In [1]:
```
import sys; sys.path.append('../..') ; sys.path.append('..') ; from my_utils

import torch
import torch.nn as nn
import torch.utils.data as data
import torch.optim as optim
# dummy trainloader
trainloader = data.DataLoader(data.TensorDataset(torch.Tensor(1), torch.Tens
device = torch.device('cpu')

import matplotlib.pyplot as plt
```

In this homework, there are three different datasets consisting of 2-dimensional input features and binary class labels, and you will be asked to implement machine learning classifiers.

Let's begin by importing some libaries.

Next, we set a random seed for reproducibility.

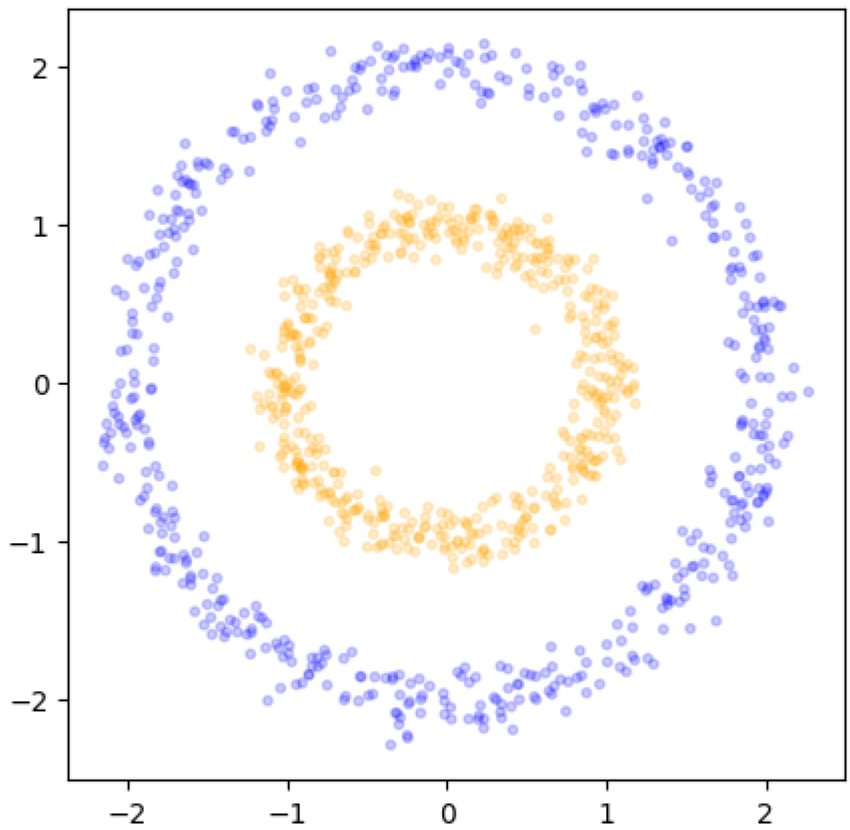
In [2]:
```
import numpy as np
import random

seed = 0
np.random.seed(seed)
torch.random.manual_seed(seed)
random.seed(seed)
```

Concentric annuli

In [3]:
```
X, y = sample_annuli()
fig, ax = plt.subplots(1,1, figsize=(5,5))
print(X.shape)
plot_scatter(ax, X, y)
```

```
torch.Size([1024, 2])
```



[2pt] Let's start by implmenting a logistic regression model. Fill the template below to complete the logisitc regression model. Use the binary cross entropy loss, torch.nn.BCELoss.

