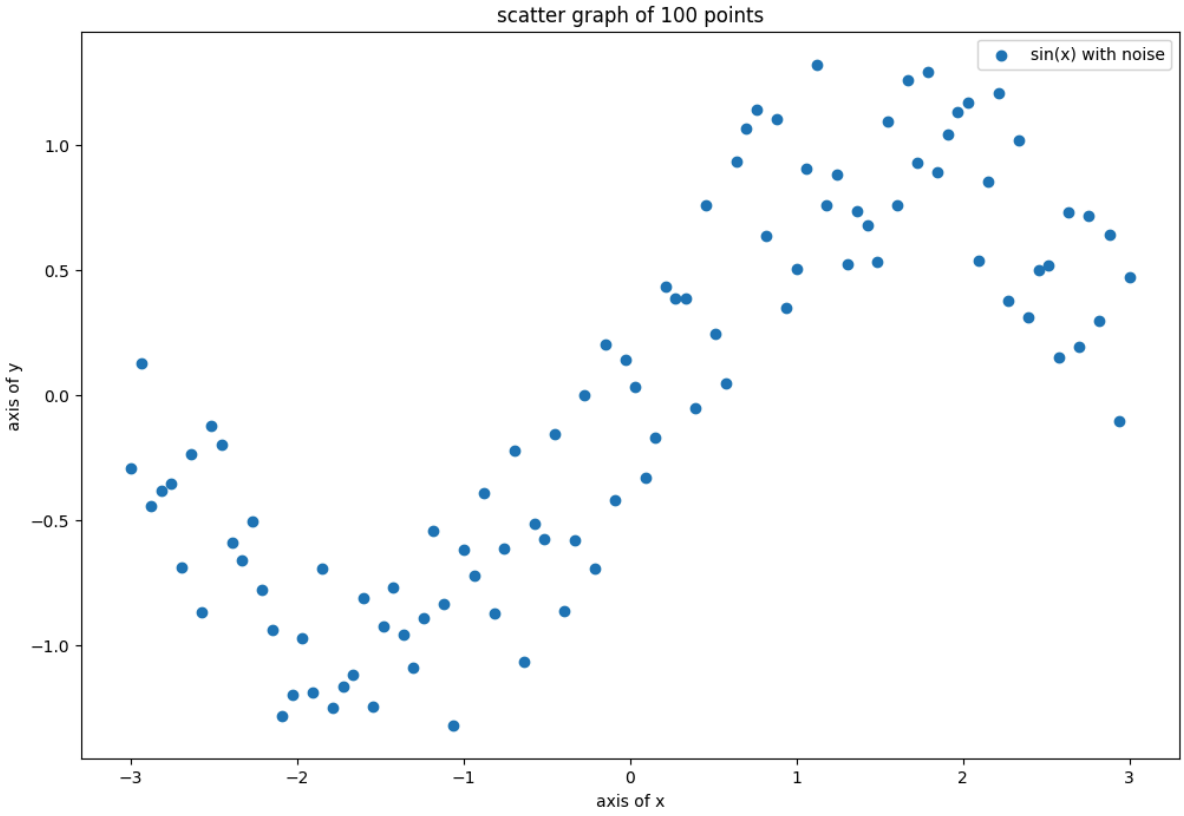
1、线性回归

(1) 生成训练数据，数据为带有服从-0.5 到 0.5 的均匀分布噪声的正弦函数，画出这 100 个样本的散点图。（提交散点图）

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
num_observations=100
train_x=np.linspace(-3,3,num_observations)
train_y=np.sin(train_x)+np.random.uniform(-0.5,0.5,num_observations)
plt.figure(figsize=(12,8))
plt.scatter(train_x,train_y,label='sin(x) with noise')
plt.xlabel("axis of x")
plt.ylabel("axis of y")
plt.title("scatter graph of 100 points")
plt.legend()
plt.show()
```



(2) 使用pytorch实现线性回归模型，训练参数w和b。

nn.Module属性和方法： 1.parameters()：返回模型的所有可训练参数（权重和偏置），用于优化器。 2.cuda()：将模型转移到 GPU 进行计算。 3.state_dict()：返回模型的所有参数（包括权重和偏置）及其对应的值，常用于保存和加载模型。 4.load_state_dict()：加载一个存储的模型参数字典。 5.train()：将模型设置为训练模式（默认状态）。这会影响某些层（如 Dropout 和 BatchNorm）的行为。 6.eval()：将模型设置为评估模式，用于测试时，固定某些层的行为。 7.forward()：前向传播的定义，用户需要在模型中重写此方法，定义模型的前向计算逻辑。

主要解决流程： 1.自定义模型类（初始化+定义前向传播过程） 2.确定损失函数 3.确定优化器 4.for循环（迭代次数） 5.前向传播: model(x_now) 6.计算损失 7.计算梯度: loss.backward() 8.反向传播：optimizer.step()

In [2]:
```
import torch
from torch import nn, optim
class LinearModel(nn.Module):
    def __init__(self):
        super(LinearModel, self).__init__()
        self.fc1 = nn.Linear(2, 1)

    def forward(self, x):
        bias = torch.tensor([1]).cuda()
        x = x.unsqueeze(0)
        x = torch.stack((x,bias),dim=1)
        x = x.reshape((1,2))
        output = self.fc1(x)
        return output

model = LinearModel().cuda()
criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=1e-6)
max_epoch=200
epoch_loss=[] #每一次迭代之后的损失值
weights_and_bias=[]

for epoch in range(max_epoch):
    optimizer.zero_grad()
    for x,y in zip(train_x,train_y):
        x, y = torch.tensor(x).float().cuda(), torch.tensor(y).float().cuda()
        output = model(x)
        loss = criterion(output, y)
        loss.backward()
        optimizer.step()
    w=model.fc1.weight.clone().detach().cpu().numpy()  #使用detach分离梯度追踪
    b=model.fc1.bias.clone().detach().cpu().numpy()
    #b是偏置项，用于调整模型输出。
    weights_and_bias.append(w)
    epoch_loss.append(loss.item())
print(np.array(weights_and_bias).shape)
```

