{

"cells": [

{

"cell\_type": "code",

"execution\_count": 2,

"id": "f6bae165-026b-4ff8-9b12-e173a53f257f",

"metadata": {},

"outputs": [],

"source": [

"import torch"

]

},

{

"cell\_type": "code",

"execution\_count": 4,

"id": "aafb1638-299b-4ca9-92a2-6fb12cac96af",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"tensor(2.)"

]

},

"execution\_count": 4,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

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

"y = torch.tensor(1.0, requires\_grad=True)\n",

"z = x\*\*2+y\*\*2\n",

"z.backward()\n",

"x.grad\n",

"y.grad"

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"id": "75fd9c22-3ce0-4e3f-8d75-631527492b42",

"metadata": {},

"outputs": [],

"source": [

"import math\n",

"e = math.e"

]

},

{

"cell\_type": "code",

"execution\_count": 6,

"id": "d98fcb30-35a5-4bfc-b27f-938ac94086ac",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"tensor(-1.)"

]

},

"execution\_count": 6,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"def func(x):\n",

" return 1/1+e\*\*-x\n",

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

"y = func(x)\n",

"y.backward()\n",

"x.grad"

]

},

{

"cell\_type": "code",

"execution\_count": 7,

"id": "cdfe7e03-2f48-4f0a-bb74-bcc29898ada7",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"关于x1的偏导数: tensor([0.], grad\_fn=<MulBackward0>)\n",

"关于x2的偏导数: tensor([0.0029], grad\_fn=<MulBackward0>)\n"

]

}

],

"source": [

"# 定义sigmoid函数\n",

"def sigmoid(x):\n",

" return 1 / (1 + torch.exp(-x))\n",

"\n",

"\n",

"# 定义函数\n",

"def func(x1, x2):\n",

" inner\_sigmoid = sigmoid(x1 \*\* 2 + 2 \* x2 + 1)\n",

" y = sigmoid(3 \* inner\_sigmoid + inner\_sigmoid + 1)\n",

" return y\n",

"\n",

"\n",

"# 设置x1和x2的值\n",

"x1 = torch.tensor([0.0], requires\_grad=True)\n",

"x2 = torch.tensor([1.0], requires\_grad=True)\n",

"\n",

"# 计算y的值\n",

"y = func(x1, x2)\n",

"\n",

"# 手动计算关于x1的偏导数\n",

"dy\_dx1 = torch.autograd.grad(y, x1, create\_graph=True)[0]\n",

"print(f\"关于x1的偏导数: {dy\_dx1}\")\n",

"\n",

"# 手动计算关于x2的偏导数\n",

"dy\_dx2 = torch.autograd.grad(y, x2, create\_graph=True)[0]\n",

"print(f\"关于x2的偏导数: {dy\_dx2}\")"

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"id": "a5eb1997-7de2-4e70-b37e-e5a8777044e4",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"关于x1的偏导数: tensor([0.], grad\_fn=<MulBackward0>)\n",

"关于x2的偏导数: tensor([0.0029], grad\_fn=<MulBackward0>)\n",

"节点1关于x1的偏导数: tensor([0.], grad\_fn=<MulBackward0>)\n",

"节点2关于x1的偏导数: tensor([0.], grad\_fn=<MulBackward0>)\n",

"节点1关于x2的偏导数: tensor([2.])\n",

"节点2关于x2的偏导数: tensor([0.3614], grad\_fn=<MulBackward0>)\n"

]

}

],

"source": [

"# 定义sigmoid函数\n",

"def sigmoid(x):\n",

" return 1 / (1 + torch.exp(-x))\n",

"\n",

"\n",

"# 定义函数\n",

"def func(x1, x2):\n",

" # 第一个关键计算节点：x1\*\*2 + 2\*x2 + 1\n",

" node1 = x1 \*\* 2 + 2 \* x2 + 1\n",

" node1.requires\_grad\_(True)\n",

"\n",

" inner\_sigmoid = sigmoid(node1)\n",

"\n",

" # 第二个关键计算节点：3 \* inner\_sigmoid + inner\_sigmoid + 1\n",

" node2 = 3 \* inner\_sigmoid + inner\_sigmoid + 1\n",

" node2.requires\_grad\_(True)\n",

"\n",

" y = sigmoid(node2)\n",

"\n",

" return y, [node1, node2]\n",

"\n",

"\n",

"# 设置x1和x2的值\n",

"x1 = torch.tensor([0.0], requires\_grad=True)\n",

"x2 = torch.tensor([1.0], requires\_grad=True)\n",

"\n",

"# 计算y的值以及关键节点列表\n",

"y, nodes = func(x1, x2)\n",

"\n",

"# 关键节点关于x1和x2的偏导数列表\n",

"d\_nodes\_dx1 = []\n",

"d\_nodes\_dx2 = []\n",

"\n",

"# 遍历关键节点列表，计算每个节点关于x1和x2的偏导数\n",

"for node in nodes:\n",

" d\_node\_dx1 = torch.autograd.grad(node, x1, create\_graph=True)[0]\n",

" d\_nodes\_dx1.append(d\_node\_dx1)\n",

" d\_node\_dx2 = torch.autograd.grad(node, x2, create\_graph=True)[0]\n",

" d\_nodes\_dx2.append(d\_node\_dx2)\n",

"\n",

"# 输出关于x1的偏导数\n",

"print(f\"关于x1的偏导数: {torch.autograd.grad(y, x1, create\_graph=True)[0]}\")\n",

"\n",

"# 输出关于x2的偏导数\n",

"print(f\"关于x2的偏导数: {torch.autograd.grad(y, x2, create\_graph=True)[0]}\")\n",

"\n",

"# 输出每个节点关于x1的偏导数\n",

"for i, d\_node\_dx1 in enumerate(d\_nodes\_dx1):\n",

" print(f\"节点{i + 1}关于x1的偏导数: {d\_node\_dx1}\")\n",

"\n",

"# 输出每个节点关于x2的偏导数\n",

"for i, d\_node\_dx2 in enumerate(d\_nodes\_dx2):\n",

" print(f\"节点{i + 1}关于x2的偏导数: {d\_node\_dx2}\")"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "0e34986b-998f-430b-bf93-37a81f3411fc",

"metadata": {},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "90964513-544d-4f5e-bc93-258810eb69c0",

"metadata": {},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "94d53059-e986-4336-95ac-f1670987641d",

"metadata": {},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "e05c6eed-9226-441a-be08-54f670939042",

"metadata": {},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "cca52af3-fcac-4edf-8441-e09e88755ffc",

"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

}