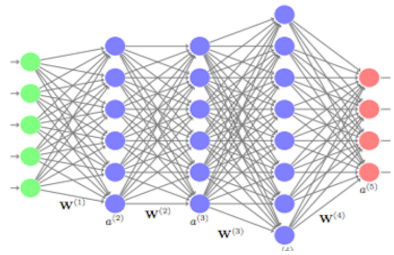
به نام خدا

علی فتحی ۹۴۱۰۹۲۰۵

تکلیف سری ۲ شبکه‌های عمیق – دکتر فاطمی‌زاده



بخش اول: پیاده سازی توابع موردنظر برای بهینه سازی

توابع موجود در صورت سوال به صورت زیر در کد پیاده شده اند:

```python
# Rastrigin Function
def rastrigin(x1, x2):
    f = 20 + np.power(x1, 2) + np.power(x2, 2) - 10 * np.cos(2 * np.pi * x1) - 10 * np.cos(2 * np.pi * x1)
    return f


# Ackley Function
def ackley(x1, x2):
    f = 20 + np.e - 20 * np.exp(-0.2 * np.sqrt(0.5 * (np.power(x1, 2) + np.power(x2, 2)))) \
        - np.exp(0.5 * (np.cos(2 * np.pi * x1) + np.cos(2 * np.pi * x2)))
    return f


# Levi Function
def levi(x1, x2):
    f = np.power(np.sin(3 * np.pi * x1), 2) + np.power(x1 - 1, 2) * (1 + np.power(np.sin(3 * np.pi * x2), 2)) \
        + np.power(x2 - 1, 2) * (1 + np.power(np.sin(2 * np.pi * x2), 2))
    return f


# Bukin Function
def bukin(x1, x2):
    f = 100 * np.sqrt(np.abs(x2 - 0.01 * np.power(x1, 2)))
    return f
```

بخش دوم: محاسبه بردار گرادیان

گرادیان توابع سوال در ذیل محاسبه می‌شود:

1) Rastrigin Function:

$$f(x_1, x_2) = 20 + (x_1^2 + x_2^2) - 10\,[\cos(2\pi x_1) + \cos(2\pi x_2)]$$
$$\Rightarrow \vec{\nabla} f = [2x_1 + 20\pi \sin(2\pi x_1)]\widehat{x_1} + [2x_2 + 20\pi \sin(2\pi x_2)]\widehat{x_2}$$

2) Ackley Function:

$$f(x_1, x_2) = 20 + e - 20e^{-0.2\sqrt{0.5(x_1^2+x_2^2)}} - e^{0.5(\cos(2\pi x_1)+\cos(2\pi x_2))}$$

$$\Rightarrow \vec{\nabla} f = \left[\frac{2x_1}{\sqrt{0.5(x_1^2+x_2^2)}} e^{-0.2\sqrt{0.5(x_1^2+x_2^2)}} + \pi \sin(2\pi x_1)\ e^{0.5(\cos(2\pi x_1)+\cos(2\pi x_2))}\right]\widehat{x_1} + \left[\frac{2x_2}{\sqrt{0.5(x_1^2+x_2^2)}} e^{-0.2\sqrt{0.5(x_1^2+x_2^2)}} + \pi \sin(2\pi x_2)\ e^{0.5(\cos(2\pi x_1)+\cos(2\pi x_2))}\right]\widehat{x_2}$$

3) Levi Function:

$$f(x_1, x_2) = \sin^2(3\pi x_1) + (x_1 - 1)^2(1 + \sin^2(3\pi x_2)) + (x_2 - 1)^2(1 + \sin^2(2\pi x_2))$$

$$\Rightarrow \vec{\nabla} f = [6\pi \sin(3\pi x_1)\cos(3\pi x_1) + 2(x_1 - 1)(1 + \sin^2(3\pi x_2))]\widehat{x_1} + [6\pi(x_1 - 1)^2 \sin(3\pi x_2)\cos(3\pi x_2) + 2(x_2 - 1)(1 + \sin^2(2\pi x_2)) + 4\pi(x_2 - 1)^2 \sin(2\pi x_2)\cos(2\pi x_2)]\widehat{x_2}$$