```
/home/cs/anaconda3/envs/pytorch/lib/python3.11/site-packages/torch/nn/modules/loss.py:536: UserWarning: Using a target size (torch.Size([])) that is different to the input size (torch.Size([1, 1])). This will likely lead to incorrect results due to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
(200, 1, 2)
```

(3) 输出参数*w*、*b*和损失。（提交运行结果）

将（2）训练过程中的w,b和损失保存在列表中，将其打印

In [3]:
```
iter=len(epoch_loss)
for i in range(iter):
    print(f"第{i+1}次迭代  参数w为:{weights_and_bias[i][0][0]}  参数b为:{weights_and_
```

第1次迭代　参数w为:-0.3430972695350647　参数b为:-0.678205668926239　loss为:2.6101813316345215
第2次迭代　参数w为:-0.32337868213653564　参数b为:-0.6840282082557678　loss为:2.4588205814361572
第3次迭代　参数w为:-0.30438506603240967　参数b为:-0.6895354390144348　loss为:2.3166823387145996
第4次迭代　参数w为:-0.2860875725746155　参数b为:-0.6947400569915771　loss为:2.183159112930298
第5次迭代　参数w为:-0.26845866441726685　参数b为:-0.6996561884880066　loss为:2.0576984882354736
第6次迭代　参数w为:-0.25147172808647156　参数b为:-0.7042956352233887　loss为:1.9397755861282349
第7次迭代　参数w为:-0.235101580619812　参数b为:-0.7086706161499023　loss为:1.8289059400558472
第8次迭代　参数w为:-0.21932393312454224　参数b为:-0.7127928137779236　loss为:1.7246407270431519
第9次迭代　参数w为:-0.20411546528339386　参数b为:-0.7166731357574463　loss为:1.6265573501586914
第10次迭代　参数w为:-0.18945367634296417　参数b为:-0.7203222513198853　loss为:1.5342634916305542
第11次迭代　参数w为:-0.1753171980381012　参数b为:-0.7237507104873657　loss为:1.4473965167999268
第12次迭代　参数w为:-0.16168524324893951　参数b为:-0.7269676327705383　loss为:1.3656094074249268
第13次迭代　参数w为:-0.14853809773921967　参数b为:-0.7299824357032776　loss为:1.2885856628417969
第14次迭代　参数w为:-0.1358567476272583　参数b为:-0.7328039407730103　loss为:1.2160272598266602
第15次迭代　参数w为:-0.1236228495836258　参数b为:-0.7354410290718079　loss为:1.1476577520370483
第16次迭代　参数w为:-0.11181904375553131　参数b为:-0.7379019856452942　loss为:1.0832196474075317
第17次迭代　参数w为:-0.10042846202850342　参数b为:-0.7401944398880005　loss为:1.0224689245224
第18次迭代　参数w为:-0.08943501859903336　参数b为:-0.7423258423805237　loss为:0.9651801586151123
第19次迭代　参数w为:-0.07882319390773773　参数b为:-0.7443041205406189　loss为:0.9111442565917969
第20次迭代　参数w为:-0.06857817620038986　参数b为:-0.746135413646698　loss为:0.8601616024971008
第21次迭代　参数w为:-0.058685656636953354　参数b为:-0.7478266358375549　loss为:0.8120480179786682
第22次迭代　参数w为:-0.049132030457258224　参数b为:-0.749384343624115　loss为:0.7666317224502563
第23次迭代　参数w为:-0.03990411013364792　参数b为:-0.7508144974708557　loss为:0.7237511873245239
第24次迭代　参数w为:-0.030989298596978188　参数b为:-0.7521231174468994　loss为:0.6832557320594788
第25次迭代　参数w为:-0.022375518456101418　参数b为:-0.7533154487609863　loss为:0.6450040340423584
第26次迭代　参数w为:-0.014051147736608982　参数b为:-0.7543976306915283　loss为:0.6088647842407227
第27次迭代　参数w为:-0.0060050394386053085　参数b为:-0.7553740739822388　loss为:0.5747125744819641
第28次迭代　参数w为:0.001773511990904808　参数b为:-0.7562499046325684　loss为:0.5424323081970215
第29次迭代　参数w为:0.00929476972669363　参数b为:-0.7570295929908752　loss为:0.5119147300720215
第30次迭代　参数w为:0.01656859740614891　参数b为:-0.7577179670333862　loss为:0.4830591380596161
第31次迭代　参数w为:0.023604458197951317　参数b为:-0.7583192586898804　loss为:0.4557693004608154
3
第32次迭代　参数w为:0.030411435291171074　参数b为:-0.7588376998901367　loss为:0.4299560487270355

第33次迭代　参数w为:0.036998238414525986　参数b为:-0.7592771649360657　loss为:0.4055
3587675094604
第34次迭代　参数w为:0.04337328299880028　参数b为:-0.7596413493156433　loss为:0.38242
945075035095
第35次迭代　参数w为:0.049544576555490494　参数b为:-0.7599344253540039　loss为:0.3605
634272098541
第36次迭代　参数w为:0.055519815534353256　参数b为:-0.7601591944694519　loss为:0.3398
682177066803
第37次迭代　参数w为:0.061306461691856384　参数b为:-0.760319173336029　loss为:0.32027
87935733795
第38次迭代　参数w为:0.06691159307956696　参数b为:-0.760418176651001　loss为:0.301734
44747924805
第39次迭代　参数w为:0.07234204560518265　参数b为:-0.760458767414093　loss为:0.284177
5715351105
第40次迭代　参数w为:0.07760433852672577　参数b为:-0.7604435086250305　loss为:0.26755
380630493164
第41次迭代　参数w为:0.08270484209060669　参数b为:-0.7603757977485657　loss为:0.25181
26368522644
第42次迭代　参数w为:0.08764954656362534　参数b为:-0.760258138179779　loss为:0.236906
70728683472
第43次迭代　参数w为:0.0924442708492279　参数b为:-0.760093092918396　loss为:0.2227908
6709022522
第44次迭代　参数w为:0.09709461033344269　参数b为:-0.7598831057548523　loss为:0.20942
296087741852
第45次迭代　参数w为:0.10160589963197708　参数b为:-0.7596306204795837　loss为:0.19676
315784454346
第46次迭代　参数w为:0.10598330944776535　参数b为:-0.7593382000923157　loss为:0.18477
441370487213
第47次迭代　参数w为:0.11023172736167908　参数b为:-0.7590073347091675　loss为:0.17342
106997966766
第48次迭代　参数w为:0.1143558993935585　参数b为:-0.7586401104927063　loss为:0.162669
91198062897
第49次迭代　参数w为:0.11836044490337372　参数b为:-0.7582388520240784　loss为:0.15249
00197982788
第50次迭代　参数w为:0.12224967032670975　参数b为:-0.757805347442627　loss为:0.142852
0679473877
第51次迭代　参数w为:0.12602779269218445　参数b为:-0.7573415040969849　loss为:0.13372
79975414276
第52次迭代　参数w为:0.12969882786273956　参数b为:-0.756848931312561　loss为:0.125091
65704250336
第53次迭代　参数w为:0.13326668739318848　参数b为:-0.7563295364379883　loss为:0.11691
838502883911
第54次迭代　参数w为:0.13673508167266846　参数b为:-0.7557846307754517　loss为:0.10918
498784303665
第55次迭代　参数w为:0.14010755717754364　参数b为:-0.7552159428596497　loss为:0.10186
944901943207
第56次迭代　参数w为:0.14338771998882294　参数b为:-0.7546245455741882　loss为:0.09495
039284229279
第57次迭代　参数w为:0.1465786248445511　参数b为:-0.7540122270584106　loss为:0.088408
84268283844
第58次迭代　参数w为:0.14968357980251312　参数b为:-0.7533800005912781　loss为:0.08222
588151693344
第59次迭代　参数w为:0.15270568430423737　参数b为:-0.7527295351028442　loss为:0.07638
422399759293
第60次迭代　参数w为:0.15564779937267303　参数b为:-0.7520619034767151　loss为:0.07086
717337369919
第61次迭代　参数w为:0.15851262211799622　参数b为:-0.7513779401779175　loss为:0.06565
912812948227
第62次迭代　参数w为:0.1613030731678009　参数b为:-0.750678539276123　loss为:0.0607448
4810233116
第63次迭代　参数w为:0.1640215516090393　参数b为:-0.7499651312828064　loss为:0.056110
76578497887
第64次迭代　参数w为:0.16667060554027557　参数b为:-0.7492390871047974　loss为:0.05174
364894628525

第65次迭代　参数w为:0.1692526787519455　参数b为:-0.7485010623931885　loss为:0.0476306714117527