(i) Complete the model, (ii) finish the training loop, (iii) present the results with a figure (see the example below) and the classification accuracy

In [4]:
```
class Model(nn.Module):
    def __init__(self,device="cpu"):
        super(Model, self).__init__()
        self.linear = nn.Linear(2, 1)
        self.sigmoid = nn.Sigmoid()


    def forward(self, x):
        layer = self.linear(x)
        y = self.sigmoid(layer)
        return y
```

In [5]:
```
model = Model().to(device)
```

In [6]:
```
optimizer = optim.AdamW(model.parameters(), lr=1e-2, weight_decay=1e-6)
```

In [7]:
```
criterion = nn.BCELoss()

# complete the following training loop.
for itr in range(1, 1001):
```

Loading [MathJax]/extensions/Safe.js

```python
    optimizer.zero_grad()
    yh = model(X) # forward pass
    y = y.float().view(-1, 1)
    loss = criterion(yh, y)
    loss.backward()  # Backward pass
    optimizer.step()
```

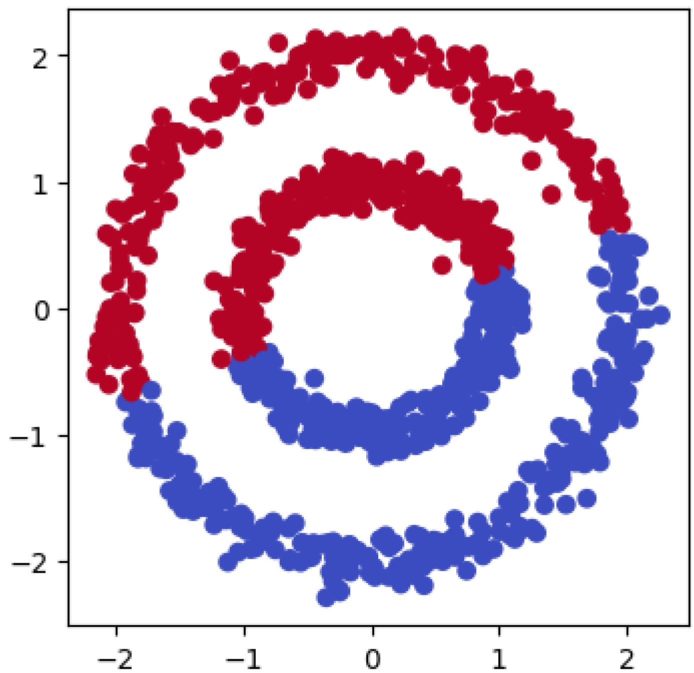
In [8]:
```python
# visualize the result and report the accuracy
with torch.no_grad():
    fig = plt.figure(figsize=(4,4))

    yh_test = model(X)  # X_test should contain your test data
    predicted_labels = (yh_test >= 0.5).float()  # Convert predicted probabi

    # Calculate accuracy
    print(predicted_labels, y)
    correct_predictions = (predicted_labels == y.float().view(-1, 1)).sum().
    total_samples = len(y)
    accuracy = correct_predictions / total_samples
    print(accuracy)

    # Visualization
    fig, ax = plt.subplots(figsize=(4, 4))
    ax.scatter(X.numpy()[:, 0], X.numpy()[:, 1], c=predicted_labels.numpy().
```

```
tensor([[0.],
        [1.],
        [0.],
        ...,
        [0.],
        [0.],
        [1.]]) tensor([[1.],
        [1.],
        [1.],
        ...,
        [0.],
        [0.],
        [0.]])
0.52734375
<Figure size 400x400 with 0 Axes>
```



It is obvious that the logistic regression would not be able to distinguish two classes (not linearly separate data). You will have to build another model.

[3pt] In the class template below, implement your own model that will achieve 100% accuracy in classifying the data poitns in training set. There is one restriction; you are allowed to use "one" linear layer for your implementation as in the logistic regression model above. But you are allowed to use as many nonlinear functions as needed.

