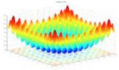
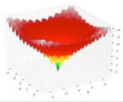
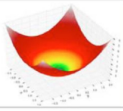
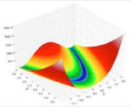
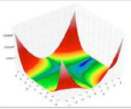
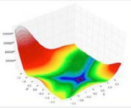
STEP24. 복잡한 함수의 미분

Test functions for optimization

그림 24-1 최적화 문제에 사용되는 벤치마크 함수 목록(위키백과[13]에서 발췌)

Name	Plot	Formula
Rastrigin function		$f(\mathbf{x}) = An + \sum_{i=1}^{n}\left[x_i^2 - A\cos(2\pi x_i)\right]$ where: $A = 10$
Ackley function		$f(x,y) = -20\exp\left[-0.2\sqrt{0.5\left(x^2+y^2\right)}\right] - \exp\left[0.5\left(\cos 2\pi x + \cos 2\pi y\right)\right] + e + 20$
Sphere function		$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$
Rosenbrock function		$f(\boldsymbol{x}) = \sum_{i=1}^{n-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + \left(1 - x_i\right)^2\right]$
Beale function		$f(x,y) = \left(1.5 - x + xy\right)^2 + \left(2.25 - x + xy^2\right)^2 + \left(2.625 - x + xy^3\right)^2$
Goldstein–Price function		$f(x,y) = \left[1 + (x+y+1)^2\left(19 - 14x + 3x^2 - 14y + 6xy + 3y^2\right)\right]\left[30 + (2x-3y)^2\left(18 - 32x + 12x^2 + 48y - 36xy + 27y^2\right)\right]$

DeZero의 연산자 지원

- 대표적인 연산자들(+, *, -, /, **)을 지원함
- 복잡한 수식을 파이썬 프로그래밍 하듯 코딩 지원

최적화 문제의 테스트 함수 (Test functions for optimization)

- 다양한 최적화 기법을 평가하는 데 사용되는 함수
- 벤치마크용 함수
- 세 함수 미분 수행
  - Sphere 함수, mathyas 함수, Goldstein-Price 함수

Dezero 기능 정리

- 아무리 복잡하게 연결된 계산 그래프라도 올바르게 역전파 할수 있음
- 연산자 오버로드를 통해 파이썬 프로그래밍처럼 코드 작성 가능
- 일반적인 프로그래밍을 미분 가능하게 만듦

Sphere 함수

1. $z = x^2 + y^2$

```
import numpy as np
from dezero import Variable

# z = x² + y²
def sphere(x, y):
    z = x ** 2 + y ** 2
    return z

x = Variable(np.array(1.0))
y = Variable(np.array(1.0))
z = sphere(x, y)
z.backward()
print(x.grad, y.grad)
```

- Sphere 함수 수식 $z = x^2 + y^2$
- (x, y) = (1.0, 1.0) 인 경우, 미분 수행 결과는 (2.0, 2.0) 이 되어야 함

matyas 함수

2. $z = 0.26(x^2 + y^2) - 0.48xy$

```
def matyas(x, y):
    z = 0.26 * (x ** 2 + y ** 2) - 0.48 * x * y
    return z

x = Variable(np.array(1.0))
y = Variable(np.array(1.0))
z = matyas(x, y)
z.backward()
print(x.grad, y.grad)
```

- matyas 함수 수식 $z = 0.26(x^2 + y^2) - 0.48xy$
- (x, y) = (1.0, 1.0) 인 경우, 미분 수행 결과는 (0.04, 0.04) 이 되어야 함

Goldstein-Price 함수

3. $f(x, y) = [1 + (x + y + 1)^2 (19 - 14x + 3x^2 -14y + 6xy + 3y^2)]$
   $[30 + (2x - 3y)^2 (18 - 32x + 12x^2 + 48y -36xy + 27y^2)]$

```
def goldstein(x, y):
    z = (1 + (x + y + 1)**2 * (19 - 14*x + 3*x**2 - 14*y + 6*x*y + 3*y**2)) * \
        (30 + (2*x - 3*y)**2 * (18 - 32*x + 12*x**2 + 48*y - 36*x*y + 27*y**2))

    return z

x = Variable(np.array(1.0))
y = Variable(np.array(1.0))
z = goldstein(x, y)
z.backward()
print(x.grad, y.grad)
```

- (x, y) = (1.0, 1.0) 인 경우, 미분 수행 결과는 (-5376.0, 8064.0) 이 되어야 함

get_dot_graph <- 그래프 내용 만드는 코드
plot_dot_graph <- 그래프 만들어주는 코드

신경망에서는 두 값의 내적값들이 다음함수로 출력으로 내보내지는

→ cosθ가 180일 때 (-1) 두 백터의 내적값 (최솟값)

→ 경사하강법 (기울기가 가장 높은 곳에 내려가야 골짜기를 찾음) ← 최적값

- 50000번 해야 겨우 도달함

→ 뉴턴방법