第66次迭代　参数w为:0.17177006602287292　参数b为:-0.7477520704269409　loss为:0.04375996068120003
第67次迭代　参数w为:0.17422501742839813　参数b为:-0.7469924688339233　loss为:0.0401199609041214
第68次迭代　参数w为:0.17661963403224945　参数b为:-0.7462237477302551　loss为:0.03670024499297142
第69次迭代　参数w为:0.17895594239234924　参数b为:-0.7454462647438049　loss为:0.033490560948848724
第70次迭代　参数w为:0.18123595416545868　参数b为:-0.7446608543395996　loss为:0.030481165274977684
第71次迭代　参数w为:0.1834614872932434　参数b为:-0.7438684105873108　loss为:0.027663104236125946
第72次迭代　参数w为:0.18563446402549744　参数b为:-0.7430698275566101　loss为:0.025027625262737274
第73次迭代　参数w为:0.18775661289691925　参数b为:-0.7422655820846558　loss为:0.022566056065583229
第74次迭代　参数w为:0.18982955813407898　参数b为:-0.7414560317993164　loss为:0.020271876826882362
第75次迭代　参数w为:0.19185492396354675　参数b为:-0.7406418919563293　loss为:0.018136126920580864
第76次迭代　参数w为:0.19383427500724792　参数b为:-0.7398237586021423　loss为:0.016152331605553627
第77次迭代　参数w为:0.19576914608478546　参数b为:-0.739001989364624　loss为:0.014313571155071259
第78次迭代　参数w为:0.19766096770763397　参数b为:-0.7381773591041565　loss为:0.012613641098144167
第79次迭代　参数w为:0.1995111107826233　参数b为:-0.7373504042625427　loss为:0.01104643102735281
第80次迭代　参数w为:0.20132093131542206　参数b为:-0.7365214824676514　loss为:0.009606100618839264
第81次迭代　参数w为:0.2030915766954422　参数b为:-0.7356909513473511　loss为:0.008287144824862486248
第82次迭代　参数w为:0.20482449233531952　参数b为:-0.7348594665527344　loss为:0.007084285374730825
第83次迭代　参数w为:0.20652082562446594　参数b为:-0.7340273261070251　loss为:0.005992462392896414
第84次迭代　参数w为:0.20818156003952026　参数b为:-0.733194887638092　loss为:0.005007013212889433
第85次迭代　参数w为:0.20980794727802277　参数b为:-0.7323627471923828　loss为:0.004123321734368801
第86次迭代　参数w为:0.21140098571777344　参数b为:-0.7315307855606079　loss为:0.003336968133226037
第87次迭代　参数w为:0.21296165883541107　参数b为:-0.7306996583938599　loss为:0.002643872750923037 5
第88次迭代　参数w为:0.21449097990989685　参数b为:-0.7298696637153625　loss为:0.002040071180090308
第89次迭代　参数w为:0.215989887714386　参数b为:-0.729040801525116　loss为:0.0015217233449220657
第90次迭代　参数w为:0.2174593061208725　参数b为:-0.7282139658927917　loss为:0.0010853087296709418
第91次迭代　参数w为:0.21890021860599518　参数b为:-0.7273883819580078　loss为:0.0007272024522535503
第92次迭代　参数w为:0.22031335532665253　参数b为:-0.7265649437904358　loss为:0.0004442414792720228 4
第93次迭代　参数w为:0.22169940173625946　参数b为:-0.7257438898086548　loss为:0.0002332837902940809 7
第94次迭代　参数w为:0.22305938601493835　参数b为:-0.72492516040802　loss为:9.126238001044840e-05
第95次迭代　参数w为:0.22439387440681458　参数b为:-0.7241095900535583　loss为:1.5329185771406628e-05
第96次迭代　参数w为:0.22570371627807617　参数b为:-0.7232967019081116　loss为:2.6994557629222 98e-06

第97次迭代　参数w为:0.22698956727981567　参数b为:-0.7224875092506409　loss为:5.0724298489512876e-05
第98次迭代　参数w为:0.228252112865448　参数b为:-0.7216812372207642　loss为:0.00015688066196162254
第99次迭代　参数w为:0.22949199378490448　参数b为:-0.7208787202835083　loss为:0.0003187239926774055
第100次迭代　参数w为:0.23070983588695526　参数b为:-0.7200801372528076　loss为:0.000533908314537257
第101次迭代　参数w为:0.23190633952617645　参数b为:-0.7192853689193726　loss为:0.0008002282120287418
第102次迭代　参数w为:0.23308193683624268　参数b为:-0.7184944152832031　loss为:0.0011155559914186597
第103次迭代　参数w为:0.23423725366592407　参数b为:-0.7177079916000366　loss为:0.0014777473406866193
第104次迭代　参数w为:0.23537281155586243　参数b为:-0.7169255018234253　loss为:0.0018849355401471257
第105次迭代　参数w为:0.23648914694786072　参数b为:-0.7161474227905273　loss为:0.00233517331071198
第106次迭代　参数w为:0.23758676648139954　参数b为:-0.715373694896698　loss为:0.00282669672742486
第107次迭代　参数w为:0.2386660873889923　参数b为:-0.714604377746582　loss为:0.0033577282447367907
第108次迭代　参数w为:0.2397276908159256　参数b为:-0.713840126991272　loss为:0.0039264969527721405
第109次迭代　参数w为:0.24077194929122925　参数b为:-0.7130805253982544　loss为:0.004531483631581068
第110次迭代　参数w为:0.24179932475090027　参数b为:-0.712325394153595　loss为:0.005171214230358601
第111次迭代　参数w为:0.24281033873558044　参数b为:-0.7115753293037415　loss为:0.005844118073582649
第112次迭代　参数w为:0.2438051700592041　参数b为:-0.7108299136161804　loss为:0.006548786070197821
第113次迭代　参数w为:0.2447844296693802　参数b为:-0.7100895047187805　loss为:0.007283865474164486
第114次迭代　参数w为:0.24574846029281616　参数b为:-0.7093544006347656　loss为:0.008047937415540218
第115次迭代　参数w为:0.24669760465621948　参数b为:-0.7086242437362671　loss为:0.008839982889354229
第116次迭代　参数w为:0.24763216078281403　参数b为:-0.7078994512557983　loss为:0.009658164344727993
第117次迭代　参数w为:0.24855254590511322　参数b为:-0.7071799039840698　loss为:0.010501843877136707
第118次迭代　参数w为:0.24945907294750214　参数b为:-0.706465482711792　loss为:0.011369784362614155
第119次迭代　参数w为:0.25035208463668823　参数b为:-0.7057565450668335　loss为:0.012260735034942627
第120次迭代　参数w为:0.2512318193912506　参数b为:-0.7050527334213257　loss为:0.013173834344207406
第121次迭代　参数w为:0.2520986497402191　参数b为:-0.7043545246124268　loss为:0.014107842929661274
第122次迭代　参数w为:0.25295302271842957　参数b为:-0.703661322593689　loss为:0.015062144957482815
第123次迭代　参数w为:0.25379478931427　参数b为:-0.7029736638069153　loss为:0.016035355627536774
第124次迭代　参数w为:0.25462448596954346　参数b为:-0.7022914290428162　loss为:0.017026707530021667
第125次迭代　参数w为:0.2554425001144409　参数b为:-0.7016144394874573　loss为:0.018035544082522392
第126次迭代　参数w为:0.2562487721443176　参数b为:-0.7009429931640625　loss为:0.019060637801885605
第127次迭代　参数w为:0.2570438086986542　参数b为:-0.7002767324447632　loss为:0.020101523026823997
第128次迭代　参数w为:0.2578277885913849　参数b为:-0.6996164917945862　loss为:0.021156949922442436

第129次迭代　参数w为:0.25860095024108887　参数b为:-0.6989611983299255　loss为:0.0222
26722911000252
第130次迭代　参数w为:0.25936344265937805　参数b为:-0.6983111500740051　loss为:0.0233
0988459289074
第131次迭代　参数w为:0.26011547446250916　参数b为:-0.6976670622825623　loss为:0.0244
05326694250107
第132次迭代　参数w为:0.2608572542667389　参数b为:-0.6970279812812805　loss为:0.02551
284059882164
第133次迭代　参数w为:0.26158928871154785　参数b为:-0.6963942050933838　loss为:0.0266
3189359009266
第134次迭代　参数w为:0.26231133937835693　参数b为:-0.6957661509513855　loss为:0.0277
6123583316803
第135次迭代　参数w为:0.2630236744880676　参数b为:-0.6951435208320618　loss为:0.02890