In [18]:
```
class Model(nn.Module):
    def __init__(self,device="cpu"):
        super(Model, self).__init__()
        self.linear = nn.Linear(2,1)
        self.silu = nn.SiLU()
        self.bn = nn.BatchNorm1d(2)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        y = self.bn(x)
        y = self.silu(y)
        y = self.bn(y)
        y = self.linear(y)
        y = self.sigmoid(y)
        return y
```

In [19]:
```
model = Model().to(device)
```

In [20]:
```
optimizer = optim.AdamW(model.parameters(), lr=1e-2, weight_decay=1e-6)
criterion = nn.BCELoss()
```

In [21]:
```
for itr in range(1, 3001):
    optimizer.zero_grad()
    yh = model(X) # forward pass
    y = y.float().view(-1, 1)
    loss = criterion(yh, y)
    loss.backward()  # Backward pass
    optimizer.step()
```

In [22]:
```
print(loss.item())
```

```
0.05006854608654976
```

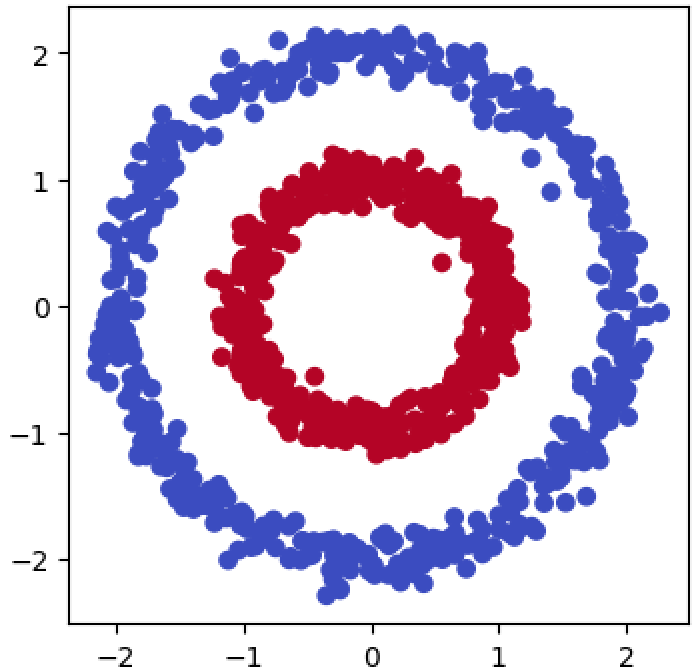
In [24]:
```
import tensorflow as tf
#print(y)
with torch.no_grad():
    fig = plt.figure(figsize=(4,4))

    yh_test = model(X)
    predicted_labels = (yh_test > 0.5).float()  # Convert predicted probabil

    # Calculate accuracy
    print(predicted_labels, y)
    correct_predictions = (predicted_labels == y.float().view(-1, 1)).sum().
    total_samples = len(y)
    accuracy = correct_predictions / total_samples
    print(accuracy)

    # Visualization
    fig, ax = plt.subplots(figsize=(4, 4))
    ax.scatter(X.numpy()[:, 0], X.numpy()[:, 1], c=predicted_labels.numpy().
```

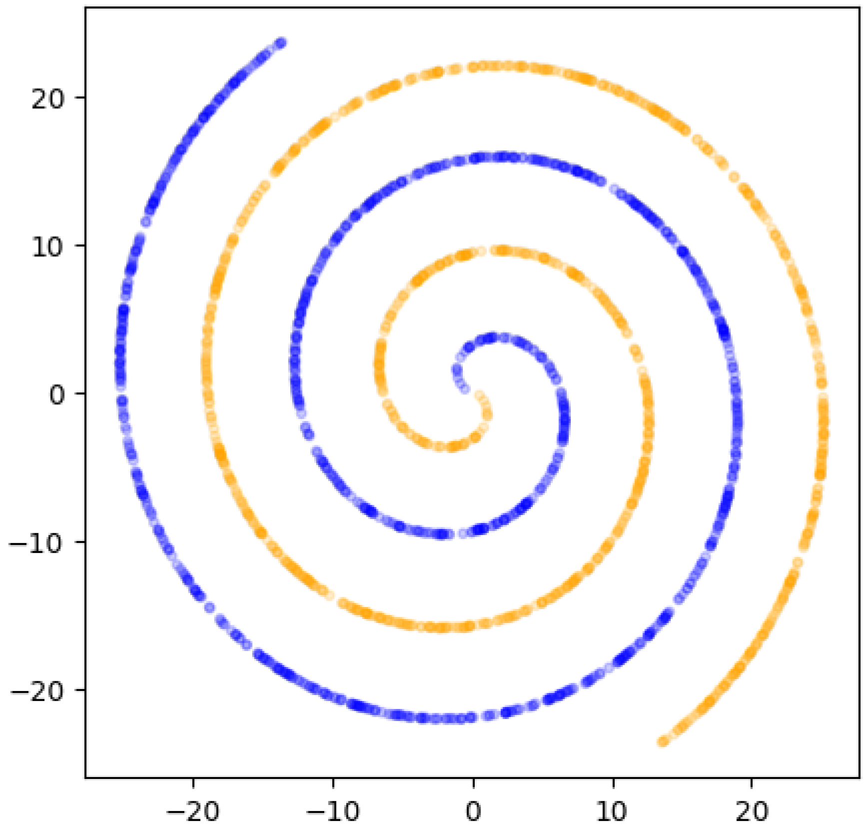
```
tensor([[1.],
        [1.],
        [1.],
        ...,
        [0.],
        [0.],
        [0.]]) tensor([[1.],
        [1.],
        [1.],
        ...,
        [0.],
        [0.],
        [0.]])
1.0
<Figure size 400x400 with 0 Axes>
```



