STEP28. 함수 최적화

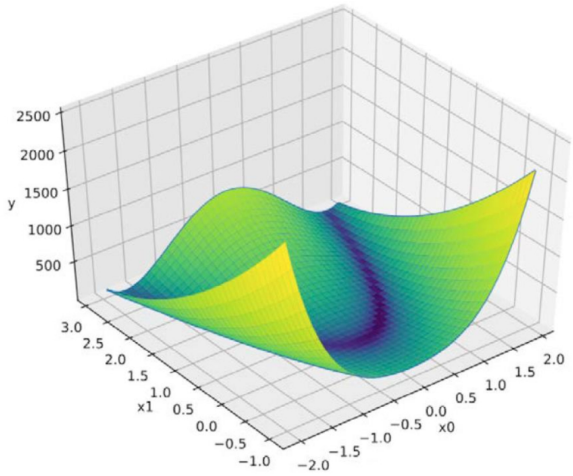
미분의 중요한 용도 == 함수 최적화
목표 : 구체적인 함수를 대상으로 최적화 계산

최적화

- 최적화 == 어떤 함수가 주어졌을 때 그 최솟값(또는 최댓값)을 반환하는 입력(함수의 인수)을 찾는 일
- 신경망 학습 목표도 손실 함수의 출력을 최소화하는 매개변수를 찾는 것이니 최적화 문제에 속함

로젠브록 함수 (Rosenbrock function)

$y = 100(x_1 - x_0^2)^2 + (1 - x_0)^2$



a, b 가 정수일 때

$f(x_0, x_1) = b(x_1 - x_0^2)^2 + (a - x_0)^2$

- a = 1, b= 100 으로 설정하여 벤치마크하는 것이 일반적임
- 형태는 포물선 모양으로 길게 뻗은 골짜기가 보임

로젠브록 함수 최적화

- 출력이 최소가 되는 $x0$ 와 $x1$ 을 찾는 것임
- 최솟값이 되는 지점은 $(x0, x1)$ = (1, 1)

DeZero 이용하여 최솟값 찾기

- 최솟값 지점을 실제로 찾아낼 수 있는지 확인

미분 계산하기

1. 로젠브록 함수의 미분

```
import numpy as np
from dezero import Variable

def rosenbrock(x0, x1):
    y = 100 * (x1 - x0 ** 2) ** 2 + (1 - x0) ** 2
    return y
```

- $(x0, x1)$ = (0.0, 2.0) 에서의 미분 ($x'0$와 $x'1$) 계산
- 수치 데이터를 Variable로 감싸서 건네주고 그 다음은 수식을 따라 코딩
- $x0$ 과 $x1$의 미분은 각각 -2.0과 400.0이 나옴
- (-2.0, 400.0) 벡터를 기울기 혹은 기울기 벡터라고 함
- 기울기는 각 지점에서 함수의 출력을 가장 크게 하는 방향을 가르킴
- $(x0, x1)$ = (0.0, 2.0) 지점에서 y값을 가장 크게 늘려주는 방향이 (-2.0, 400.0) 이라는 의미
- 반대로 기울기에 마이너스를 곱한 (2.0, -400.0) 방향은 y 값을 가장 작게 줄여주는 방향

경사하강법 구현

경사하강법(gradient descent)

- 복잡한 형상의 함수라면 기울기가 가리키는 방향에 반드시 최솟값이 존재하지는 않음
- 국소적으로 보면 기울기는 함수의 출력을 가장 크게 하는 방향을 나타냄
- 기울기 방향으로 일정 거리만큼 이동하여 다시 기울기를 구하는 작업을 반복하면 점차 최솟값(혹은 최댓값)에 접근하리라 기대할 수 있음
- 좋은 초깃값은 경사하강법을 목적지까지 효율적으로 도달하게 함

로젠브록 함수의 최솟값 찾기

```
# 경사하강법
x0 = Variable(np.array(0.0))
x1 = Variable(np.array(2.0))

lr = 0.001 # 학습률
iters = 1000 # 반복횟수

for i in range(iters):
    print(x0, x1)

    y = rosenbrock(x0, x1)

    x0.cleargrad()
    x1.cleargrad()
    y.backward()

    # 경사하강법을 이용한 변수 업데이트
    x0.data -= lr * x0.grad
    x1.data -= lr * x1.grad
```

- iters == iterations (반복)
- lr == learning rate (학습률)
- 기울기 방향에 마이너스를 곱한 방향으로 이동함
- cleargrad 메서드
  - x0.grad, x1.grad 는 미분값이 누적되기 때문에 새롭게 미분할 때는 누적된 값을 초기화 해야 함
- 코드를 실행해보면 (x0, x1) 값이 갱신되는 과정을 볼수 있음

로젠브록 함수의 최솟값에 접근 경로

그림 28-2 경신 경로(빨간 점의 궤적이 경신 과정, 별이 최솟값의 위치)

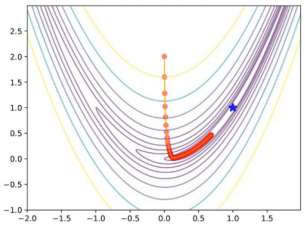
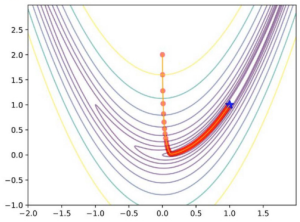


그림 28-3 iters = 10000일 때의 결과



- 반복횟수 iters = 1000 일때 최솟값에 접근하는 도중 멈춤
- 반복횟수 iters = 10000 으로 늘려 다시 실행했을 때, 최솟값이 더욱 가까워짐
- 반복횟수 iters = 50000 으로 설정시에 실제 (1.0, 1.0) 위치에 도달함
- 경사하강법은 로젠블록 함수 같이 골짜기가 길게 뻗은 함수에서 잘 대응 못함