0370001792908
第136次迭代　参数w为:0.2637269198894501　参数b为:-0.6945258975028992　loss为:0.03004
947118461132
第137次迭代　参数w为:0.26442083716392517　参数b为:-0.6939137578010559　loss为:0.0312
07239255309105
第138次迭代　参数w为:0.2651056945323944　参数b为:-0.6933068037033081　loss为:0.03237
331658601761
第139次迭代　参数w为:0.26578158140182495　参数b为:-0.6927052140235901　loss为:0.0335
4702144861221
第140次迭代　参数w为:0.2664487063884735　参数b为:-0.6921089291572571　loss为:0.03472
784906625748
第141次迭代　参数w为:0.2671075463294983　参数b为:-0.691518247127533　loss为:0.035915
41200876236
第142次迭代　参数w为:0.26775801181793213　参数b为:-0.69093257188797　loss为:0.037109
438329935074
第143次迭代　参数w为:0.26840007305145264　参数b为:-0.6903524398803711　loss为:0.0383
08750838041306
第144次迭代　参数w为:0.26903393864631653　参数b为:-0.6897775530815125　loss为:0.0395
13181895017624
第145次迭代　参数w为:0.26966002583503723　参数b为:-0.6892074942588806　loss为:0.0407
2292149066925
第146次迭代　参数w为:0.2702782452106476　参数b为:-0.6886427998542786　loss为:0.04193
686693906784
第147次迭代　参数w为:0.2708887457847595　参数b为:-0.688083291053772　loss为:0.043154
62335944176
第148次迭代　参数w为:0.271491676568985　参数b为:-0.6875293254852295　loss为:0.044375
643134117126
第149次迭代　参数w为:0.2720872759819031　参数b为:-0.6869803071022034　loss为:0.04560
002684593201
第150次迭代　参数w为:0.27267566323280334　参数b为:-0.6864363551139832　loss为:0.0468
27346086502075
第151次迭代　参数w为:0.27325671911239624　参数b为:-0.6858973503112793　loss为:0.0480
5690422654152
第152次迭代　参数w为:0.27383074164390564　参数b为:-0.6853634119033813　loss为:0.0492
8857833147049
第153次迭代　参数w为:0.2743980288505554　参数b为:-0.6848339438438416　loss为:0.05052
2711127996445
第154次迭代　参数w为:0.2749583125114441　参数b为:-0.6843096017837524　loss为:0.05175
791308283806
第155次迭代　参数w为:0.2755119204521179　参数b为:-0.6837902665138245　loss为:0.05299
412086606026
第156次迭代　参数w为:0.27605894207954407　参数b为:-0.6832761168479919　loss为:0.0542
3089861869812
第157次迭代　参数w为:0.27659958600997925　参数b为:-0.6827667355537415　loss为:0.0554
6841025352478
第158次迭代　参数w为:0.2771335244178772　参数b为:-0.6822620630264282　loss为:0.05670
5914437770844
第159次迭代　参数w为:0.2776613235473633　参数b为:-0.6817623376846313　loss为:0.05794
329196214676
第160次迭代　参数w为:0.2781829833984375　参数b为:-0.6812670826911926　loss为:0.05918
077751994133

第161次迭代　参数w为:0.27869856357574463　参数b为:-0.680776834487915　loss为:0.06041
7503118515015
第162次迭代　参数w为:0.27920830249786377　参数b为:-0.6802915334701538　loss为:0.0616
532526910305
第163次迭代　参数w为:0.27971214056015015　参数b为:-0.679810643196106　loss为:0.06288
830190896988
第164次迭代　参数w为:0.2802099287509918　参数b为:-0.679334282875061　loss为:0.064121
82748317719
第165次迭代　参数w为:0.2807020843029022　参数b为:-0.6788626313209534　loss为:0.06535
380333662033
第166次迭代　参数w为:0.28118857741355896　参数b为:-0.6783956289291382　loss为:0.0665
8390909433365
第167次迭代　参数w为:0.2816694378852844　参数b为:-0.6779329180717468　loss为:0.06781
225651502609
第168次迭代　参数w为:0.28214478492736816　参数b为:-0.6774746179580688　loss为:0.0690
3848797082901
第169次迭代　参数w为:0.28261464834213257　参数b为:-0.6770212054252625　loss为:0.0702
6179879903793
第170次迭代　参数w为:0.2830792963504791　参数b为:-0.6765720248222351　loss为:0.07148
291915655136
第171次迭代　参数w为:0.2835385203361511　参数b为:-0.676127016544342　loss为:0.072701
31260156631
第172次迭代　参数w为:0.2839926481246948　参数b为:-0.6756865978240967　loss为:0.07391
667366027832
第173次迭代　参数w为:0.284441739320755　参数b为:-0.6752503514289856　loss为:0.075129
30780649185
第174次迭代　参数w为:0.28488561511039734　参数b为:-0.6748184561729431　loss为:0.0763
382688164711
第175次迭代　参数w为:0.28532469272613525　参数b为:-0.6743910312652588　loss为:0.0775
4389941692352
第176次迭代　参数w为:0.2857587933540344　参数b为:-0.6739675402641296　loss为:0.07874
619960784912
第177次迭代　参数w为:0.2861880958080292　参数b为:-0.6735482215881348　loss为:0.07994
484156370163
第178次迭代　参数w为:0.28661248087882996　参数b为:-0.6731330156326294　loss为:0.0811
3935589790344
第179次迭代　参数w为:0.2870321571826935　参数b为:-0.672721803188324　loss为:0.082330
17474412918
第180次迭代　参数w为:0.28744709491729736　参数b为:-0.6723145246505737　loss为:0.0835
1688832044601
第181次迭代　参数w为:0.28785762190818787　参数b为:-0.6719111204147339　loss为:0.0846
998393535614
第182次迭代　参数w为:0.288263738155365　参数b为:-0.6715120077133179　loss为:0.085878
23063135147
第183次迭代　参数w为:0.2886650860309601　参数b为:-0.6711167693138123　loss为:0.08705
18758893013
第184次迭代　参数w为:0.28906211256980896　参数b为:-0.6707258820533752　loss为:0.0882
2057396173477
第185次迭代　参数w为:0.289454847574234　参数b为:-0.6703384518623352　loss为:0.089385
15186309814
第186次迭代　参数w为:0.28984320163726807　参数b为:-0.6699549555778503　loss为:0.0905
4478257894516
第187次迭代　参数w为:0.2902275323867798　参数b为:-0.6695752143859863　loss为:0.09169
996529817581
第188次迭代　参数w为:0.2906074821949005　参数b为:-0.6691990494728088　loss为:0.09285
02008318901
第189次迭代　参数w为:0.2909832000732422　参数b为:-0.6688268184661865　loss为:0.09399
495273828506
第190次迭代　参数w为:0.2913551330566406　参数b为:-0.6684584021568298　loss为:0.09513
49213719368
第191次迭代　参数w为:0.2917228937149048　参数b为:-0.6680934429168701　loss为:0.09626
973420381546
第192次迭代　参数w为:0.29208678007125854　参数b为:-0.6677321791648865　loss为:0.0973
9941358566284

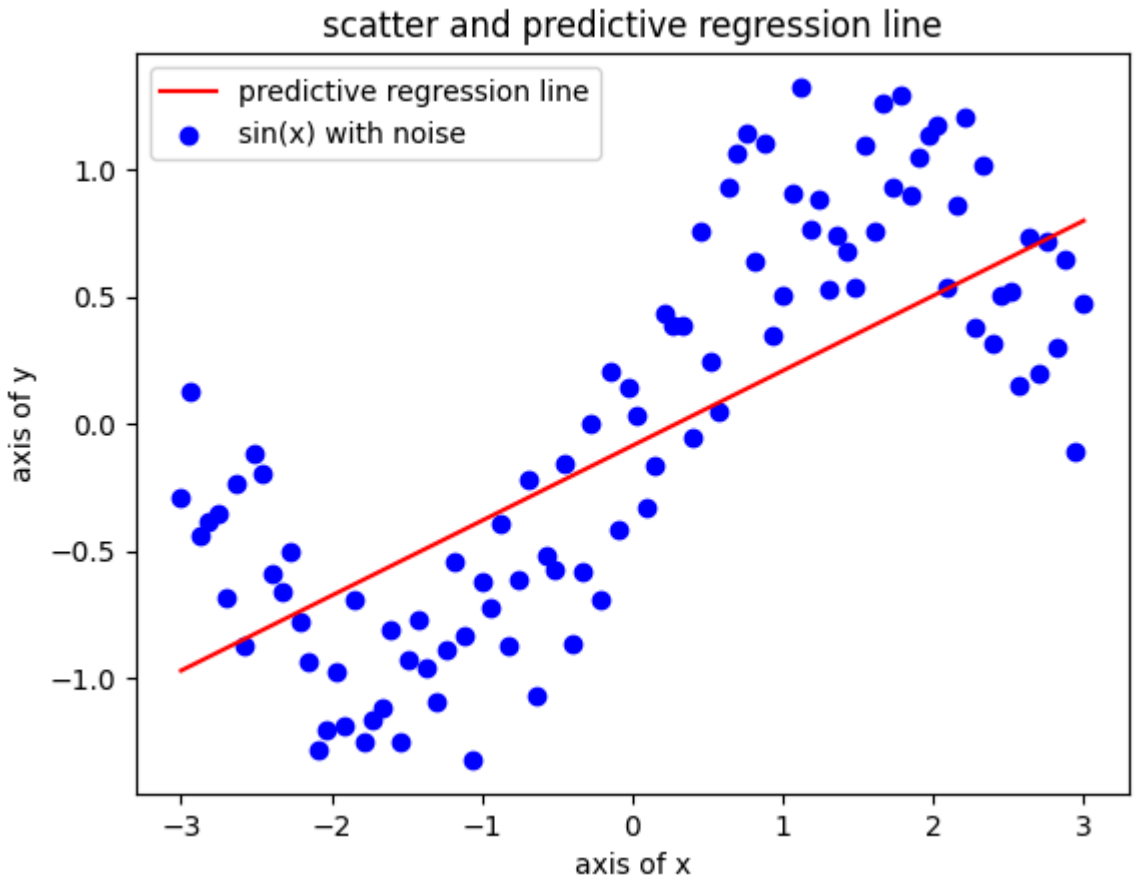
```
第193次迭代  参数w为:0.29244664311408997  参数b为:-0.6673749089241028  loss为:0.0985231027007103
第194次迭代  参数w为:0.29280269145965576  参数b为:-0.6670211553573608  loss为:0.0996415764093399
第195次迭代  参数w为:0.29315483570098877  参数b为:-0.6666709184646606  loss为:0.10075436532497406
第196次迭代  参数w为:0.2935031056404114  参数b为:-0.6663244366645813  loss为:0.10186123102903366
第197次迭代  参数w为:0.2938477098941803  参数b为:-0.665981113910675  loss为:0.10296297818422318
第198次迭代  参数w为:0.2941887378692627  参数b为:-0.6656416654586792  loss为:0.10405872762203217
第199次迭代  参数w为:0.2945256531238556  参数b为:-0.6653051376342773  loss为:0.10514844954013824
第200次迭代  参数w为:0.2948592007160187  参数b为:-0.664972186088562  loss为:0.1062326580286026
```

(4) 画出预测回归曲线以及训练数据散点图，对比回归曲线和散点图并分析原因。

主要思路是将model设置为test模式让其前向传播算出y的预测值，然后用pyplot画出拟合的直线即可。对比散点图和回归曲线可以发现线性回归只能拟合线性的关系，它无法很好地捕捉正弦函数这种明显的非线性模式。虽然线性回归不适合这种数据，但比较好地捕捉了数据的整体趋势，如果希望较好拟合仍需要用非线性模型进行拟合。

In [4]:
```
y_hat = []
for x in train_x:
    x = torch.tensor(x).float().cuda()
    y_hat.append(model(x).detach().cpu().item())

