{

"cells": [

{

"cell\_type": "code",

"execution\_count": 2,

"id": "c577a7f8-7006-4ef9-9e57-7fec8077e4d9",

"metadata": {},

"outputs": [],

"source": [

"import torch"

]

},

{

"cell\_type": "code",

"execution\_count": 4,

"id": "9adc8ce3-c298-4686-9996-8cfc73b20b24",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"x0: tensor(0.0121, requires\_grad=True) y0: tensor(0.0001, grad\_fn=<PowBackward0>) x0.grad: tensor(0.0242)\n"

]

}

],

"source": [

"#例题\n",

"import numpy as np\n",

"import matplotlib.pyplot as plt\n",

"\n",

"x0 = torch.tensor(5.0, requires\_grad=True)\n",

"y0 = x0\*\*2\n",

"\n",

"alpha = 0.1\n",

"epsilon = 0.0001\n",

"\n",

"x\_values = []\n",

"y\_values = []\n",

"\n",

"# 循环直到y的变化小于epsilon\n",

"while True:\n",

" # 计算梯度\n",

" y0.backward()\n",

" y = y0.item() # 保存y0在当前迭代的取值，用于判断迭代的停止条件\n",

" # 更新x0\n",

" with torch.no\_grad():\n",

" x0 -= alpha \* x0.grad # x0 = x0-alpha \* x0.grad 创建一个新的张量并赋值给x\n",

" x0.grad.zero\_()\n",

" # 计算新的y0\n",

" y0 = x0\*\*2\n",

" # 检查是否满足结束条件\n",

" if abs(y0.item() - y) < epsilon:\n",

" break\n",

"\n",

"y0.backward()\n",

"print(\"x0: \", x0, \"y0: \", y0, \"x0.grad: \", x0.grad)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "64f5ea60-ae63-4c77-999c-2101020a9864",

"metadata": {},

"outputs": [],

"source": [

"#设计两个波谷的一元函数，使用梯度下降算法找到全局最小值(作业第一题)"

]

},

{

"cell\_type": "code",

"execution\_count": 21,

"id": "b0b5f73f-a9e5-4853-a593-b5e8a7acbe2d",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"第一个波谷处最小值近似解：\n",

"x0: tensor(0., requires\_grad=True) y0: tensor(1., grad\_fn=<PowBackward0>)\n",

"第二个波谷处最小值近似解：\n",

"x0: tensor(-1.0035, requires\_grad=True) y0: tensor(4.9036e-05, grad\_fn=<PowBackward0>)\n"

]

}

],

"source": [

"# 定义目标函数\n",

"def objective\_function(x):\n",

" return (x\*\*2 - 1)\*\*2\n",

"\n",

"# 用于判断是否陷入局部最小值的计数器及阈值（通用设置）\n",

"stuck\_count\_general = 0\n",

"stuck\_threshold\_general = 100\n",

"\n",

"# 第一个波谷附近求最小值近似解\n",

"x0\_first = torch.tensor(0.0, requires\_grad=True)\n",

"y0\_first = objective\_function(x0\_first)\n",

"\n",

"alpha = 0.01\n",

"epsilon = 0.00001\n",

"\n",

"x\_values\_first = []\n",

"y\_values\_first = []\n",

"\n",

"while True:\n",

" # 计算梯度\n",

" y0\_first.backward()\n",

" y\_first = y0\_first.item() # 保存y0在当前迭代的取值，用于判断迭代的停止条件\n",

" # 更新x0\n",

" with torch.no\_grad():\n",

" x0\_first -= alpha \* x0\_first.grad # x0 = x0-alpha \* x0.grad 创建一个新的张量并赋值给x\n",

" x0\_first.grad.zero\_()\n",

" # 计算新的y0\n",

" y0\_first = objective\_function(x0\_first)\n",

"\n",

" # 检查是否满足结束条件\n",

" if abs(y0\_first.item() - y\_first) < epsilon:\n",

" break\n",

"\n",

" # 判断是否陷入局部最小值\n",

" if abs(y0\_first.item() - y\_first) < 0.0001: # 如果目标函数值变化很小\n",

" stuck\_count\_general += 1\n",

" else:\n",

" stuck\_count\_general = 0\n",

"\n",

" if stuck\_count\_general >= stuck\_threshold\_general: # 如果连续多次目标函数值变化很小，视为陷入局部最小值\n",

