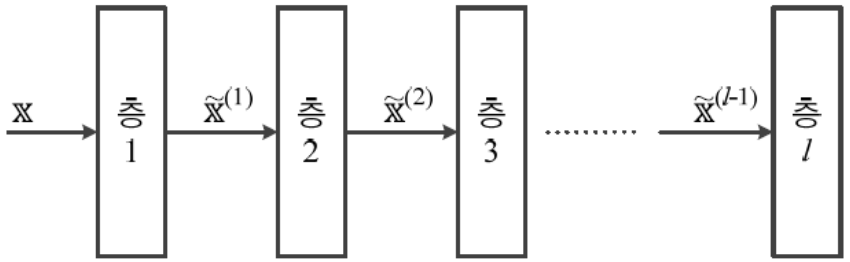


그림 5-17 공변량 시프트 현상

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.6 배치 정규화

배치 정규화

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

$$x_i^{new} = \frac{x_i^{old} - \mu_i}{\sigma_i} \tag{5.9}$$

2. 정규화를 적용하는 곳이 중요

   식 (5.15)의 연산 과정 중 식 (5.9)를 어디에 적용하나?

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

   입력 $\tilde{\mathbf{x}}$ 또는 중간 결과 z 중 어느 것에 적용? → z에 적용하는 것이 유리

3. 훈련집합 전체 또는 미니배치 중 어느 것에 적용?

   미니배치에 적용하는 것이 유리

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.6 배치 정규화

정규화 변환을 수행하는 코드

1. 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
2. 노드마다 독립적으로 코드 1을 수행
3. α와 β는 노드마다 고유한 매개변수로서 학습으로 알아냄

코드 1:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} z_i$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (z_i - \mu_B)^2$$

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

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

1. 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

2. 예측 단계
각 노드는 독립적으로 식 (5.18)을 적용(코드 1의 마지막 두 라인을 수행하는 셈)

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

Chapter05. 딥러닝 최적화

5.2. 성능 향상을 위한 요령

5.2.6 배치 정규화

1. CNN에서는
    1-1. 노드 단위가 아니라 특징 맵 단위로 코드 1과 코드 2를 적용

2. 배치 정규화의 긍정적 효과를 측정한 실험사례 [Ioffe2015]
    2-1. 가중치 초기화에 덜 민감함
    2-2. 학습률을 크게 하여 수렴 속도 향상 가능
    2-3. 시그모이드를 활성함수로 사용하는 깊은 신경망도 학습이 이루어짐
3. 배치 정규화는 규제 효과를 제공
    3-1. 드롭아웃이라는 규제 기법을 적용하지 않아도 높은 성능

Chapter05. 딥러닝 최적화

5.3 규제의 필요성과 원리

5.3.1 과잉적합에 빠지는 이유와 과잉적합을 피하는 전략

학습 모델의 용량에 따른 일반화 능력

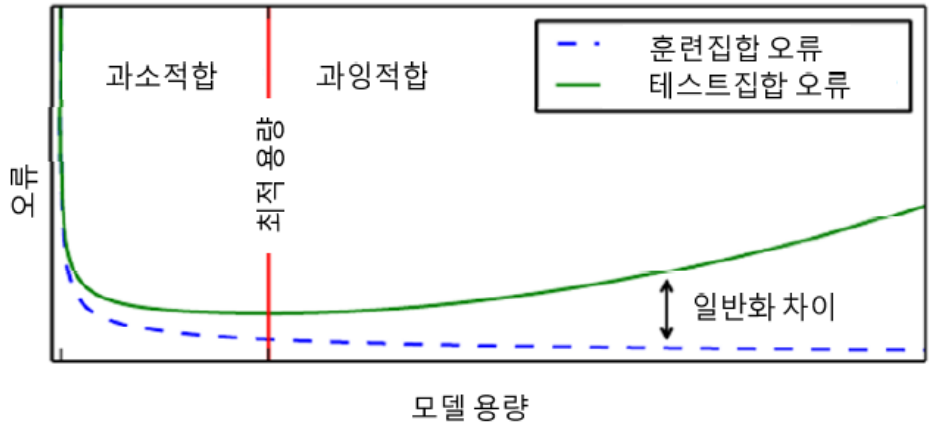


그림 5-18 학습 모델의 용량과 일반화 능력의 관계

Chapter05. 딥러닝 최적화

5.3 규제의 필요성과 원리

5.3.1 과잉적합에 빠지는 이유와 과잉적합을 피하는 전략

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

Chapter05. 딥러닝 최적화

5.3 규제의 필요성과 원리

5.3.2 규제의 정의

규제는 오래 전부터 수학과 통계학에서 연구해온 주제

1. 모델 용량에 비해 데이터가 부족한 경우의 불량 문제를ill-posed problem 푸는 데 사용
2. 적절한 가정을 투입하여 문제를 품 → 입력과 출력 사이의 매핑은 매끄럽다는 사전 지식