plt.plot(train_x,y_hat,label="predictive regression line",color='red')
plt.xlabel("axis of x")
plt.ylabel("axis of y")
plt.title("scatter and predictive regression line")
plt.scatter(train_x,train_y,label="sin(x) with noise",color='blue')
plt.legend()
plt.show()
```



2、线性回归（使用多项式函数对原始数据进行变换）

(1) 生成训练数据，数据同上

In [5]:
```
num_observations=100
train_x2=np.linspace(-3,3,num_observations)
train_y2=np.sin(train_x)+np.random.uniform(-0.5,0.5,num_observations)
```

(2)使用pytorch实现线性回归模型，这里我们假设y是x的 3 次多项式，那么 我们可以将数据扩展为：x、x^2、x^3三维数据，此时模型变为： $y = w1 * x + w2 * x^2 + w3 * x^3 + b$

在这一步先自定义非线性回归模型，重写前向传播函数，然后进行初始化模型，定义损失函数，选择优化器（如何反向传播）。

In [6]:
```
#定义一个模型
class PolynomialLinearModel(nn.Module):
    def __init__(self):
        super(PolynomialLinearModel,self).__init__()
        self.fc1=nn.Linear(4,1)

    def forward(self,x):
        #扩展输入，输入的矩阵是（M，N）M是样本数量，N是特征数量。
        x_expanded=torch.cat([x,x**2,x**3,torch.ones_like(x)],dim=1)
        output=self.fc1(x_expanded)
        return output