" # 重新初始化x0到第一个波谷附近区域\n",

" x0\_first = torch.tensor(0.0, requires\_grad=True)\n",

" y0\_first = objective\_function(x0\_first)\n",

" stuck\_count\_general = 0\n",

"\n",

"# 输出第一个波谷处最小值近似解\n",

"print(\"第一个波谷处最小值近似解：\")\n",

"print(\"x0: \", x0\_first, \"y0: \", y0\_first)\n",

"\n",

"# 第二个波谷附近求最小值近似解\n",

"x0\_second = torch.tensor(-2.0, requires\_grad=True)\n",

"y0\_second = objective\_function(x0\_second)\n",

"\n",

"x\_values\_second = []\n",

"y\_values\_second = []\n",

"\n",

"while True:\n",

" # 计算梯度\n",

" y0\_second.backward()\n",

" y\_second = y0\_second.item() # 保存y0在当前迭代的取值，用于判断迭代的停止条件\n",

" # 更新x0\n",

" with torch.no\_grad():\n",

" x0\_second -= alpha \* x0\_second.grad # x0 = x0-alpha \* x0.grad 创建一个新的张量并赋值给x\n",

" x0\_second.grad.zero\_()\n",

" # 计算新的y0\n",

" y0\_second = objective\_function(x0\_second)\n",

"\n",

" # 检查是否满足结束条件\n",

" if abs(y0\_second.item() - y\_second) < epsilon:\n",

" break\n",

"\n",

" # 判断是否陷入局部最小值\n",

" if abs(y0\_second.item() - y\_second) < 0.0001: # 如果目标函数值变化很小\n",

" stuck\_count\_general += 1\n",

" else:\n",

" stuck\_count\_general = 0\n",

"\n",

" if stuck\_count\_general >= stuck\_threshold\_general: # 如果连续多次目标函数值变化很小，视为陷入局部最小值\n",

" # 重新初始化x0到第二个波谷附近区域\n",

" x0\_second = torch.tensor(-2.0, requires\_grad=True)\n",

" y0\_second = objective\_function(x0\_second)\n",

" stuck\_count\_general = 0\n",

"\n",

"# 输出第二个波谷处最小值近似解\n",

"print(\"第二个波谷处最小值近似解：\")\n",

"print(\"x0: \", x0\_second, \"y0: \", y0\_second)"

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"id": "06806cf4-0c55-43de-b932-98e9f363c448",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"近似全局最小值： 1.0231815394945443e-12\n",

"x值： 0.9999996423721313\n",

"y值： 0.9999996423721313\n"

]

}

],

"source": [

"#第二题\n",

"import numpy as np\n",

"# 定义二元函数\n",

"def f(x,y):\n",

" return (x\*\*2 - 1)\*\*2+(y\*\*2 - 1)\*\*2\n",

"\n",

"# 定义偏导数\n",

"def df\_dx(x):\n",

" return 4\*x\*(x\*\*2 - 1)\n",

"\n",

"def df\_dy(y):\n",

" return 4\*y\*(y\*\*2 - 1)\n",

"\n",

"# 初始化参数\n",

"x = torch.tensor(0.5, requires\_grad=True)\n",

"y = torch.tensor(0.5, requires\_grad=True)\n",

"learning\_rate = 0.01\n",

"num\_iterations = 1000\n",

"\n",

"for i in range(num\_iterations):\n",

" # 计算函数值\n",

" z = f(x,y)\n",

" # 计算梯度\n",

" z.backward()\n",

" # 更新参数\n",

" with torch.no\_grad():\n",

" x -= learning\_rate \* x.grad\n",

" y -= learning\_rate \* y.grad\n",

" # 清零梯度\n",

" x.grad.zero\_()\n",

" y.grad.zero\_()\n",

"\n",

"# 输出近似解\n",

"print(\"近似全局最小值：\", f(x,y).item())\n",

"print(\"x值：\", x.item())\n",

"print(\"y值：\", y.item())"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "2dac46f1-6853-402c-b007-bc21b8dc7d3c",

"metadata": {},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "7b814bfb-bf8a-4d33-932a-5b7295d6e345",

"metadata": {},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "11d3e3de-6a19-4031-98cc-17c144ee3cf8",

"metadata": {},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "4f6f0ac4-4478-46d0-b6a0-97ef946c6257",

"metadata": {},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3 (ipykernel)",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.9.7"

}

},

"nbformat": 4,

"nbformat\_minor": 5

}