4) Bukin Function:

$$f(x_1, x_2) = 100\sqrt{|x_2 - 0.01x_1|} + 0.01|x_1 + 10|$$

$$\Rightarrow \vec{\nabla} f = \left[0.01 sign(x_1 + 10) - \frac{sign(x_2 - 0.01x_1)}{2\sqrt{|x_2 - 0.01x_1|}}\right]\widehat{x_1} + [\frac{50 sign(x_2 - 0.01x_1)}{\sqrt{|x_2 - 0.01x_1|}}]\widehat{x_2}$$

5) n-D Rastrigin Function:

$$f(x_1, x_2) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i))$$

$$\Rightarrow \vec{\nabla} f = 2\sum_{i=1}^{n}[x_i + 10\pi \sin(2\pi x_i)]\widehat{x_i}$$

بخش سوم: پیاده‌سازی بردارهای گرادیان

گرادیان توابع سوال به صورت زیر پیاده شده اند:

```python
# ***************************************** Gradient Implementation *****************************************
# Rastrigin Gradient
def rastrigin_grad(x1, x2):
    g1 = 2 * x1 + 20 * np.pi * np.sin(2 * np.pi * x1)
    g2 = 2 * x2 + 20 * np.pi * np.sin(2 * np.pi * x2)
    return g1, g2


# Ackley Gradient
def ackley_grad(x1, x2):
    r = np.sqrt(0.5 * (np.power(x1, 2) + np.power(x2, 2)))
    e_cos = np.exp(0.5 * (np.cos(2 * np.pi * x1) + np.cos(2 * np.pi * x2)))
    g1 = 2 * x1 * np.exp(-0.2 * r) / r + np.pi * np.sin(2 * np.pi * x1) * e_cos
    g2 = 2 * x2 * np.exp(-0.2 * r) / r + np.pi * np.sin(2 * np.pi * x2) * e_cos
    return g1, g2


# Levi Gradient
def levi_grad(x1, x2):
    g1 = 6 * np.pi * np.sin(3 * np.pi * x1) * np.cos(3 * np.pi * x1) \
         + 2 * (x1 - 1) * (1 + np.power(np.sin(3 * np.pi * x2), 2))
    g2 = 6 * np.pi * np.power((x1 - 1), 2) * np.sin(3 * np.pi * x1) * np.cos(3 * np.pi * x1) \
        + 2 * (x2 - 1) * (1 + np.power(np.sin(2 * np.pi * x2), 2)) \
        + 4 * np.pi * np.power((x2 - 1), 2) * np.sin(2 * np.pi * x1) * np.cos(2 * np.pi * x1)
    return g1, g2


# Bukin Gradient
def bukin_grad(x1, x2):
    sign_rad = np.sign(x2 - 0.01 * np.power(x1, 2)) / np.sqrt(np.abs(x2 - 0.01 * np.power(x1, 2)))
    g1 = 0.01 * np.sign(x1 + 10) - 0.5 * x1 * sign_rad
    g2 = 50 * sign_rad
    return g1, g2


# n-D Rastrigin Gradient
def n_d_rastrigin_grad(x):
    g = 2 * x + 20 * np.pi * np.sin(2 * np.pi * x)
    return g
# ************************************************************************************************************
```

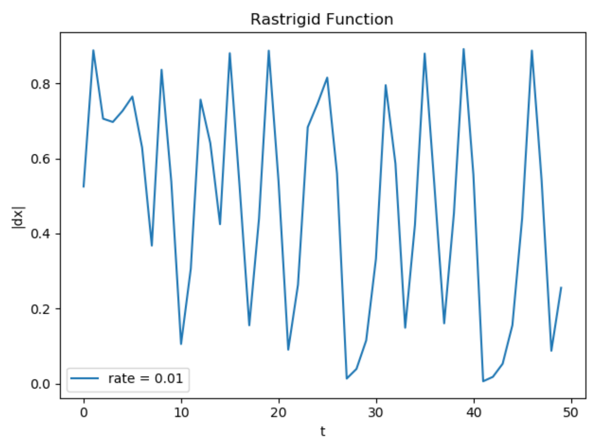
بخش چهارم: پیاده‌سازی الگوریتم‌های بهینه‌سازی