# 初始化模型、损失函数和优化器
model_Poly = PolynomialLinearModel().cuda()
criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=1e-6)
```

(3) 训练模型并输出参数$w1$、$w2$、$w3$、b和损失。（提交运行结果）

主要步骤如下： 1.定义超参数 2.将数据从CPU转到GPU 3.进行迭代 4.打印相关参数

In [25]:
```
num_epochs=500
num_observations=100
# 使数据形式符合输入形式
train_x2 = train_x2.reshape(num_observations, 1)
x_train_tensor = torch.tensor(train_x2).float().cuda()  # 移动到 GPU
train_y2 = train_y2.reshape(num_observations, 1)
y_train_tensor = torch.tensor(train_y2).float().cuda()  # 移动到 GPU

for epoch in range(num_epochs):
    model_Poly.train()
    optimizer.zero_grad()
    output = model_Poly(x_train_tensor)  # 确保 x_train_tensor 已经在 GPU 上
    loss = criterion(output, y_train_tensor)
    loss.backward()
    optimizer.step()

    # 获取权重和偏置
    w1, w2, w3, b = model_Poly.fc1.weight[0, 0].item(), model_Poly.fc1.weight[0, 1].
    if epoch%10==0:
        print(f'epoch:{epoch+1} 参数: w1 = {w1}, w2 = {w2}, w3 = {w3}, b = {b}, loss
```

epoch:1 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:11 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:21 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:31 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:41 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:51 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:61 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:71 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:81 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:91 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:101 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:111 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:121 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:131 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:141 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:151 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:161 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:171 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:181 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:191 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:201 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:211 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:221 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:231 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:241 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:251 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:261 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:271 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:281 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:291 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:301 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0
epoch:311 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436164856, b = 0.44015467166900635, loss=-0.0

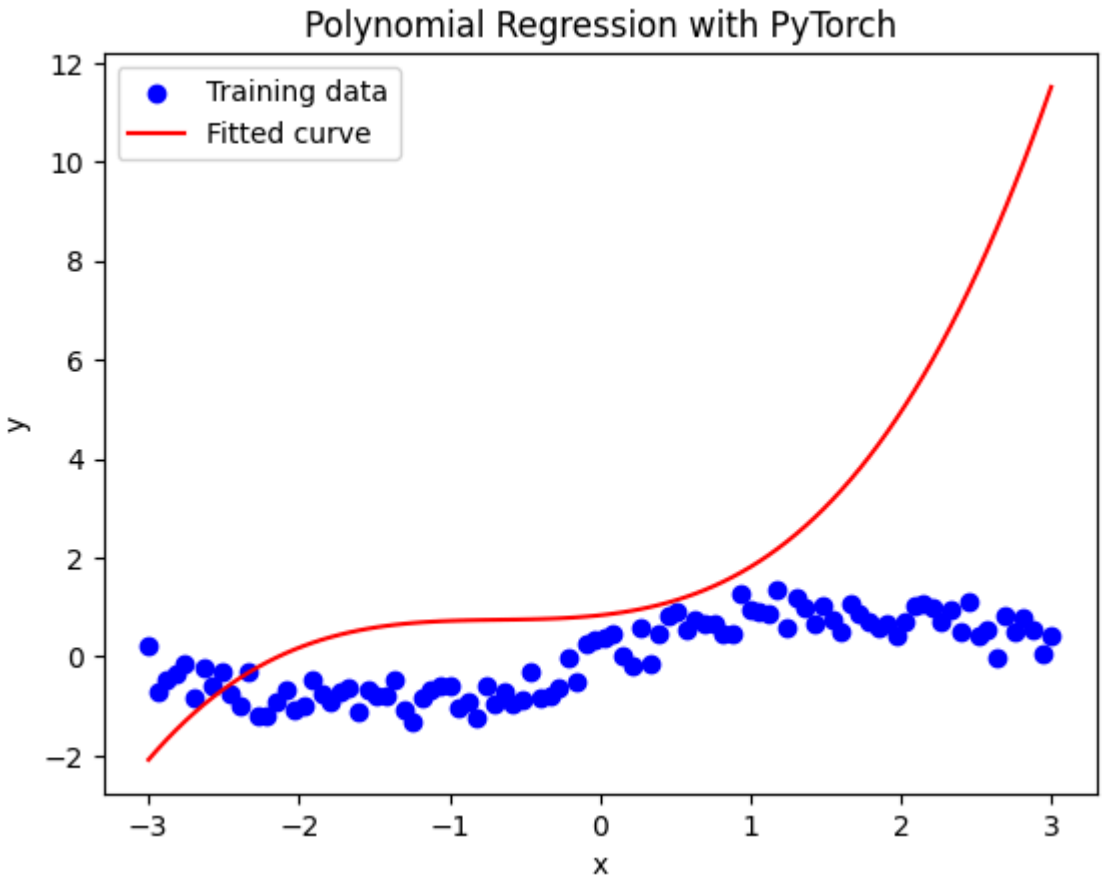
```
epoch:321 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
epoch:331 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
epoch:341 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
epoch:351 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
epoch:361 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
epoch:371 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
epoch:381 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
epoch:391 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
epoch:401 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
epoch:411 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
epoch:421 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
epoch:431 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
epoch:441 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
epoch:451 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
epoch:461 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
epoch:471 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
epoch:481 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
epoch:491 参数: w1 = 0.32896900177001953, w2 = 0.4309638738632202, w3 = 0.2151673436
164856, b = 0.44015467166900635, loss=-0.0
```

(4) 画出预测回归曲线以及训练数据散点图，对比回归曲线和散点图并分析原因。

In [26]:

```
model_Poly.eval()
with torch.no_grad():#关闭梯度计算,避免这种追踪，从而节省内存
    y_pred = model_Poly(x_train_tensor).cpu().numpy()