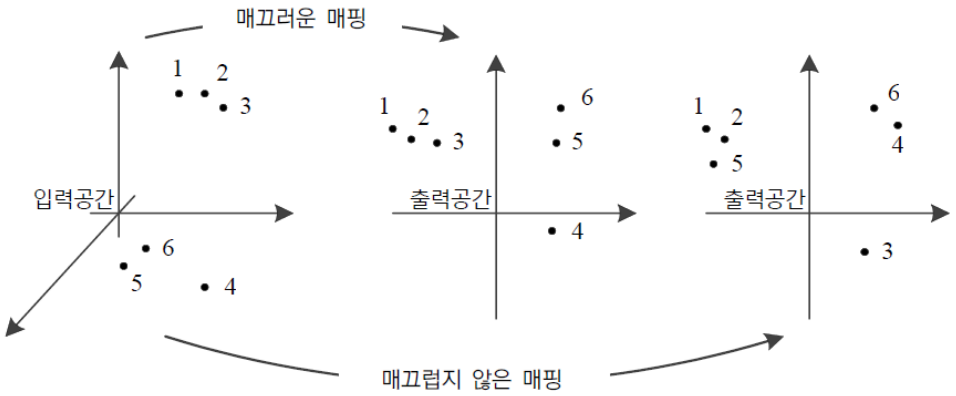


그림 5-20 사전 지식으로서 매끄러움의 특성

3. 티호노프의 규제 기법는 매끄러움 가정에 기반을 둔 식 (5.19)를 사용

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

Chapter05. 딥러닝 최적화

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

1. 명시적 규제: 가중치 감쇠나 드롭아웃처럼 목적함수나 신경망 구조를 직접 수정하는 방식

2. 암시적 규제: 조기 멈춤, 데이터 증대, 잡음 추가, 앙상블처럼 간접적으로 영향을 미치는 방식

5.4 규제 기법

5.4.1 가중치 벌칙

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

식 (5.19)를 관련 변수가 드러나도록 다시 쓰면,

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

1. 규제항은 훈련집합과 무관하며, 데이터 생성 과정에 내재한 사전 지식에 해당
2. 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

Chapter05. 딥러닝 최적화

5.4 규제 기법

5.4.1 가중치 벌칙

$L2$ 놈

규제 항 R로 $L2$ 놈을 사용하는 규제 기법을 '가중치 감쇠'라weight decay 부름 → 식 (5.21)

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

식 (5.21)의 그레이디언트 계산

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

5.4 규제 기법

5.4.1 가중치 벌칙

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

식 (5.22)를 이용하여 매개변수를 갱신하는 수식

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

$\lambda = 0$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)

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

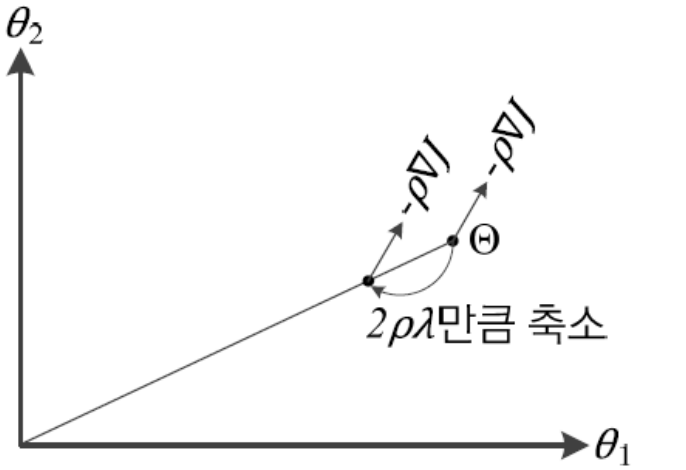


그림 5-21 L2 놈을 사용한 가중치 감쇠 기법의 효과

Chapter05. 딥러닝 최적화

5.4 규제 기법

5.4.1 가중치 벌칙

선형 회귀에 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,

$$\mathbf{Xw} = \mathbf{y} \tag{5.25}$$

가중치 감쇠를 적용한 목적함수

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

5.4 규제 기법

5.4.1 가중치 벌칙

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

식 (5.27)을 미분하여 0으로 놓으면,

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

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.