Spiral dataset

In [15]:
```
X, y = sample_spiral()
print(X.shape)
print(y)
fig, ax = plt.subplots(1,1, figsize=(5,5))
plot_scatter(ax, X, y)
```

```
torch.Size([2048, 2])
tensor([0, 0, 0,  ..., 1, 1, 1])
```



It's obvious that neither the logistic regression nor the model you developed for the second dataset would not work for this dataset.

[2pt] implemented a neural network of your choice and achieve 100% classification accuracy

In [16]:
```
class Model(nn.Module):
    def __init__(self,device="cpu"):
        super(Model, self).__init__()

        self.net = nn.Sequential(
            nn.Linear(2,50),
            nn.Tanh(),
            nn.Linear(50,50),
            nn.Tanh(),
            nn.Linear(50,1))
        #for p in self.linear_layer.parameters(): torch.nn.init.zeros_(p)

    def forward(self, x):
        y = self.net(x)
        y = torch.sigmoid(y)
        return y
```

In [17]:
```
model = Model().to(device)
```

In [18]:
```
optimizer = optim.AdamW(model.parameters(), lr=1e-2, weight_decay=1e-6)
```

In [19]:
```
for itr in range(1, 1001):
    optimizer.zero_grad()
#    x0, y = sample_annuli(device=device, n_samples=1024)

    yh = model(X)
    loss = nn.BCELoss()(yh.squeeze(), y.float())
    print(itr,loss.item())
    #loss_traj[itr-1] = loss.item()
    loss.backward()
    optimizer.step()
```