# 绘制预测回归曲线和训练数据散点图
plt.scatter(train_x2,train_y2, label='Training data', color='blue')
plt.plot(train_x2, y_pred, label='Fitted curve', color='red')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.title('Polynomial Regression with PyTorch')
plt.show()
```



观察和分析： 1.三次函数不能很好地拟合散点：分析后得出多项式的自由度有限，对于复杂且周期性的正弦函数数据，多项式的表达能力并不足以捕捉全部特征。 2.模型可能会有过拟合现象。多项式回归容易对噪声和局部小波动产生较高的敏感度。 3.可以用更高次的函数去拟合，三次函数无法很好地拟合。

3、Softmax 分类

(1) 获取 MNIST 数据集，每张图片像素为28 × 28

先下载好数据集到本地，分为训练数据集合测试数据集。

In [9]:
```
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms

# 1. 加载 MNIST 数据集
batch_size = 64
transform = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.5,),

train_dataset = datasets.MNIST(root='./data', train=True, transform=transform, down
test_dataset = datasets.MNIST(root='./data', train=False, transform=transform)

train_loader = torch.utils.data.DataLoader(dataset=train_dataset, batch_size=batch_s
test_loader = torch.utils.data.DataLoader(dataset=test_dataset, batch_size=batch_siz
```

(2) 模型架构为：$y = softmax(w * x + b)$,其中w的维度为784 × 10，b的维度为10。

主要编码流程为： 1.定义 Softmax 分类模型。 2.初始化模型、损失函数（交叉熵）和优化器（SGD）。 3.实现训练循环，包括前向传播、损失计算、反向传播和参数更新。 4.在每个

epoch 后进行模型评估，并在训练和测试集上验证模型性能。

In [10]:
```
# 2. 定义 Softmax 分类模型
class SoftmaxClassifier(nn.Module):
    def __init__(self):
        super(SoftmaxClassifier, self).__init__()
        self.fc = nn.Linear(28*28, 10)  # w: [784, 10], b: [10]

    def forward(self, x):
        x = x.view(-1, 28*28)  # Flatten the input image
        output = self.fc(x)
        return nn.functional.softmax(output, dim=1)  # Apply softmax to output

# 初始化模型、损失函数和优化器
model = SoftmaxClassifier().cuda()
criterion = nn.CrossEntropyLoss()  # CrossEntropyLoss combines softmax + NLLLoss
optimizer = optim.SGD(model.parameters(), lr=0.1)

# 训练模型
num_epochs = 50
train_losses = []
test_losses = []
train_accuracies = []
test_accuracies = []

def compute_accuracy(output, target):
    _, pred = torch.max(output, 1)
    correct = (pred == target).sum().item()
    return correct / target.size(0)

for epoch in range(num_epochs):
    model.train()
    total_train_loss = 0
    total_train_accuracy = 0

    # Training loop
    for images, labels in train_loader:
        images, labels = images.cuda(), labels.cuda()
        optimizer.zero_grad()
        output = model(images)
        loss = criterion(output, labels)
        loss.backward()
        optimizer.step()

        total_train_loss += loss.item()
        total_train_accuracy += compute_accuracy(output, labels)

    avg_train_loss = total_train_loss / len(train_loader)
    avg_train_accuracy = total_train_accuracy / len(train_loader)
    train_losses.append(avg_train_loss)
    train_accuracies.append(avg_train_accuracy)

    # 在测试集上验证
    model.eval()
    total_test_loss = 0
    total_test_accuracy = 0
    with torch.no_grad():
        for images, labels in test_loader:
            images, labels = images.cuda(), labels.cuda()
            output = model(images)
            loss = criterion(output, labels)
            total_test_loss += loss.item()
            total_test_accuracy += compute_accuracy(output, labels)
```

```
    avg_test_loss = total_test_loss / len(test_loader)
    avg_test_accuracy = total_test_accuracy / len(test_loader)
    test_losses.append(avg_test_loss)
    test_accuracies.append(avg_test_accuracy)

    print(f'Epoch [{epoch+1}/{num_epochs}], Train Loss: {avg_train_loss:.4f}, Train
```

Epoch [1/50], Train Loss: 1.7345, Train Acc: 0.7630, Test Loss: 1.6454, Test Acc: 0.8372
Epoch [2/50], Train Loss: 1.6458, Train Acc: 0.8324, Test Loss: 1.6312, Test Acc: 0.8436
Epoch [3/50], Train Loss: 1.6358, Train Acc: 0.8379, Test Loss: 1.6263, Test Acc: 0.8461
Epoch [4/50], Train Loss: 1.6030, Train Acc: 0.8755, Test Loss: 1.5689, Test Acc: 0.9102
Epoch [5/50], Train Loss: 1.5693, Train Acc: 0.9087, Test Loss: 1.5616, Test Acc: 0.9155
Epoch [6/50], Train Loss: 1.5633, Train Acc: 0.9127, Test Loss: 1.5576, Test Acc: 0.9176
Epoch [7/50], Train Loss: 1.5594, Train Acc: 0.9160, Test Loss: 1.5528, Test Acc: 0.9212
Epoch [8/50], Train Loss: 1.5568, Train Acc: 0.9168, Test Loss: 1.5530, Test Acc: 0.9189
Epoch [9/50], Train Loss: 1.5546, Train Acc: 0.9187, Test Loss: 1.5507, Test Acc: 0.9213
Epoch [10/50], Train Loss: 1.5528, Train Acc: 0.9196, Test Loss: 1.5513, Test Acc: 0.9200
Epoch [11/50], Train Loss: 1.5513, Train Acc: 0.9208, Test Loss: 1.5489, Test Acc: 0.9218
Epoch [12/50], Train Loss: 1.5500, Train Acc: 0.9218, Test Loss: 1.5461, Test Acc: 0.9245
Epoch [13/50], Train Loss: 1.5489, Train Acc: 0.9228, Test Loss: 1.5464, Test Acc: 0.9241
Epoch [14/50], Train Loss: 1.5479, Train Acc: 0.9233, Test Loss: 1.5454, Test Acc: 0.9244
Epoch [15/50], Train Loss: 1.5468, Train Acc: 0.9246, Test Loss: 1.5455, Test Acc: 0.9236
Epoch [16/50], Train Loss: 1.5461, Train Acc: 0.9247, Test Loss: 1.5458, Test Acc: 0.9233
Epoch [17/50], Train Loss: 1.5451, Train Acc: 0.9257, Test Loss: 1.5450, Test Acc: 0.9240
Epoch [18/50], Train Loss: 1.5445, Train Acc: 0.9262, Test Loss: 1.5437, Test Acc: 0.9244
Epoch [19/50], Train Loss: 1.5438, Train Acc: 0.9262, Test Loss: 1.5449, Test Acc: 0.9244
Epoch [20/50], Train Loss: 1.5432, Train Acc: 0.9270, Test Loss: 1.5438, Test Acc: 0.9246
Epoch [21/50], Train Loss: 1.5426, Train Acc: 0.9275, Test Loss: 1.5432, Test Acc: 0.9267
Epoch [22/50], Train Loss: 1.5420, Train Acc: 0.9283, Test Loss: 1.5426, Test Acc: 0.9269
Epoch [23/50], Train Loss: 1.5414, Train Acc: 0.9290, Test Loss: 1.5427, Test Acc: 0.9253
Epoch [24/50], Train Loss: 1.5409, Train Acc: 0.9297, Test Loss: 1.5413, Test Acc: 0.9267
Epoch [25/50], Train Loss: 1.5406, Train Acc: 0.9292, Test Loss: 1.5414, Test Acc: 0.9263
Epoch [26/50], Train Loss: 1.5402, Train Acc: 0.9293, Test Loss: 1.5411, Test Acc: 0.9257
Epoch [27/50], Train Loss: 1.5396, Train Acc: 0.9300, Test Loss: 1.5411, Test Acc: 0.9263
Epoch [28/50], Train Loss: 1.5391, Train Acc: 0.9302, Test Loss: 1.5402, Test Acc: 0.9273
Epoch [29/50], Train Loss: 1.5389, Train Acc: 0.9307, Test Loss: 1.5415, Test Acc: 0.9264
Epoch [30/50], Train Loss: 1.5385, Train Acc: 0.9308, Test Loss: 1.5401, Test Acc: 0.9260
Epoch [31/50], Train Loss: 1.5382, Train Acc: 0.9310, Test Loss: 1.5403, Test Acc: 0.9271
Epoch [32/50], Train Loss: 1.5377, Train Acc: 0.9314, Test Loss: 1.5401, Test Acc: 0.9276