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

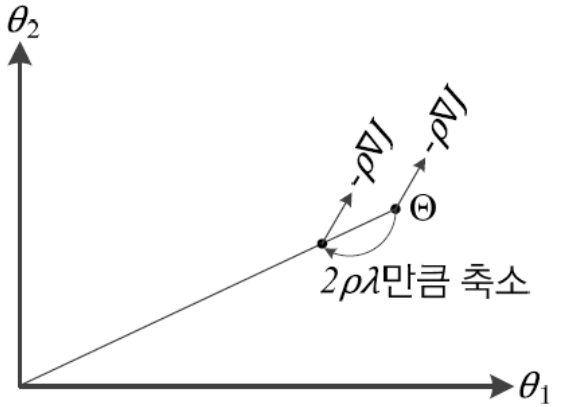


그림 5-21 L2 놈을 사용한 가중치 감쇠 기법의 효과

5.4 규제 기법

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 규제 기법

5.4.1 가중치 벌칙

MLP와 DMLP에 적용

식 (3.21)에 식 (5.23)의 가중치 감쇠라는 규제 기법을 적용하면

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

5.4 규제 기법

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,

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

Chapter05. 딥러닝 최적화

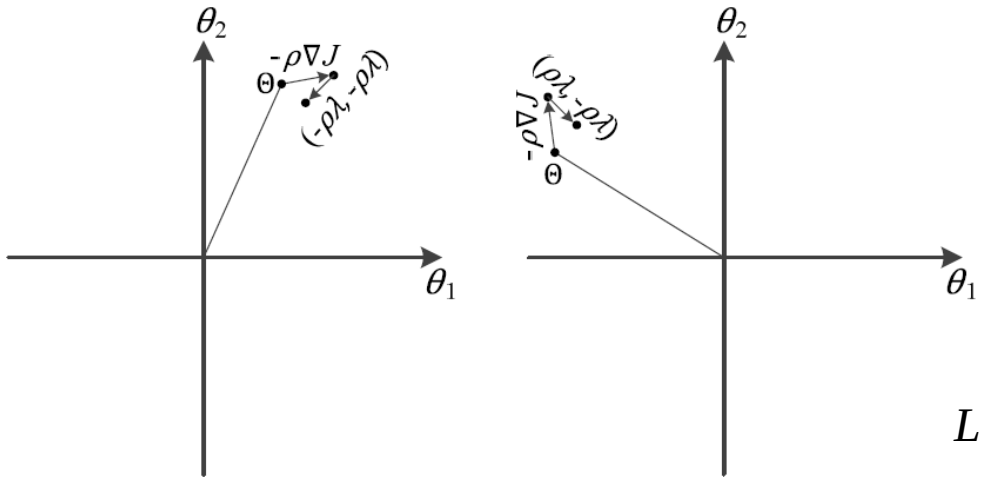
5.4 규제 기법

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 희소성 효과(0이 되는 매개변수가 많음)

선형 회귀에 적용하면 특징 선택 효과

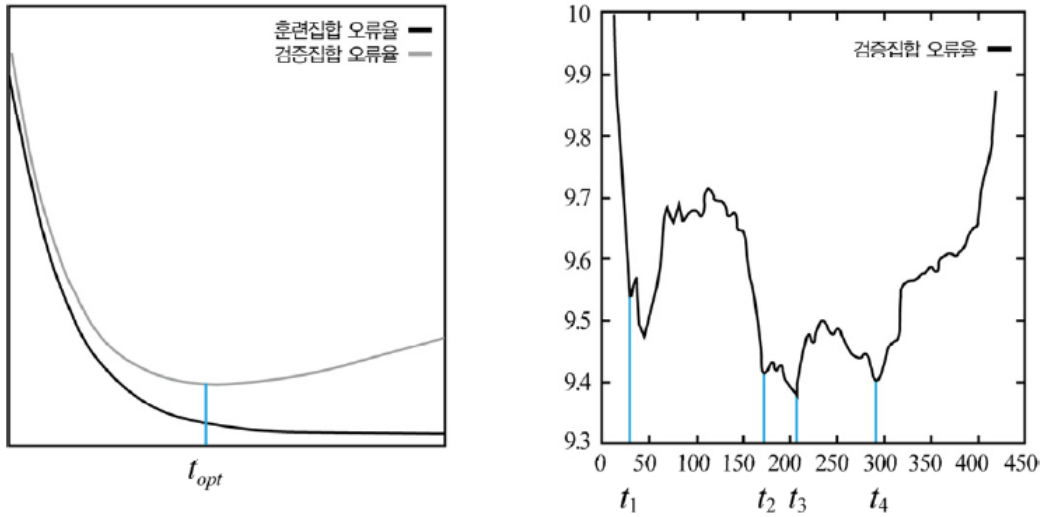
Chapter05. 딥러닝 최적화

5.4 규제 기법

5.4.2 조기 멈춤

학습 시간에 따른 일반화 능력 [그림 5-23(a)]

1. 일정 시간(t_{opt})이 지나면 과잉적합 현상이 나타남 → 일반화 능력 저하
2. 즉 훈련 데이터를 단순히 암기하기 시작



(a) 개념적인 도표 (b) 실제 데이터에 나타나는 지그재그 현상

그림 5-23 학습 시간에 따른 성능 추이

조기 멈춤이라는 규제 기법

검증집합의 오류가 최저인 점 t_{opt}에서 학습을 멈춤

5.4 규제 기법

5.4.2 조기 멈춤

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
```

Chapter05. 딥러닝 최적화

5.4 규제 기법

5.4.2 조기 멈춤

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 Θ_0을 설정한다.
2  Θ̂ = Θ_0, t̂ = 0
3  t = 0, ê = 1.0, j = 0
4  while (j<p)
5    학습 알고리즘의 세대를 q번 반복하여 Θ_{t+q}를 얻는다.
6    Θ_{t+q}로 검증집합에 대한 오류율 e_{t+q}를 측정한다.
7    if (e_{t+q} < ê) // 새로운 최적을 발견한 상황
8      j = 0   // 참는 과정을 처음부터 새로 시작
9      Θ̂ = Θ_{t+q}, ê = e_{t+q}, t̂ = t + q
10   else
11     j=j+1
12   t=t+q
```

Chapter05. 딥러닝 최적화

5.4 규제 기법

5.4.3 데이터 확대

과잉적합 방지하는 가장 확실한 방법은 큰 훈련집합 사용

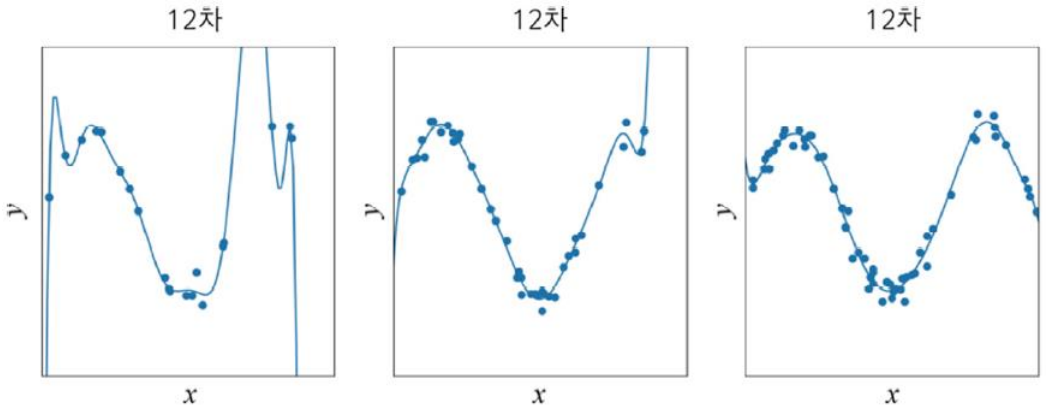


그림 1-17 데이터를 확대하여 일반화 능력을 향상함

하지만 데이터 수집은 비용이 많이 드는 작업

Chapter05. 딥러닝 최적화

5.4 규제 기법

5.4.3 데이터 확대

데이터 확대라는 규제 기법

1. 데이터를 인위적으로 변형하여 확대함
2. 자연계에서 벌어지는 잠재적인 변형을 프로그램으로 흉내 내는 셈

예) MNIST에 어파인 변환(이동, 회전, 크기)을 적용

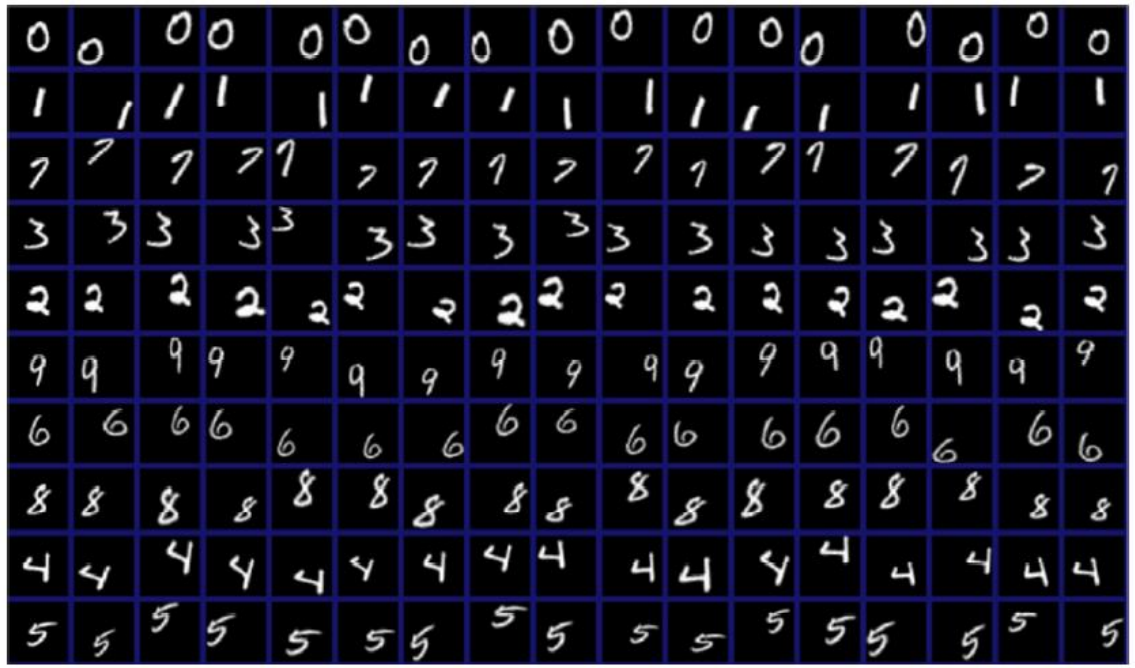


그림 5-24 필기 숫자 데이터의 다양한 변형[8]

한계

1. 수작업 변형
2. 모든 부류가 같은 변형 사용

Chapter05. 딥러닝 최적화

5.4 규제 기법

5.4.3 데이터 확대

예) 모핑을 이용한 변형 [Hauberg2016]

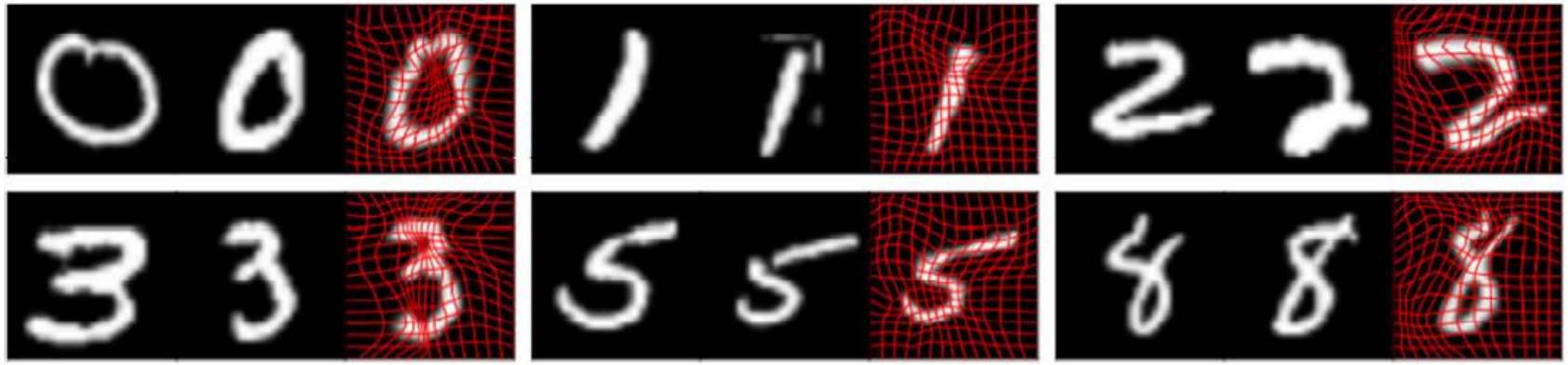


그림 5-25 비선형 변환 학습

1. 비선형 변환으로서 어파인 변환에 비해 훨씬 다양한 형태의 확대
2. 학습 기반: 데이터에 맞는 '비선형 변환 규칙을 학습'하는 셈

Chapter05. 딥러닝 최적화

5.4 규제 기법

5.4.3 데이터 확대

예) 자연영상 확대 [Krizhevsky2012]

1. 256*256 영상에서 224*224 영상을 1024장 잘라내어 이동 효과. 좌우 반전까지 시도하여 2048배로 확대
2. PCA를 이용한 색상 변환으로 추가 확대
3. 예측 단계에서는 [그림 5-26]과 같이 5장 잘라내고 좌우 반전하여 10장을 만든 다음 앙상블 적용

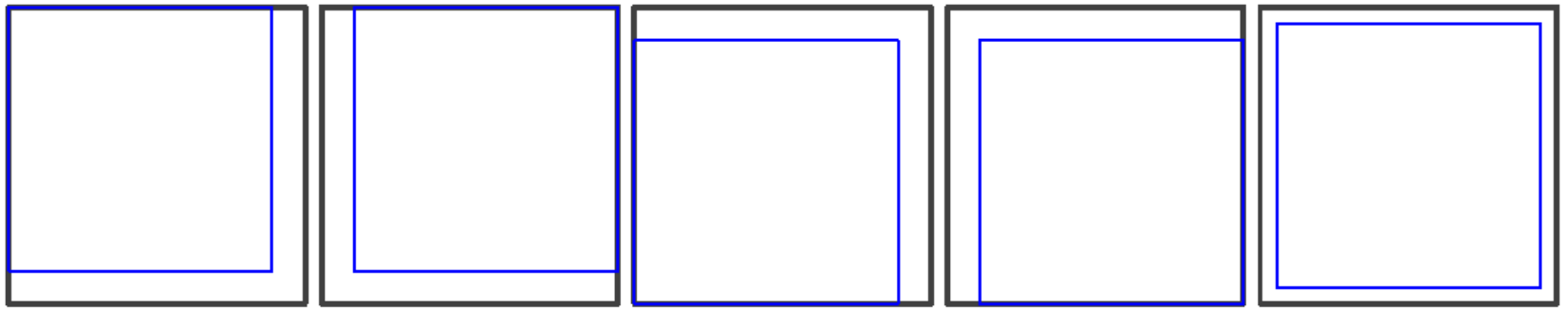


그림 5-26 예측 단계에서 영상 잘라내기

예) 잡음을 섞어 확대하는 기법

1. 입력 데이터에 잡음을 섞는 기법
2. 은닉 노드에 잡음을 섞는 기법 (고급 특징 수준에서 데이터를 확대하는 셈)

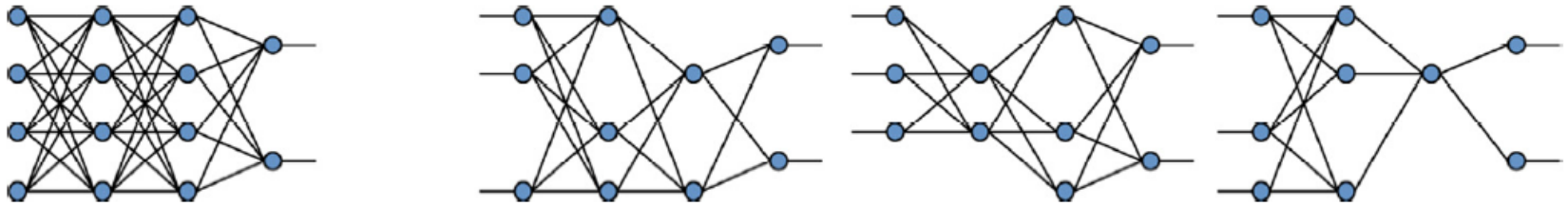
Chapter05. 딥러닝 최적화

5.4 규제 기법

5.4.4 드롭아웃

드롭아웃 규제 기법

1. 입력층과 은닉층의 노드 중 일정 비율을 임의로 선택하여 제거
2. 남은 부분 신경망을 학습
3. 많은 부분 신경망을 만들고, 예측 단계에서 앙상블 결합하는 기법으로 볼 수 있음



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

그림 5-27 드롭아웃된 신경망

많은 부분 신경망을 학습하고, 저장하고, 앙상블 결합하는 데 따른 계산 시간과 메모리 공간 측면의 부담

5.4 규제 기법

5.4.4 드롭아웃

알고리즘 5-8 드롭아웃을 채택한 기계 학습 알고리즘

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

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

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)

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

Chapter05. 딥러닝 최적화

5.4 규제 기법

5.4.4 드롭아웃

예측 단계

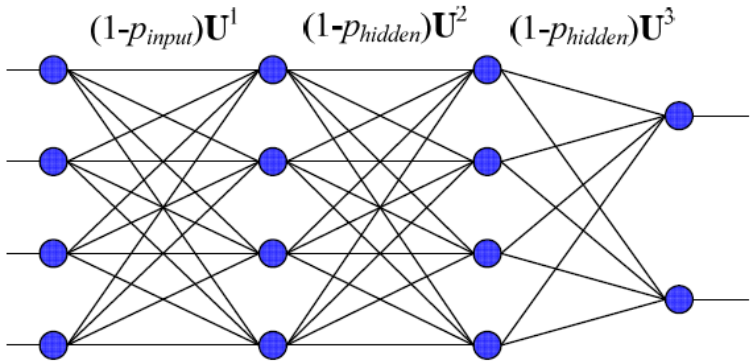


그림 5-28 드롭아웃의 예측 단계

앙상블 효과 모방

1. 가중치에 생존 비율 (1-드롭아웃 비율)을 곱하여 전방 계산
2. 학습 과정에서 가중치가 (1-드롭아웃 비율)만큼만 참여했기 때문

메모리와 계산 효율

1. 추가 메모리는 불린 배열 $\boldsymbol{\pi}$, 추가 계산은 작음
2. 실제 부담은 신경망의 크기에서 옴: 보통 은닉 노드 수를 $\frac{1}{P_{hidden}}$만큼 늘림

Chapter05. 딥러닝 최적화

5.4 규제 기법

5.4.5 앙상블 기법

앙상블

1. 서로 다른 여러 개의 모델을 결합하여 일반화 오류를 줄이는 기법
2. 현대 기계 학습은 앙상블도 규제로 여김

두 가지 일

1. 서로 다른 예측기를 학습하는 일
    1-1. 예, 서로 다른 구조의 신경망 여러 개를 학습 또는 같은 구조를 사용하되 서로 다른 초깃값과 하이퍼 매개변수를 설정하고 학습
    1-2. 예, 배깅(훈련집합을 여러 번 샘플링하여 서로 다른 훈련집합을 구성)
    1-3. 예, 부스팅(i번째 예측기가 틀린 샘플을 $i+1$번째 예측기가 잘 인식하도록 연계성을 고려)
2. 학습된 예측기를 결합하는 일
    2-1. 주로 투표 방식을 사용

자세한 내용은 12장

학습 모델에는 두 종류의 매개변수가 있음

1. 내부 매개변수
    1-1. 신경망의 경우 에지 가중치로서 이 책은 $\boldsymbol{\theta}$로 표기 (예, 식 (4.1)의 가중치 행렬 $\mathbf{U}^{(l)}$)
    1-2. 학습 알고리즘이 최적화함
2. 하이퍼 매개변수
    2-1. 모델의 외부에서 모델의 동작을 조정함
    2-2. 예, 은닉층의 개수, CNN 마스크 크기와 보폭, 학습률, 모멘텀과 관련된 매개변수 등

Chapter05. 딥러닝 최적화

5.5 하이퍼 매개변수 최적화

하이퍼 매개변수 선택

1. 표준 문헌이 제시하는 기본값을 사용하면 됨
   1-1. 보통 여러 후보 값 또는 범위를 제시
2. 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 $\hat{\mathrm{H}}$

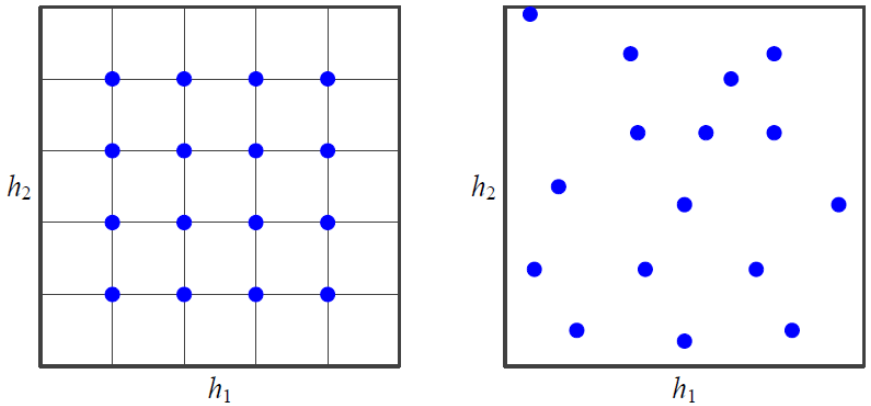
1 $q_{max} = 0$
2 repeat
3 하이퍼 매개변수 조합 $\tilde{\mathrm{H}}$을 생성한다.
4 $\tilde{\mathrm{H}}$으로 설정된 모델을 $\mathbb{X}$로 학습한다.
5 학습된 모델을 $\mathbb{X}_{validate}$로 성능 q를 측정한다.
6 if $(q > q_{max})$ $q_{max} = q$, $\hat{\mathrm{H}} = \tilde{\mathrm{H}}$
7 until(멈춤 조건)

라인 3을 구현하는 방법에 따라 수동 탐색, 격자 탐색, 임의 탐색

Chapter05. 딥러닝 최적화

5.5 하이퍼 매개변수 최적화

격자 탐색과 임의 탐색



(a) 격자 탐색 (b) 임의 탐색

그림 5-29 하이퍼 매개변수 탐색 방법

임의 탐색은 난수로 매개변수 조합을 생성함

로그 규모 간격

1. 어떤 매개변수는 로그 규모를 사용해야 함
2. 예, 학습률 범위가 [0.0001~0.1]일 때
    2-1. 등간격은 0.0001, 0.0002, 0.0003, ..., 0.0998, 0.0999로 총 1000개 값을 조사
    2-2. 로그 규모는 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

1. 매개변수가 m개이고 각각이 q개 구간이라면 q^m개의 점을 조사해야 함

임의 탐색이 우월함

1. [Bergstra2012]의 실험 (32개의 매개변수) → 임의 탐색이 유리함
2. [그림 5-30]은 임의 탐색이 유리한 이유를 설명

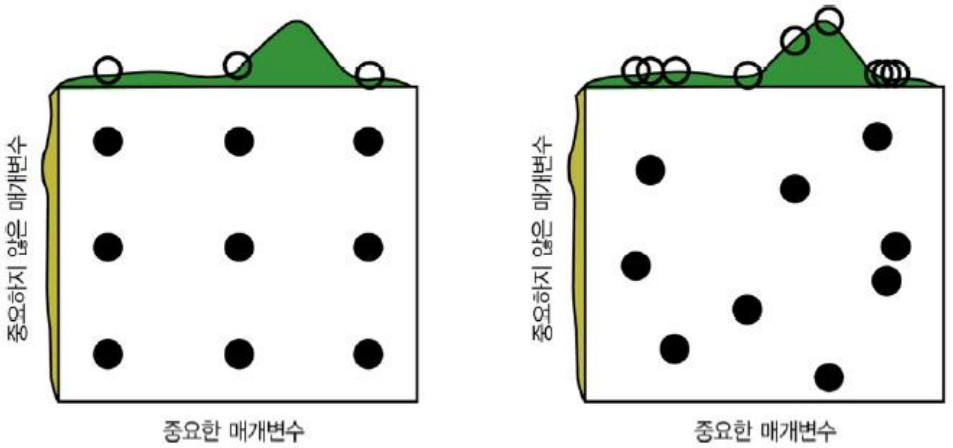


그림 5-30 격자 탐색과 임의 탐색의 성능 비교

Chapter05. 딥러닝 최적화

5.6 2차 미분을 이용한 방법

5.6.1 뉴턴 방법
5.6.2 켤레 그레이디언트 방법
5.6.3 유사 뉴턴 방법

예제 5-2 **경사 하강법의 동작**

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

$$J(\mathbf{w}) = J(w_1, w_2) = (w_1 - 2)^2 + 4(w_2 - 1)^2$$

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

경사 하강법 다시 보기

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 방법

5.6.1 뉴턴 방법
5.6.2 켤레 그레이디언트 방법
5.6.3 유사 뉴턴 방법

경사 하강법을 더 빠르게 할 수 있나

1. [그림 5-31]에서 파란 경로는 경사 하강법이 해를 찾아가는 과정
2. 1차 미분 정보로는 빨간 경로를 알 수 없음
    2-1. 1차 미분은 현재 위치에서 지역적인 기울기 정보만 알려주기 때문

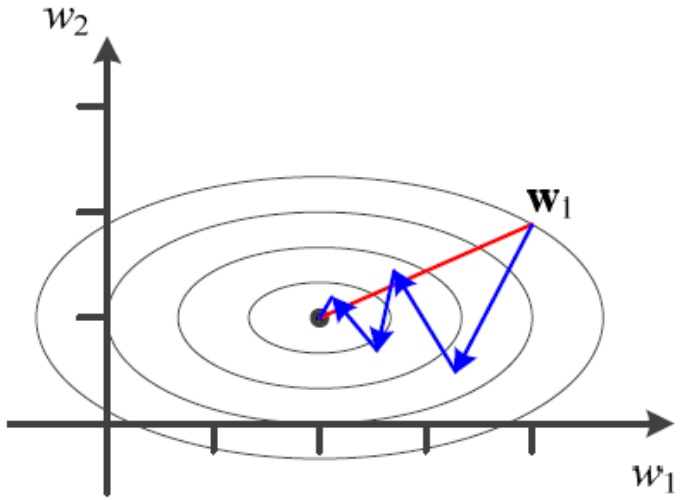


그림 5-31 1차 미분을 사용하는 경사 하강법을 더 빠르게 할 수 있는가

뉴턴 방법은 2차 미분 정보를 활용하여 빨간 경로를 알아냄

5.6 2차 미분을 이용한 방법

5.6.1 뉴턴 방법

테일러 급수를 적용하면, https://sasamath.com/blog/articles/calculus-taylor-series-and-maclaurin-series/