1 0.686331570148468
2 0.7187483906745911
3 0.6712204813957214
4 0.6821600794792175
5 0.6861546039581299
6 0.6722385883331299
7 0.6652229428291321
8 0.6684250235557556
9 0.671859860420227
10 0.6701979637145996
11 0.6657152771949768
12 0.6624469757080078
13 0.6621698141098022
14 0.6636547446250916
15 0.6645035147666931
16 0.6634788513183594
17 0.6611378192901611
18 0.6589213609695435
19 0.6578884124755859
20 0.657916784286499
21 0.6578779816627502
22 0.656735897064209
23 0.6545669436454773
24 0.652484655380249
25 0.6514768600463867
26 0.6511526703834534
27 0.6500186324119568
28 0.6476423144340515
29 0.6454055309295654
30 0.6442955136299133
31 0.643032968044281
32 0.6404210925102234
33 0.6379827857017517
34 0.6368457078933716
35 0.6345685124397278
36 0.6316235065460205
37 0.6301279664039612
38 0.6273697018623352
39 0.6249841451644897
40 0.6232141852378845
41 0.6201590895652771
42 0.618785560131073
43 0.6163374781608582
44 0.6154015064239502
45 0.6133973598480225
46 0.6116297841072083
47 0.6104554533958435
48 0.6090543270111084
49 0.610548734664917
50 0.610389769077301
51 0.6044207811355591
52 0.6061080694198608
53 0.6059073209762573
54 0.600985050201416
55 0.603423535823822
56 0.601421131362915

```
57 0.5983671545982361
58 0.6005373597145081
59 0.5975260138511658
60 0.5956800580024719
61 0.5974017381668091
62 0.5945259928703308
63 0.5935441851615906
64 0.5963867902755737
65 0.5984927415847778
66 0.5951334238052368
67 0.5931817889213562
68 0.5928484797477722
69 0.5912097096443176
70 0.5917071104049683
71 0.5894709825515747
72 0.588786244392395
73 0.5897322297096252
74 0.5864490866661072
75 0.5873163342475891
76 0.5870151519775391
77 0.5841821432113647
78 0.585780143737793
79 0.5842284560203552
80 0.5826447606086731
81 0.5838415622711182
82 0.581992506980896
83 0.581304669380188
84 0.5816841125488281
85 0.5802556276321411
86 0.5798788666725159
87 0.5794996023178101
88 0.5784872770309448
89 0.5783880352973938
90 0.5776345133781433
91 0.5765941739082336
92 0.5766661167144775
93 0.5761460661888123
94 0.575021505355835
95 0.5747208595275879
96 0.5743622779846191
97 0.5734759569168091
98 0.5730525255203247
99 0.5727274417877197
100 0.5718552470207214
101 0.5710753202438354
102 0.5706718564033508
103 0.5700648427009583
104 0.5692545771598816
105 0.5686742067337036
106 0.5682392716407776
107 0.5676270723342896
108 0.5669965147972107
109 0.5667231678962708
110 0.5668889284133911
111 0.5677401423454285
112 0.5692542195320129
```

```
113 0.5711880922317505
114 0.5678861737251282
115 0.5629802942276001
116 0.5616155862808228
117 0.5636441111564636
118 0.5635136961936951
119 0.5598823428153992
120 0.5587342381477356
121 0.559718668460846
122 0.5584359169006348
123 0.5560513734817505
124 0.5554752945899963
125 0.5554963946342468
126 0.5538300275802612
127 0.5520559549331665
128 0.5518475770950317
129 0.5510408878326416
130 0.5490224361419678
131 0.5480377674102783
132 0.5474992394447327
133 0.545799970626831
134 0.544193685054779
135 0.5433371067047119
136 0.5420005321502686
137 0.5402101874351501
138 0.5387676954269409
139 0.5374072194099426
140 0.5357283353805542
141 0.5338956713676453
142 0.5320900678634644
143 0.5303150415420532
144 0.5284233093261719
145 0.5262577533721924
146 0.5239927172660828
147 0.5218395590782166
148 0.5195722579956055
149 0.5169869065284729
150 0.5142572522163391
151 0.5115010738372803
152 0.5086411237716675
153 0.5056253671646118
154 0.5024505257606506
155 0.4990848898887634
156 0.49552080035209656
157 0.4918251633644104
158 0.488020122051239
159 0.4840660095214844
160 0.4799480736255646
161 0.4756740927696228
162 0.471274733543396
163 0.46679550409317017
164 0.46237725019454956
165 0.4585846960544586
166 0.45765697956085205
167 0.46276023983955383
168 0.47595569491386414
```

169 0.44803595542907715
170 0.435818076133728
171 0.44560661911964417
172 0.43046486377716064
173 0.4230736494064331
174 0.4266189932823181
175 0.41389000415802
176 0.4117535352706909
177 0.41056498885154724
178 0.39958494901657104
179 0.4004629850387573
180 0.3951207399368286
181 0.38711920380592346
182 0.38788720965385437
183 0.3797956109046936
184 0.3755916655063629
185 0.3737361431121826
186 0.36537501215934753
187 0.3636234998703003
188 0.3588811159133911
189 0.3525477349758148
190 0.35055822134017944
191 0.3443615734577179
192 0.34023725986480713
193 0.3370012640953064
194 0.33093371987342834
195 0.3277689218521118
196 0.3236619532108307
197 0.3188432455062866
198 0.3173898756504059
199 0.31758448481559753
200 0.3189599812030792
201 0.31959885358810425
202 0.30364546179771423
203 0.2923550307750702
204 0.29244521260261536
205 0.2924617826938629
206 0.2863767147064209
207 0.27463042736053467
208 0.2765117585659027
209 0.2758740186691284
210 0.2647605240345001
211 0.2613036632537842
212 0.2617607116699219
213 0.2557279169559479
214 0.2469063699245453
215 0.24817976355552673
216 0.24411113560199738
217 0.23705001175403595
218 0.23363839089870453
219 0.23278287053108215
220 0.22820721566677094
221 0.2218119204044342
222 0.221212700009346
223 0.21862590312957764
224 0.21358713507652283

225	0.20882336795330048
226	0.20856459438800812
227	0.20486803352832794
228	0.20041769742965698
229	0.19712550938129425
230	0.19610688090324402
231	0.19272872805595398
232	0.1884710043668747
233	0.18596197664737701
234	0.18442995846271515
235	0.1810649335384369
236	0.17724041640758514
237	0.17511829733848572
238	0.17314547300338745
239	0.17015960812568665
240	0.16666868329048157
241	0.16458730399608612
242	0.16244591772556305
243	0.15949074923992157
244	0.15609800815582275
245	0.1537482887506485
246	0.1513235867023468
247	0.14843711256980896
248	0.14501217007637024
249	0.14237433671951294
250	0.13981837034225464
251	0.1369674950838089
252	0.13350646197795868
253	0.13048630952835083
254	0.1275634616613388
255	0.12455550581216812
256	0.12114935368299484
257	0.11813998222351074
258	0.11530288308858871
259	0.11251887679100037
260	0.10944855958223343
261	0.10654760897159576
262	0.10378184914588928
263	0.10114625841379166
264	0.09831950813531876
265	0.09541311115026474
266	0.0925297886133194
267	0.08979246765375137
268	0.08714412152767181
269	0.08448575437068939
270	0.08184422552585602
271	0.07926013320684433
272	0.07683107256889343
273	0.07445966452360153
274	0.07215417176485062
275	0.06992983818054199
276	0.0678582563996315
277	0.06583121418952942
278	0.06382083147764206
279	0.06183823198080063
280	0.059945594519376755

281 0.05808888375759125
282 0.056287314742803574
283 0.054565995931625366
284 0.052927371114492416
285 0.05131681263446808
286 0.04976304620504379
287 0.048257533460855484
288 0.04678323492407799
289 0.0453374981880188
290 0.04394678398966789
291 0.04258756339550018
292 0.04126405343413353
293 0.03998356685042381
294 0.03873506933450699
295 0.03752433508634567
296 0.03636776655912399
297 0.035262688994407654
298 0.0342131070792675
299 0.033217817544937134
300 0.03225824609398842
301 0.03133504092693329
302 0.030437439680099487
303 0.029565561562776566
304 0.028724607080221176
305 0.027913561090826988
306 0.02714136429131031
307 0.0264054574072361
308 0.025706274434924126
309 0.025039050728082657
310 0.02439667098224163
311 0.0237770676612854
312 0.023175477981567383
313 0.022594155743718147
314 0.022032607346773148
315 0.021493174135684967
316 0.020975779742002487
317 0.02047969214618206
318 0.0200030654668808
319 0.019543834030628204
320 0.019100140780210495
321 0.01867058500647545
322 0.01825486309826374
323 0.017852336168289185
324 0.01746336929500103
325 0.017087377607822418
326 0.016724267974495888
327 0.016373267397284508
328 0.016033826395869255
329 0.01570495404303074
330 0.015386177226901054
331 0.015076704323291779
332 0.014776241034269333
333 0.014484405517578125
334 0.014200989156961441
335 0.01392573956400156
336 0.013658428564667702

337 0.013398761861026287
338 0.013146435841917992
339 0.012901260517537594
340 0.01266296673566103
341 0.01243146788328886
342 0.012206570245325565
343 0.011988199315965176
344 0.011776150204241276
345 0.011570297181606293
346 0.01137040089815855
347 0.011176259256899357
348 0.010987639427185059
349 0.010804316028952599
350 0.010626080445945263
351 0.010452723130583763
352 0.01028407271951437
353 0.01011993270367384
354 0.009960155934095383
355 0.00980456918478012
356 0.009653022512793541
357 0.009505368769168854
358 0.009361459873616695
359 0.009221164509654045
360 0.009084340184926987
361 0.008950862102210522
362 0.008820602670311928
363 0.00869343988597393
364 0.008569256402552128
365 0.008447941392660141
366 0.008329387754201889
367 0.008213498629629612
368 0.008100179955363274
369 0.007989345118403435
370 0.007880918681621552
371 0.007774824276566505
372 0.007670997641980648
373 0.007569379638880491
374 0.007469904609024525
375 0.007372518535703421
376 0.007277166470885277
377 0.007183795794844627
378 0.007092351093888283
379 0.007002776954323053
380 0.006915022153407335
381 0.006829032674431801
382 0.00674476008862257
383 0.0066621555015444756
384 0.006581174675375223
385 0.006501774303615093
386 0.006423915270715952
387 0.006347562186419964
388 0.006272681523114443
389 0.006199241615831852
390 0.006127209402620792
391 0.006056561134755611
392 0.005987268406897783

393 0.005919307470321655
394 0.005852647125720978
395 0.005787266418337822
396 0.005723136011511087
397 0.005660233087837696
398 0.005598528776317835
399 0.0055379983969032764
400 0.00547861447557807
401 0.00542034674435854
402 0.005363165866583586
403 0.005307046230882406
404 0.005251958966255188
405 0.0051978714764118195
406 0.005144758615642786
407 0.0050925882533192635
408 0.005041338503360748
409 0.004990979563444853
410 0.00494148675352335
411 0.0048928335309028625
412 0.004844998475164175
413 0.004797955509275198
414 0.004751686938107014
415 0.004706171806901693
416 0.0046613882295787334
417 0.004617316648364067
418 0.004573941696435213
419 0.0045312438160181046
420 0.004489204380661249
421 0.004447811283171177
422 0.004407044034451246
423 0.004366891458630562
424 0.0043273367919027805
425 0.004288364667445421
426 0.004249964840710163
427 0.0042121196165680885
428 0.004174817353487015
429 0.004138047341257334
430 0.004101796541363001
431 0.0040660519152879715
432 0.0040308027528226376
433 0.003996037412434816
434 0.003961748909205198
435 0.003927924204617739
436 0.0038945511914789677
437 0.0038616275414824486
438 0.0038291369564831257
439 0.0037970738485455513
440 0.0037654293701052666
441 0.0037341956049203873
442 0.00370336277410388
443 0.0036729255225509405
444 0.0036428761668503284
445 0.0036132042296230793
446 0.0035839076153934
447 0.003554974915459752
448 0.003526402171701193