```
Epoch [33/50], Train Loss: 1.5376, Train Acc: 0.9315, Test Loss: 1.5394, Test Acc:
0.9276
Epoch [34/50], Train Loss: 1.5371, Train Acc: 0.9318, Test Loss: 1.5394, Test Acc:
0.9276
Epoch [35/50], Train Loss: 1.5368, Train Acc: 0.9327, Test Loss: 1.5389, Test Acc:
0.9288
Epoch [36/50], Train Loss: 1.5367, Train Acc: 0.9319, Test Loss: 1.5387, Test Acc:
0.9272
Epoch [37/50], Train Loss: 1.5363, Train Acc: 0.9330, Test Loss: 1.5394, Test Acc:
0.9278
Epoch [38/50], Train Loss: 1.5360, Train Acc: 0.9330, Test Loss: 1.5389, Test Acc:
0.9281
Epoch [39/50], Train Loss: 1.5357, Train Acc: 0.9335, Test Loss: 1.5394, Test Acc:
0.9261
Epoch [40/50], Train Loss: 1.5355, Train Acc: 0.9334, Test Loss: 1.5388, Test Acc:
0.9283
Epoch [41/50], Train Loss: 1.5352, Train Acc: 0.9334, Test Loss: 1.5386, Test Acc:
0.9278
Epoch [42/50], Train Loss: 1.5350, Train Acc: 0.9338, Test Loss: 1.5381, Test Acc:
0.9277
Epoch [43/50], Train Loss: 1.5349, Train Acc: 0.9337, Test Loss: 1.5390, Test Acc:
0.9270
Epoch [44/50], Train Loss: 1.5345, Train Acc: 0.9342, Test Loss: 1.5387, Test Acc:
0.9273
Epoch [45/50], Train Loss: 1.5344, Train Acc: 0.9345, Test Loss: 1.5380, Test Acc:
0.9275
Epoch [46/50], Train Loss: 1.5342, Train Acc: 0.9341, Test Loss: 1.5391, Test Acc:
0.9264
Epoch [47/50], Train Loss: 1.5340, Train Acc: 0.9345, Test Loss: 1.5384, Test Acc:
0.9277
Epoch [48/50], Train Loss: 1.5337, Train Acc: 0.9350, Test Loss: 1.5376, Test Acc:
0.9283
Epoch [49/50], Train Loss: 1.5336, Train Acc: 0.9354, Test Loss: 1.5381, Test Acc:
0.9273
Epoch [50/50], Train Loss: 1.5334, Train Acc: 0.9349, Test Loss: 1.5374, Test Acc:
0.9282
```

(3) 画出训练和测试过程的准确率随迭代次数变化图，画出训练和测试过程的损 失随迭代次数变化图。（提交最终分类精度、分类损失以及两张变化图）

In [11]:

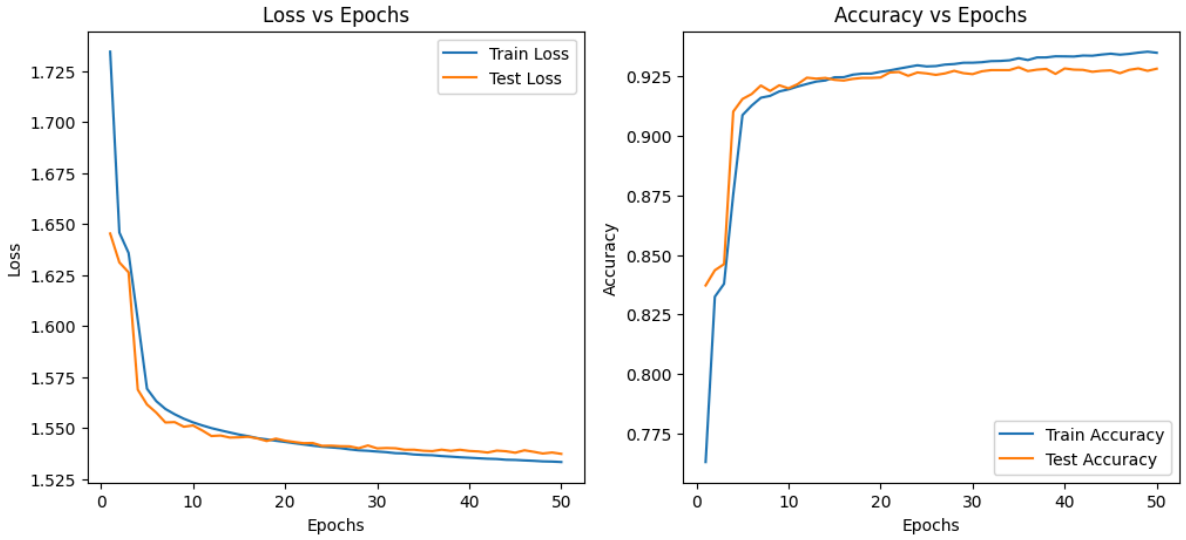
```python
import matplotlib.pyplot as plt

# 5. 绘制损失和准确率变化图
epochs = range(1, num_epochs + 1)

# 绘制损失图
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(epochs, train_losses, label='Train Loss')
plt.plot(epochs, test_losses, label='Test Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Loss vs Epochs')
plt.legend()

# 绘制准确率图
plt.subplot(1, 2, 2)
plt.plot(epochs, train_accuracies, label='Train Accuracy')
plt.plot(epochs, test_accuracies, label='Test Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Accuracy vs Epochs')
plt.legend()
```

```
plt.show()
```


Loss vs Epochs
Train Loss
Test Loss
Loss
Epochs
1.725
1.700
1.675
1.650
1.625
1.600
1.575
1.550
1.525
0
10
20
30
40
50
Accuracy vs Epochs
Train Accuracy
Test Accuracy
Accuracy
Epochs
0.925
0.900
0.875
0.850
0.825
0.800
0.775

In [ ]: