AI 라이브러리 활용

8장 오차역전파

9장 신경망에서 딥러닝으로

이 찬 우

학 습 내 용

1 | 오차 역전파의 개념

2 | 코딩으로 확인하는 오차 역전파

3 | 기울기 소실 문제와 활성화 함수

4 | 속도와 정확도 문제를 해결하는 고급 경사 하강법

8. 오차 역전파

- 신경망 내부의 가중치는 오차 역전파 방법을 사용해 수정함
- 오차 역전파는 경사 하강법의 확장 개념임

1 | 오차 역전파의 개념

- 가중치를 구하는 방법은 경사 하강법을 그대로 이용하면 됨
- 임의의 가중치를 선언하고 결괏값을 이용해 오차를 구한 뒤

  이 오차가 최소인 지점으로 계속해서 조금씩 이동시킴
- 이 오차가 최소가 되는 점(미분했을 때 기울기가 0이 되는 지점)을 찾으면

  그것이 바로 우리가 알고자 하는 답임

1 | 오차 역전파의 개념

- 단일 퍼셉트론에서 결괏값을 얻으면 오차를 구해 이를 토대로 앞 단계에서 정한 가중치를 조정함

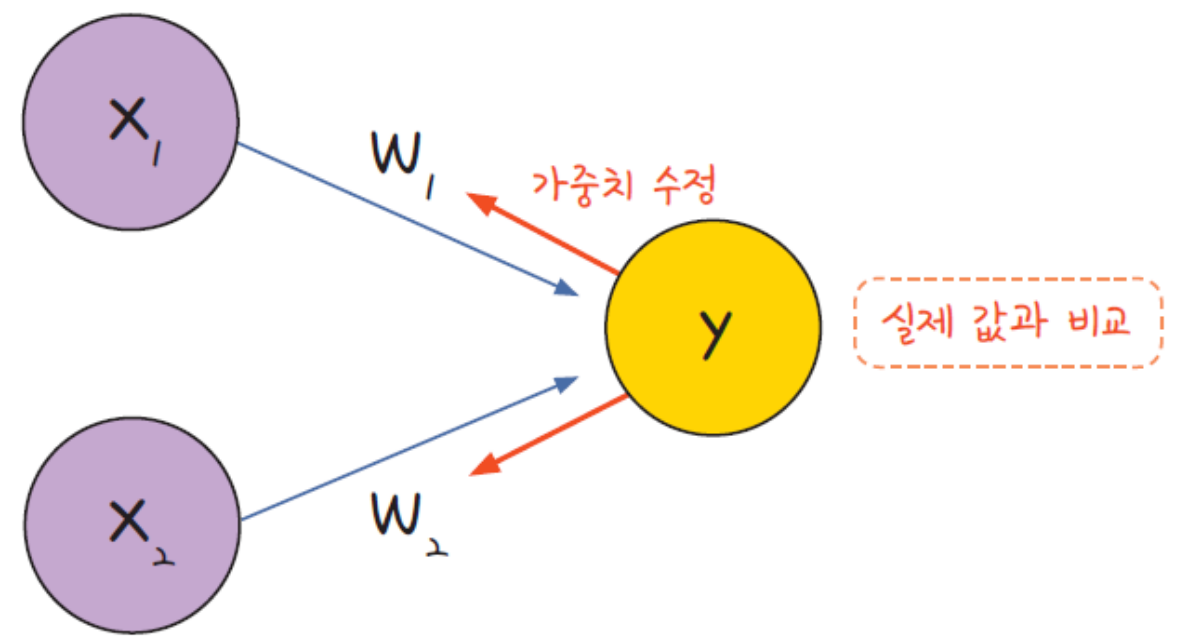

그림 8-1 단일 퍼셉트론에서의 오차 수정

1 | 오차 역전파의 개념

- 다층 퍼셉트론 역시 결괏값의 오차를 구해 이를 토대로 하나 앞선 가중치를 차례로 거슬러 올라가며 조정함

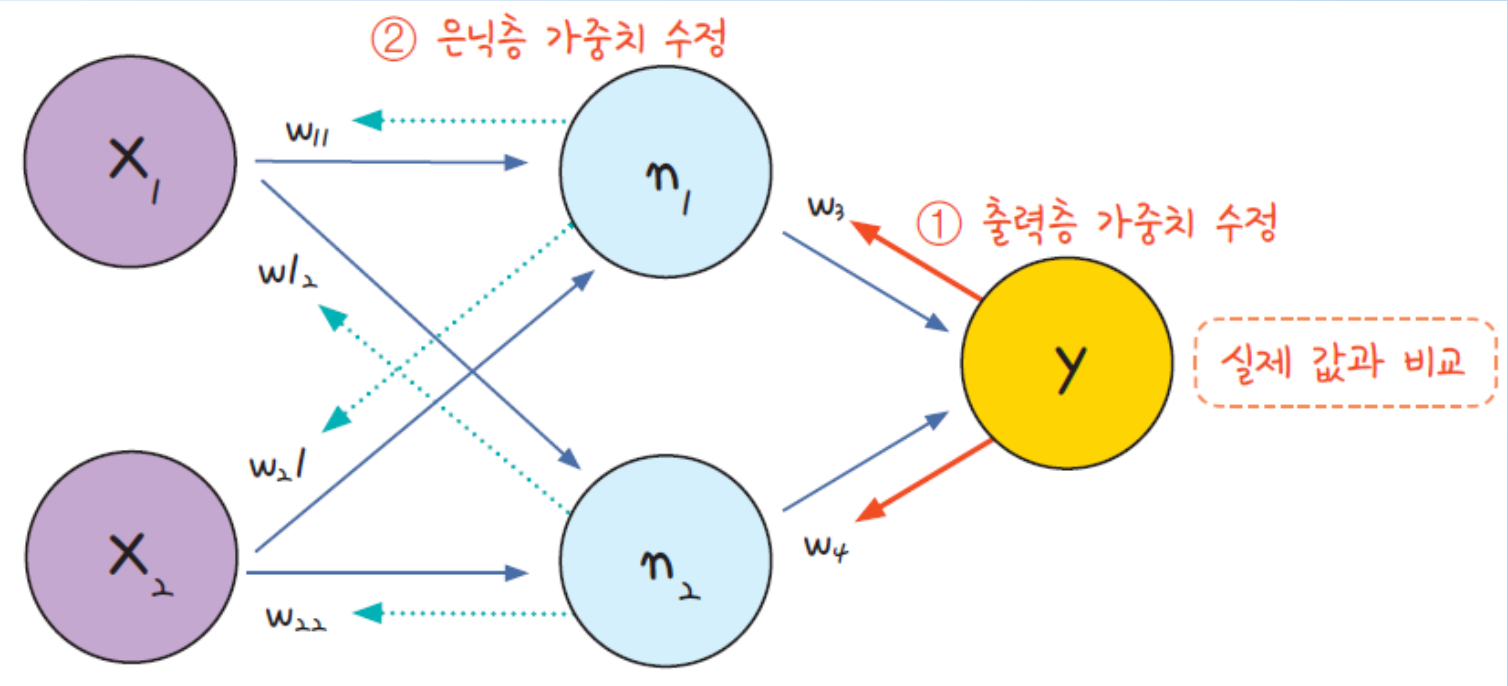

그림 8-2 다층 퍼셉트론에서의 오차 수정

1 | 오차 역전파의 개념

- 오차 역전파(back propagation) :

  다층 퍼셉트론에서의 최적화 과정

- 오차 역전파 구동 방식은 다음과 같이 정리할 수 있음

  1 | 임의의 초기 가중치(w)를 준 뒤 결과(y_{out})를 계산함

  2 | 계산 결과와 우리가 원하는 값 사이의 오차를 구함

  3 | 경사 하강법을 이용해 바로 앞 가중치를 오차가 작아지는 방향으로 업데이트함

  4 | 위 과정을 더이상 오차가 줄어들지 않을 때까지 반복함

1 | 오차 역전파의 개념

- 여기서 '오차가 작아지는 방향으로 업데이트한다'는 의미는

  미분 값이 0에 가까워지는 방향으로 나아간다는 말임

- 즉, '기울기가 0이 되는 방향'으로 나아가야 하는데, 이 말은 **가중치에서 기울기를 뺐을 때 가중치의 변화가 전혀 없는 상태를 말함**

- 오차 역전파를 다른 방식으로 표현하면 가중치에서 기울기를 빼도 값의 변화가 없을 때까지 계속해서 가중치 수정 작업을 반복하는 것

새 가중치는 현 가중치에서 '가중치에 대한 기울기'를 뺀 값!

$$W(t+1) = Wt - \frac{\partial 오차}{\partial W}$$

2 | 코딩으로 확인하는 오차 역전파

- 입력된 실제 값과 다층 퍼셉트론의 계산 결과를 비교하여 가중치를 역전파 방식으로 수정해 가는 코딩은 그림 8-3과 같은 순서로 구현함

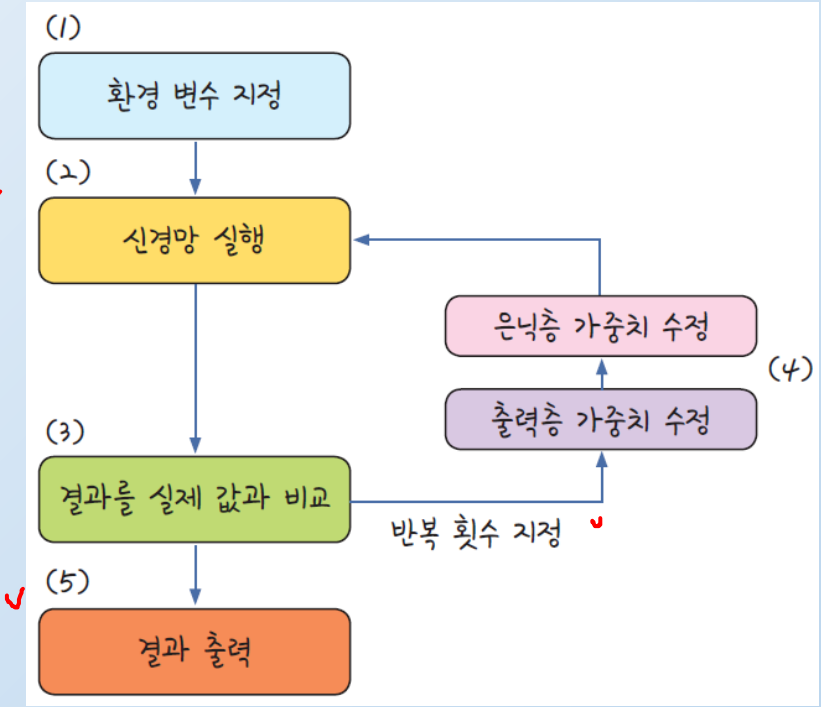

그림 8-3 신경망의 구현 과정

오차 역전파의 개념

- 각각을 조금 더 자세히 설명하면 다음과 같음

  1 | **환경 변수 지정:** 환경 변수에는 입력 값과 타깃 결괏값이 포함된 데이터셋, 학습률 등이 포함되고 활성화 함수와 가중치 등도 선언되어야 함

  2 | **신경망 실행:** 초깃값을 입력하여 활성화 함수와 가중치를 거쳐 결괏값이 나오게 함

  3 | **결과를 실제 값과 비교:** 오차를 측정함

  4 | **역전파 실행:** 출력층과 은닉층의 가중치를 수정함

  5 | **결과 출력**

9장 신경망에서 딥러닝으로

3 | 기울기 소실 문제와 활성화 함수

4 | 속도와 정확도 문제를 해결하는 고급 경사 하강법

9. 신경망에서 딥러닝으로

- 다층 퍼셉트론이 오차 역전파를 만나 신경망이 되었고, 신경망은 XOR문제를 가볍게 해결함
- 이제 신경망을 차곡차곡 쌓아 올리면 마치 사람처럼 생각하고 판단하는 인공지능이 금방이라도 완성될 것처럼 보임

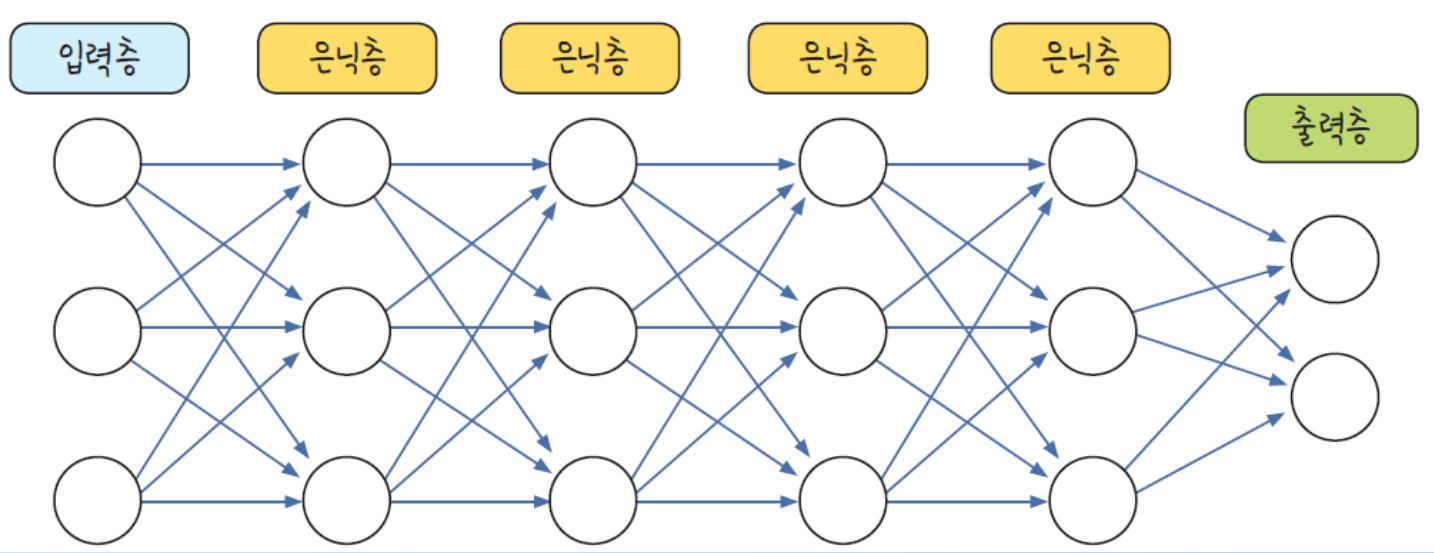

그림 9-1 다층 확장

3 | 기울기 소실 문제와 활성화 함수

- 오차 역전파 :

  출력층으로 부터 하나씩 앞으로 되돌아가며 각 층의 가중치를 수정하는 방법

- 가중치를 수정하려면 미분 값, 즉 기울기가 필요하다고 배움

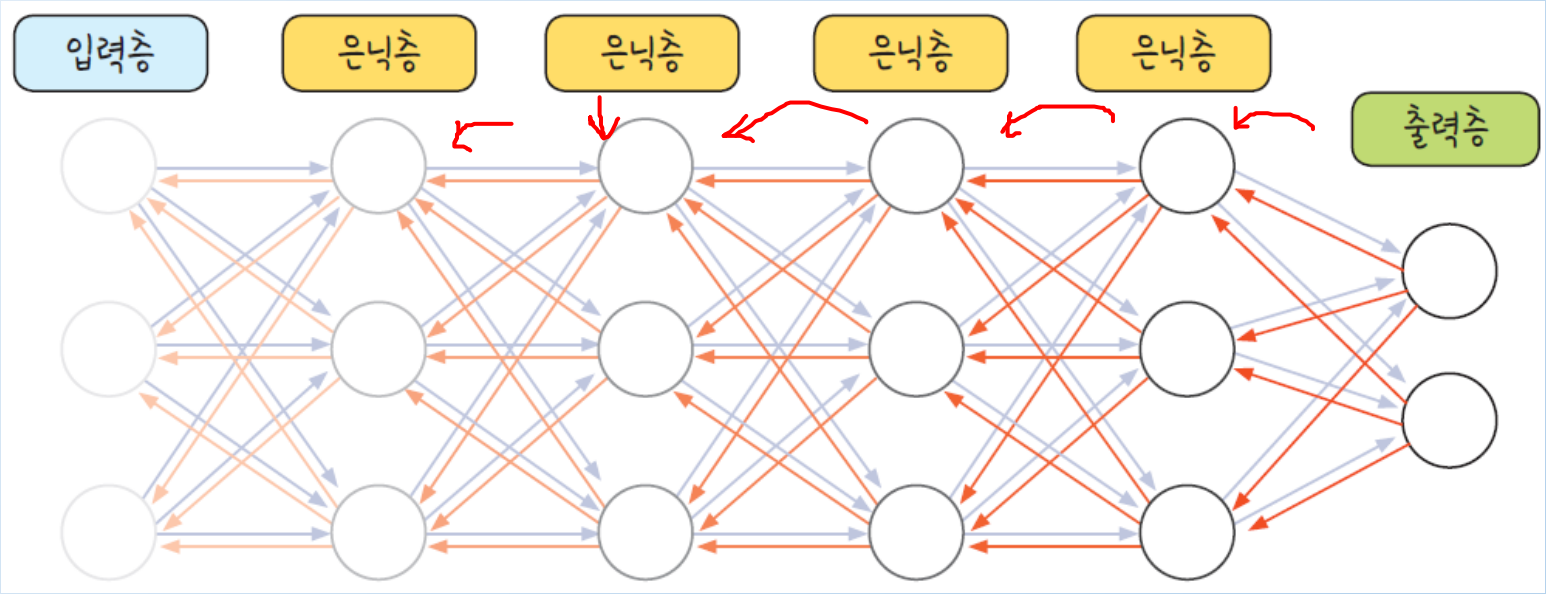

그림 9-2 기울기 소실 문제 발생!

3 | 기울기 소실 문제와 활성화 함수

- 기울기 소실(vanishing gradient) 문제가 발생하기 시작한 것은 활성화 함수로 사용된 시그모이드 함수의 특성 때문임
- 여러 층을 거칠수록 기울기가 사라져 가중치를 수정하기가 어려워지는 것임

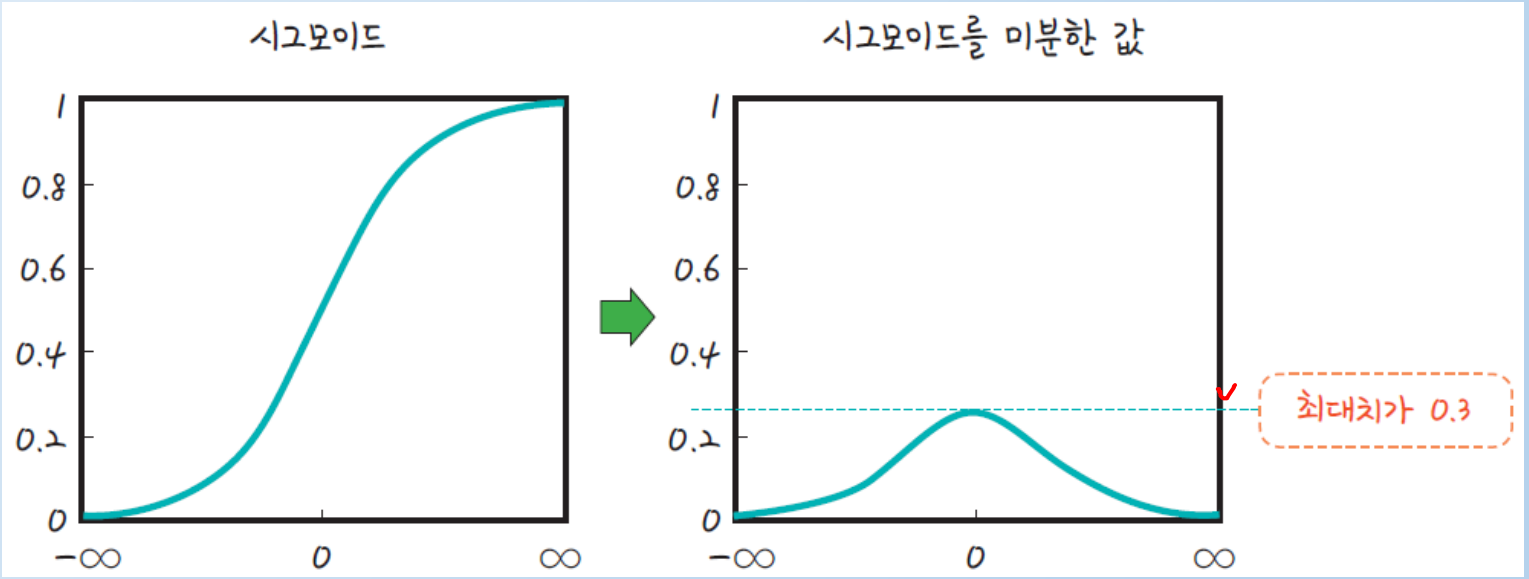

그림 9-3 시그모이드의 미분

3 | 기울기 소실 문제와 활성화 함수

- 이를 해결하고자 활성화 함수를 시그모이드가 아닌 여러 함수로 대체하기 시작함

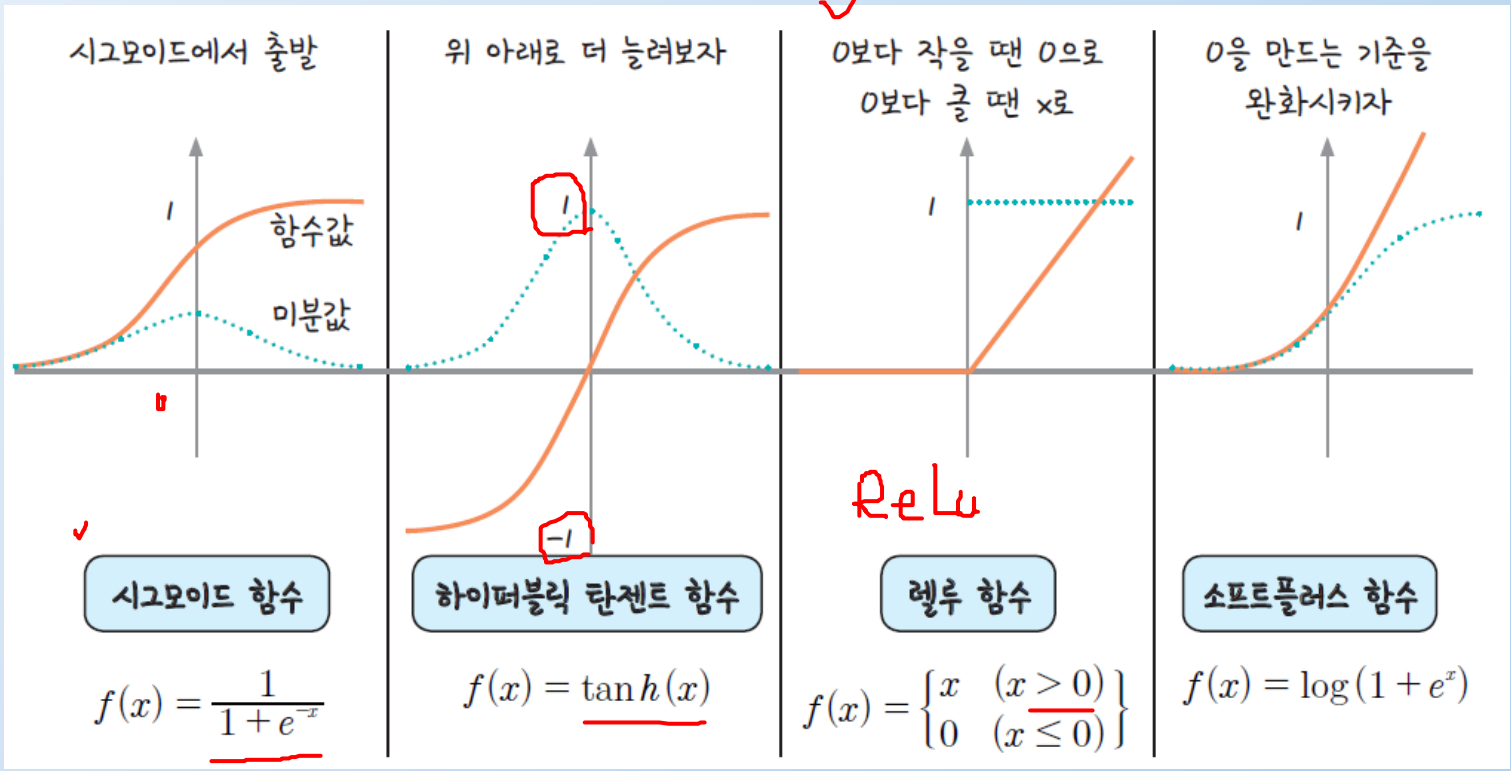

그림 9-4 여러 활성화 함수의 도입

3 | 기울기 소실 문제와 활성화 함수

- 하이퍼볼릭 탄젠트(tanh) 함수
  - 미분한 값의 범위가 함께 확장되는 효과를 가져옴
  - 다양한 여전히 1보다 작은 값이 존재하므로 기울기 소실 문제는 사라지지 않음
- 렐루(ReLU) 함수
  - 시그모이드 함수의 대안으로 떠오르며 현재 가장 많이 사용되는 활성화 함수임
  - 여러 은닉층을 거치며 곱해지더라도 맨 처음 층까지 사라지지 않고 남아있을 수 있음
  - 이 간단한 방법이 여러 층을 쌓을 수 있게 했고, 이로써 딥러닝의 발전에 속도가 붙게 됨

3 | 기울기 소실 문제와 활성화 함수

- 소프트플러스 (softplus) 함수
  - 이후 렐루 함수의 값이 0 이 되는 순간을 완화

4 | 속도와 정확도 문제를 해결하는 고급 경사 하강법

- 가중치를 업데이트하는 방법으로 우리는 경사 하강법을 배웠음
- 경사 하강법은 정확하게 가중치를 찾아가지만, 한 번 업데이트할 때마다 전체 데이터를 미분해야 하므로 계산량이 매우 많다는 단점이 있음
- 이러한 점을 보완한 고급 경사 하강법이 등장하면서 딥러닝의 발전 속도는 더 빨라짐

4 | 속도와 정확도 문제를 해결하는 고급 경사 하강법

확률적 경사 하강법

- 경사 하강법은 불필요하게 많은 계산량으로 속도를 느리게 할 뿐 아니라,
  최적 해를 찾기 전에 최적화 과정이 멈출 수도 있음
- 확률적 경사 하강법의 이러한 단점을 보완한 방법
- 전체 데이터를 사용하는 것이 아니라, 랜덤하게 추출한 일부 데이터를 사용함
- 일부 데이터를 사용하므로 더 빨리 그리고 자주 업데이트를 하는 것이 가능해짐

4 | 속도와 정확도 문제를 해결하는 고급 경사 하강법

- 랜덤한 일부 데이터를 사용하는 만큼 확률적 경사 하강법은 중간 결과의 진폭이 크고 불안정해 보일 수도 있음
- 속도가 확연히 빠르면서도 최적 해에 근사한 값을 찾아낸다는 장점 덕분에 경사 하강법의 대안으로 사용되고 있음

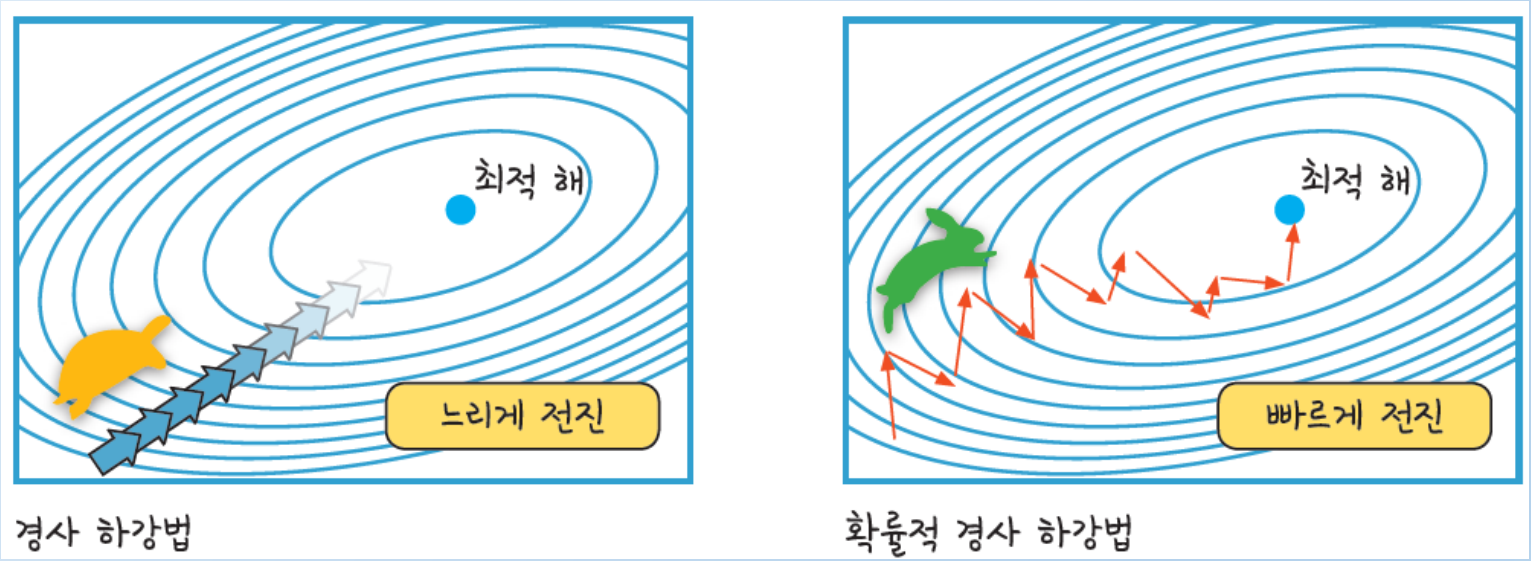

그림 9-5 경사 하강법과 확률적 경사 하강법의 비교

4 | 속도와 정확도 문제를 해결하는 고급 경사 하강법

모멘텀

- 모멘텀(momentum)이란 단어는 '관성, 탄력, 가속도'라는 뜻
- 모멘텀 SGD란 말 그대로 경사 하강법에 탄력을 더해 주는 것
- 다시 말해서, 경사 하강법과 마찬가지로 매번 기울기를 구하지만,
  이를 통해 오차를 수정하기 전 바로 앞 수정 값과 방향(+, -)을 참고하여
  같은 방향으로 일정한 비율만 수정되게 하는 방법
- 수정 방향이 양수(+) 방향으로 한 번, 음수(-) 방향으로 한 번 지그재그로 일어나는 현상이
  줄어들고, 이전 이동 값을 고려하여 일정 비율만큼만 효과를 낼 수 있음

4 | 속도와 정확도 문제를 해결하는 고급 경사 하강법

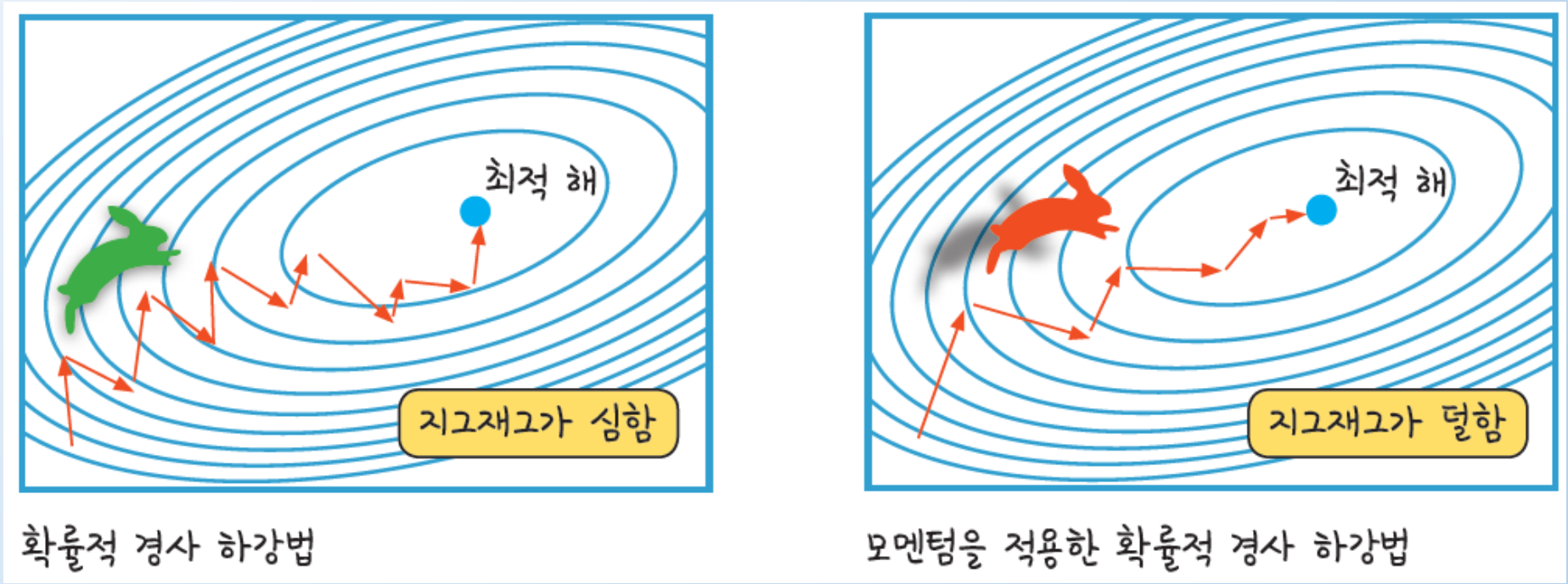

그림 9-6 모멘텀을 적용했을 때

4 | 속도와 정확도 문제를 해결하는 고급 경사 하강법

고급 경사 하강법	개요	효과	케라스 사용법
확률적 경사 하강법 (SGD)	랜덤하게 추출한 일부 데이터를 사용해 더 빨리, 자주 업데이트를 하게 하는 것	속도 개선	keras.optimizers.SGD(lr = 0.1) 케라스 최적화 함수를 이용합니다.
모멘텀 (Momentum)	관성의 방향을 고려해 진동과 폭을 줄이는 효과	정확도 개선	keras.optimizers.SGD(lr = 0.1, momentum = 0.9) 모멘텀 계수를 추가합니다.
네스테로프 모멘텀 (NAG)	모멘텀이 이동시킬 방향으로 미리 이동해서 그레이디언트를 계산. 불필요한 이동을 줄이는 효과	정확도 개선	keras.optimizers.SGD(lr = 0.1, momentum = 0.9, nesterov = True) 네스테로프 옵션을 추가합니다.
아다그라드 (Adagrad)	변수의 업데이트가 잦으면 학습률을 적게 하여 이동 보폭을 조절하는 방법	보폭 크기 개선	keras.optimizers.Adagrad(lr = 0.01, epsilon = 1e - 6) 아다그라드 함수를 사용합니다. ※ 참고: 여기서 epsilon, rho, decay 같은 파라미터는 바꾸지 않고 그대로 사용하기를 권장하고 있습니다. 따라서 lr, 즉 learning rate(학습률) 값만 적절히 조절하면 됩니다.

4 | 속도와 정확도 문제를 해결하는 고급 경사 하강법

고급 경사 하강법	개요	효과	케라스 사용법
알엠에스프롭 (RMSProp)	아다그라드의 보폭 민감도를 보완한 방법	보폭 크기 개선	keras.optimizers.RMSprop(lr = 0.001, rho = 0.9, epsilon = 1e - 08, decay = 0.0) 알엠에스프롭 함수를 사용합니다.
아담(Adam)	모멘텀과 알엠에스프롭 방법을 합친 방법	정확도와 보폭 크기 개선	keras.optimizers.Adam(lr = 0.001, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e - 08, decay = 0.0) 아담 함수를 사용합니다.

표 9-1 딥러닝 구동에 사용되는 고급 경사 하강법 개요 및 활용법

정 리 학 습

1 | 오차 역전파의 개념

2 | 코딩으로 확인하는 오차 역전파

3 | 기울기 소실 문제와 활성화 함수

4 | 속도와 정확도 문제를 해결하는 고급 경사 하강법

다음 수업

파이썬의 자료구조II 부분을 학습