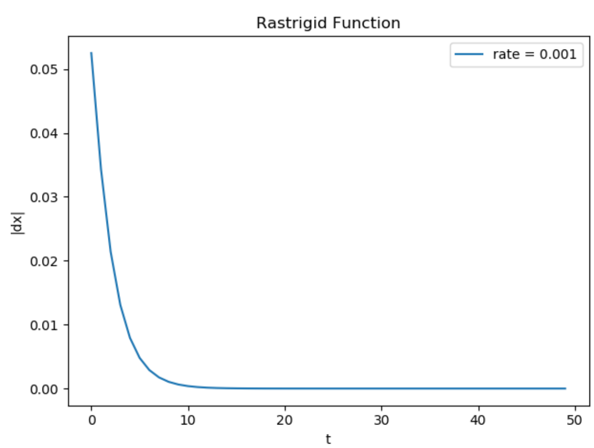
الگوریتم‌های بهینه‌سازی هرکدام بصورت یک class پیاده شده‌اند؛ به صورتی که هر کدام با تابع گرادیان و نقطه اولیه‌ای مشخص می‌شود و سپس به وسیله متد update، به سمت نقطه با گرادیان کمتر حرکت می‌کند. برای مثال، پیاده‌سازی یکی از این کلاس‌ها مربوط به روش Gradient Descend ساده در زیر آمده است:

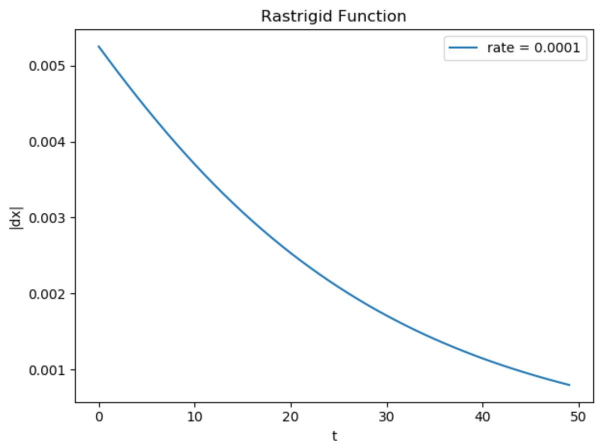
```python
# Simple Gradient Descend
class SimpleGradientDescend:
    def __init__(self, grad_func, init_point):
        self.eta = 0.001
        self.grad_func = grad_func
        self.position = init_point
        self.movements = []
        self.all_positions = []
        self.all_positions.append(init_point)

    def update(self):
        dx, dy = self.grad_func(self.position.x, self.position.y)
        self.position.move(-self.eta * dx, -self.eta * dy)
        self.movements.append(np.sqrt(np.power(self.eta * dx, 2) + np.power(self.eta * dy, 2)))
        self.all_positions.append(Point(self.position.x, self.position.y))
```

بخش پنجم: نمودارها و مقایسه

ابتدا به کمک روش Gradient Descend ساده، پارامتر Learning Rate یا η را مشخص می‌کنیم:

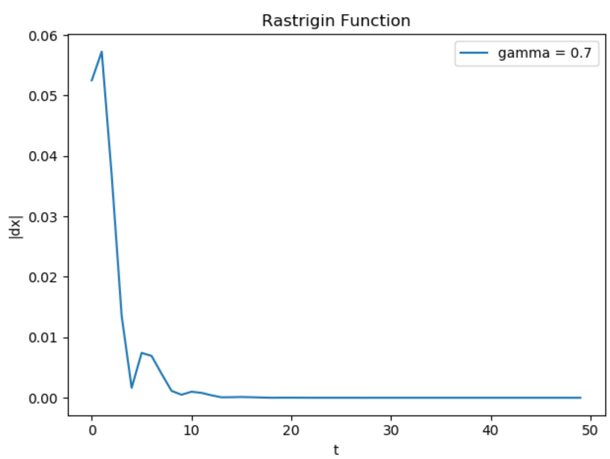


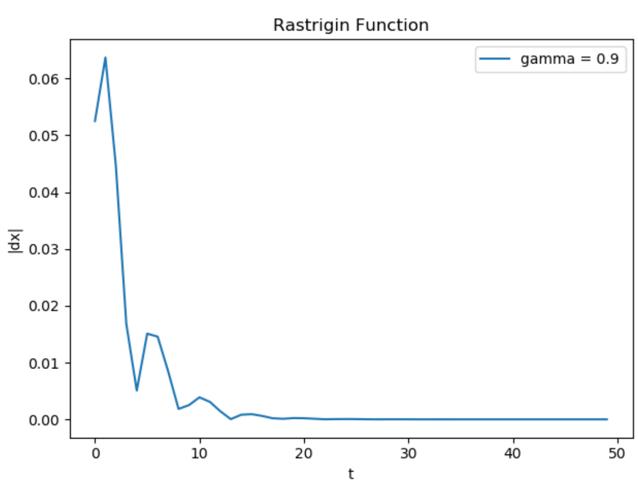


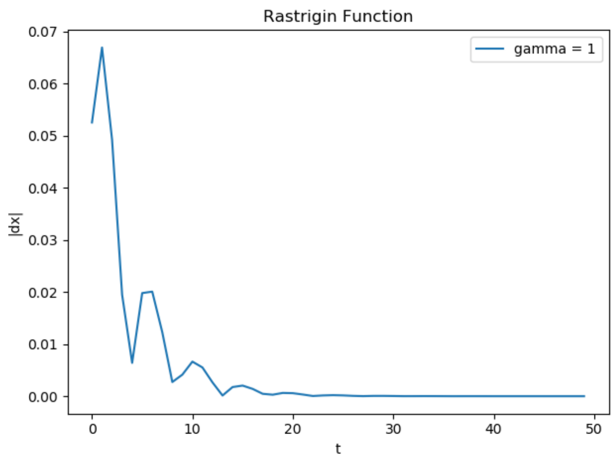


که مقدار ۰.۰۱ همگرا نشده و سرعت همگرایی ۰.۰۰۰۱ نیز کم است. پس η را برابر ۰.۰۰۱ می‌گیریم. (قسمت مربوط به رسم این نمودارها در کد comment شده اند.)

مقایسه gamma در روش Nesterov :



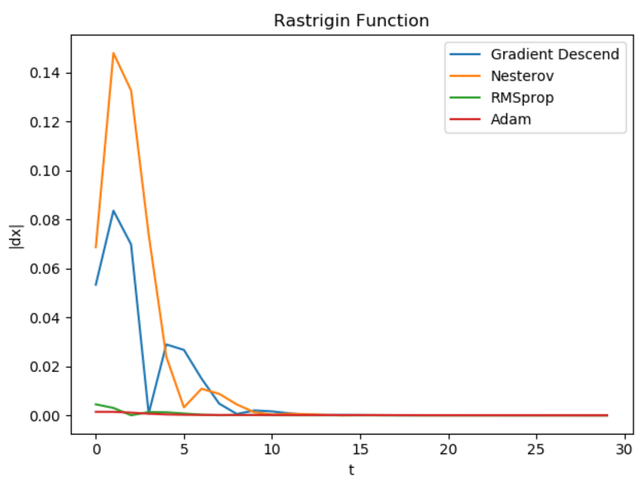




که ملاحظه می‌شود ۰.۹ همگرایی تاحدی سریعتر دارد (در مثال‌های بالا، نقطه شروع یک نقطه ساده مثل (۰.۱، ۰.۱) است).