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

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

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

δ로 미분하면,

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

[그림 5-32]처럼 $w+\delta$를 최소점이라 가정하면,

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

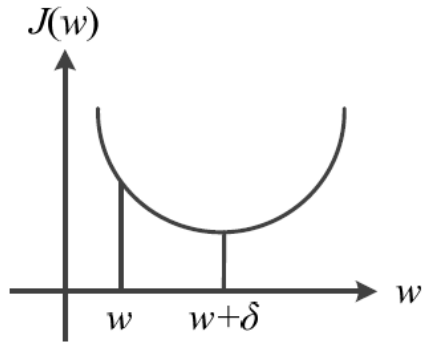


그림 5-32 변수가 하나인 상황의 뉴턴 방법

5.6 2차 미분을 이용한 방법

5.6.1 뉴턴 방법

식을 조금 정리하면,

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

변수가 여러 개인 경우로 확장하면,

$$\boldsymbol{\delta} = -\mathbf{H}^{-1}\nabla J \tag{5.39}$$

5.6 2차 미분을 이용한 방법

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 2차 미분을 이용한 방법

5.6.1 뉴턴 방법

뉴턴 방법의 적용

1. [예제 5.3]은 2차 함수에 뉴턴 방법을 적용했으므로, 3차 항 이상을 무시한 식 (5.36)을 사용했음에도 최적 경우 제시
2. 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
```

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

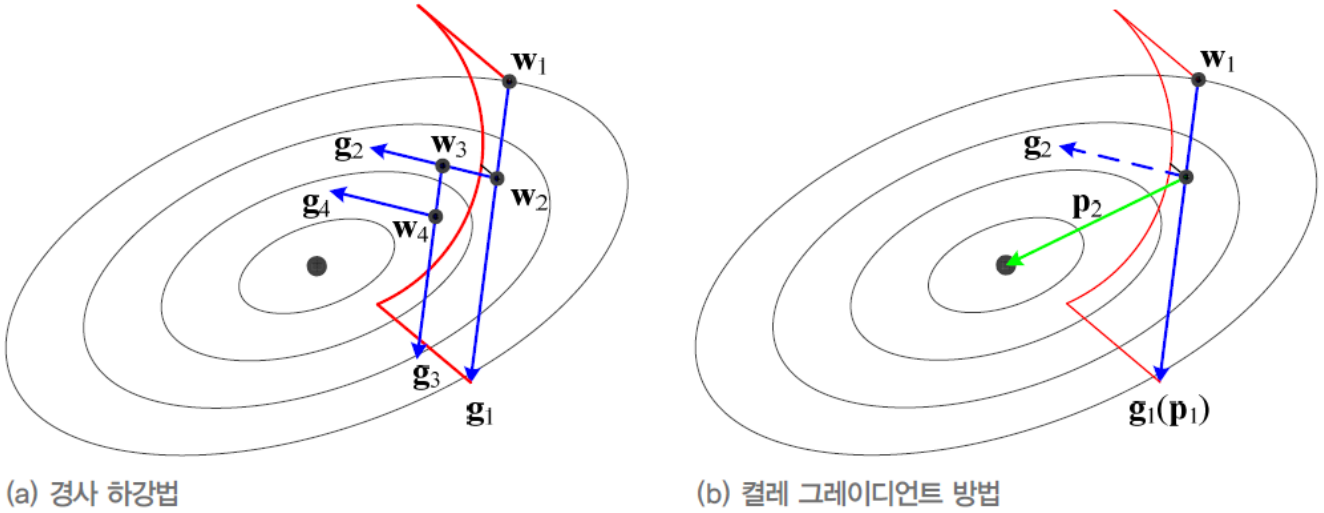
Chapter05. 딥러닝 최적화

5.6 2차 미분을 이용한 방법

5.6.2 켤레 그레이디언트 방법

직선 탐색line search

1. [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

2. 켤레 그레이디언트 방법 1. [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
2. [그림 5-33(b)]의 켤레 그레이디언트는 직전 정보를 사용하여 해에 빨리 접근

5.6 2차 미분을 이용한 방법

5.6.2 켤레 그레이디언트 방법

켤레 그레이디언트

1. 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

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

2. 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

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

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

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

Chapter05. 딥러닝 최적화

5.6 2차 미분을 이용한 방법

5.6.3 유사 뉴턴 방법

1. 유사 뉴턴 방법의 기본 아이디어

1-1. 헤시언 $\mathbf{H}$의 역행렬을 근사하는 행렬 $\mathbf{M}$을 사용

1-2. 처음에 단위 행렬 $\mathbf{I}$로 시작하여 그레이디언트 정보를 이용하여 점점 개선

1-3. LFGS가 많이 사용됨

1-4. 기계 학습에서는 $\mathbf{M}$을 저장하는 메모리를 적게 쓰는 L-BFGS를 주로 사용함

2. 기계 학습에서 2차 미분 정보의 활용

2-1. 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."

기계 학습
Machine Learning

순천향대학교 AI·빅데이터학과
20211464 민현식
minun001@gmail.com