مقایسه:

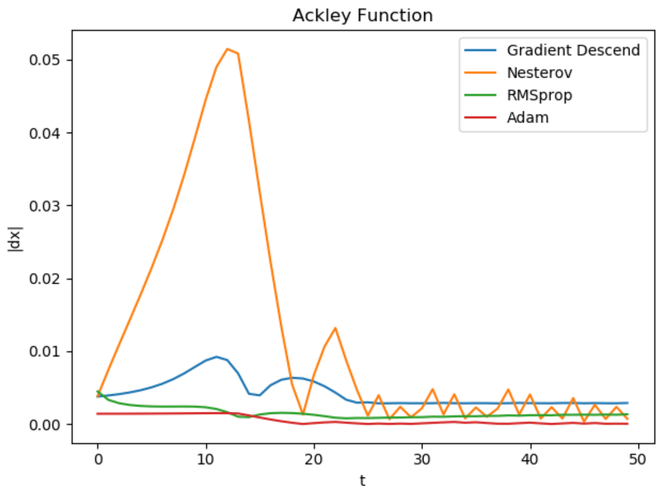
در ۳۰ حرکت، نتایج زیر حاصل شده‌اند:



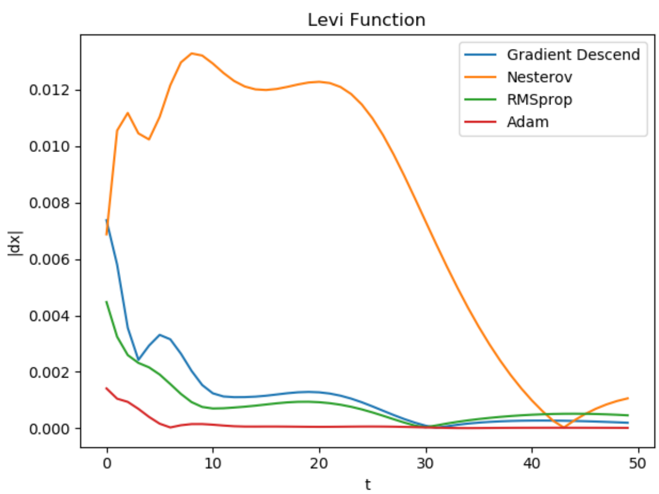
و نتیجه همگرایی:

```
Gradient Descend Last Point: ('-9.129670079060527e-06', '-9.129670079060527e-06')
Nesterov Last Point: ('-1.390215207389888e-07', '-1.390215207389888e-07')
RMSprop Last Point: ('-1.297911609194876e-07', '-1.297911609194876e-07')
Adam Last Point: ('-1.3805904444734007e-05', '-1.3805904444734007e-05')
```

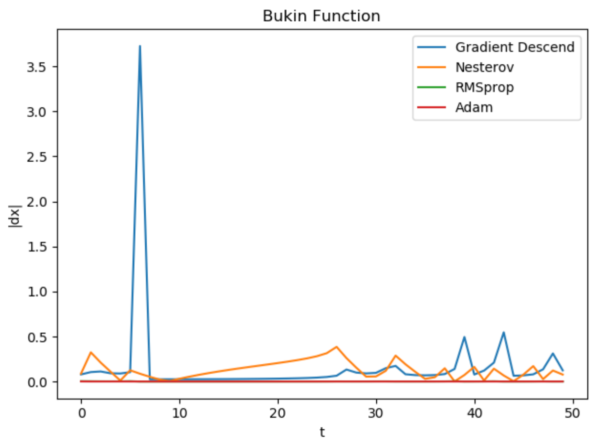
همچنین برای سایر توابع سوال، شکل همگرایی و نقاط نهایی با شروع از نقطه (۰.۴، ۰.۴) به صورت زیر اند:




```
Gradient Descend Last Point: ('0.0012321825892946258', '0.0012321825892946258')
Nesterov Last Point: ('0.0017437057592395936', '0.0017437057592395936')
RMSprop Last Point: ('0.0007943596068777677', '0.0007943596068777677')
Adam Last Point: ('0.000764565466681859', '0.000764565466681859')
```




```
Gradient Descend Last Point: ('0.347971276389936', '0.8137487318113412')
Nesterov Last Point: ('0.3479520487363761', '0.8126864044389059')
RMSprop Last Point: ('0.3479293395651266', '0.8122244871998697')
Adam Last Point: ('0.34792813618511303', '0.8122061444719787')
```




```
Gradient Descend Last Point: ('0.3956829570864227', '0.040707440453546545')
Nesterov Last Point: ('0.39526174222724864', '0.11830154107420456')
RMSprop Last Point: ('0.3960213074837382', '0.11752517171760636')
Adam Last Point: ('0.39603980154942703', '0.11682365269610823')
```

که بصورت کلی، Adam سریعتر به حول نقطه کمینه می‌رسد ولی در آنجا کند می‌شود،
دو الگوریتم Nesterov وRMSpop هر دو نوسان دارند ولی RMSprop معمولا بهتر عمل می‌کند،
و GD نیز نکته خاصی ندارد.

بخش ششم: روش نیوتون

محاسبه ماتریس هسیان، برای بری از توابع سوال به صورت ذیل است:

1) Rastrigin Function:

$$\vec{\nabla} f = [2x_1 + 20\pi \sin(2\pi x_1)]\widehat{x_1} + [2x_2 + 20\pi \sin(2\pi x_2)]\widehat{x_2}$$

$$\Rightarrow \mathrm{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 2 + 40\pi^2 \cos(2\pi x_1) & 0 \\ 0 & 2 + 40\pi^2 \cos(2\pi x_2) \end{bmatrix}$$

2) Ackley Function:

$$\vec{\nabla} f = \left[\frac{2x_1}{\sqrt{0.5(x_1^2 + x_2^2)}} e^{-0.2\sqrt{0.5(x_1^2+x_2^2)}} + \pi \sin(2\pi x_1)\ e^{0.5(\cos(2\pi x_1)+\cos(2\pi x_2))}\right]\widehat{x_1}$$
$$+ \left[\frac{2x_2}{\sqrt{0.5(x_1^2 + x_2^2)}} e^{-0.2\sqrt{0.5(x_1^2+x_2^2)}} + \pi \sin(2\pi x_2)\ e^{0.5(\cos(2\pi x_1)+\cos(2\pi x_2))}\right]\widehat{x_2}$$

$$\Rightarrow \mathrm{H} = \begin{bmatrix} - & - \\ - & - \end{bmatrix}$$

3) Levi Function:

$$\vec{\nabla} f = [6\pi \sin(3\pi x_1)\cos(3\pi x_1) + 2(x_1 - 1)(1 + \sin^2(3\pi x_2))\ ]\widehat{x_1}$$
$$+ [6\pi(x_1 - 1)^2 \sin(3\pi x_2)\cos(3\pi x_2) + 2(x_2 - 1)(1 + \sin^2(2\pi x_2))$$
$$+ 4\pi(x_2 - 1)^2 \sin(2\pi x_2)\cos(2\pi x_2)]\widehat{x_2}$$

$$\Rightarrow \mathrm{H} = \begin{bmatrix} 18\pi^2 \cos(6\pi x_1) + 2(1 + \sin^2(3\pi x_2)) & 12\pi(x_2 - 1)\sin(3\pi x_1)\cos(3\pi x_1) \\ 12\pi(x_2 - 1)\sin(3\pi x_1)\cos(3\pi x_1) & [h_{22}] \end{bmatrix}$$

$$h_{22} = 18\pi^2(x_1 - 1)^2\cos(6\pi x_2) + 4\pi(x_2 - 1)\sin(4\pi x_2) + 2(1 + \sin^2(2\pi x_2))$$
$$+ 8\pi^2(x_2 - 1)^2\cos(4\pi x_2)$$

4) Bukin Function:

$$\vec{\nabla} f = \left[0.01 sign(x_1 + 10) - \frac{sign(x_2 - 0.01x_1)}{2\sqrt{|x_2 - 0.01x_1|}}\right]\widehat{x_1} + [\frac{50 sign(x_2 - 0.01x_1)}{\sqrt{|x_2 - 0.01x_1|}}]\widehat{x_2}$$

$$\Rightarrow \mathrm{H} = \begin{bmatrix} - & - \\ - & - \end{bmatrix}$$

5) n-D Rastrigin Function:

$$\vec{\nabla} f = 2\sum_{i=1}^{n}[x_i + 10\pi \sin(2\pi x_i)]\widehat{x_i}$$
$$\Rightarrow h_{ii} = 2 + 40\pi^2 \cos(2\pi x_i)\,, h_{ij} = 0\ (i \neq j)$$

.... پایان گزارش ....