449 0.003498181700706482
450 0.0034703088458627462
451 0.0034427752252668142
452 0.0034155789762735367
453 0.0033887079916894436
454 0.0033621604088693857
455 0.003335932269692421
456 0.0033100140281021595
457 0.0032844021916389465
458 0.0032590925693511963
459 0.003234079573303461
460 0.003209359012544155
461 0.0031849246006458998
462 0.0031607705168426037
463 0.0031368951313197613
464 0.003113294020295143
465 0.003089959267526865
466 0.0030668878462165594
467 0.00304407742805779
468 0.003021521959453821
469 0.0029992188792675734
470 0.0029771612025797367
471 0.002955346368253231
472 0.002933771815150976
473 0.0029124324209988117
474 0.0028913263231515884
475 0.0028704472351819277
476 0.0028497911989688873
477 0.002829359844326973
478 0.002809145487844944
479 0.002789146499708295
480 0.002769360551610589
481 0.0027497834526002407
482 0.0027304133400321007
483 0.0027112485840916634
484 0.0026922840625047684
485 0.0026735207065939903
486 0.002654955256730318
487 0.0026365863159298897
488 0.0026184101589024067
489 0.002600425388664007
490 0.0025826296769082546
491 0.0025650253519415855
492 0.0025476058945059776
493 0.0025303736329078674
494 0.002513324376195669
495 0.0024964548647403717
496 0.0024797688238322735
497 0.002463260432705283
498 0.002446927595883608
499 0.002430771943181753
500 0.0024147892836481333
501 0.0023989796172827482
502 0.0023833380546420813
503 0.002367864828556776
504 0.002352558309212327

505 0.0023374129086732864
506 0.0023224304895848036
507 0.002307608723640442
508 0.002292945282533765
509 0.0022784338798373938
510 0.0022640784736722708
511 0.002249873708933592
512 0.0022358165588229895
513 0.0022219079546630383
514 0.002208143239840865
515 0.002194523112848401
516 0.0021810419857501984
517 0.0021677003242075443
518 0.002154496032744646
519 0.0021414251532405615
520 0.0021284883841872215
521 0.002115682465955615
522 0.002103004837408662
523 0.002090456895530224
524 0.002078032586723566
525 0.0020657333079725504
526 0.0020535560324788094
527 0.0020414988975971937
528 0.0020295591093599796
529 0.0020177378319203854
530 0.0020060325041413307
531 0.0019944391679018736
532 0.0019829580560326576
533 0.001971587771549821
534 0.001960325287654996
535 0.0019491735147312284
536 0.001938123838044703
537 0.0019271794008091092
538 0.0019163392717018723
539 0.0019055979792028666
540 0.0018949606455862522
541 0.001884418772533536
542 0.001873976900242269
543 0.0018636290915310383
544 0.0018533761613070965
545 0.0018432178767398
546 0.001833151327446103
547 0.0018231769790872931
548 0.001813290175050497
549 0.0018034944077953696
550 0.001793785602785647
551 0.0017841620137915015
552 0.0017746243393048644
553 0.0017651707166805863
554 0.0017557998653501272
555 0.001746511086821556
556 0.0017373040318489075
557 0.0017281752079725266
558 0.0017191260121762753
559 0.001710155513137579
560 0.0017012609168887138

561 0.0016924423398450017
562 0.0016836990835145116
563 0.0016750289360061288
564 0.0016664337599650025
565 0.001657908782362938
566 0.0016494567971676588
567 0.0016410753596574068
568 0.00163276307284832
569 0.0016245185397565365
570 0.0016163422260433435
571 0.001608233549632132
572 0.0016001909971237183
573 0.001592213986441493
574 0.0015843021683394909
575 0.0015764539130032063
576 0.0015686685219407082
577 0.001560946344397962
578 0.0015532851684838533
579 0.0015456853434443474
580 0.0015381451230496168
581 0.0015306663699448109
582 0.001523245475254953
583 0.0015158825553953648
584 0.001508577959612012
585 0.0015013281954452395
586 0.0014941365225240588
587 0.001487002125941217
588 0.0014799196505919099
589 0.0014728937530890107
590 0.0014659196604043245
591 0.0014590012142434716
592 0.0014521328266710043
593 0.0014453165931627154
594 0.001438553910702467
595 0.0014318402390927076
596 0.0014251777902245522
597 0.0014185644686222076
598 0.0014119995757937431
599 0.0014054854400455952
600 0.001399017870426178
601 0.001392599195241928
602 0.0013862267369404435
603 0.0013799004955217242
604 0.0013736207038164139
605 0.0013673873618245125
606 0.0013611993053928018
607 0.001355054322630167
608 0.0013489543925970793
609 0.0013428995152935386
610 0.0013368858490139246
611 0.001330917002633214
612 0.0013249899493530393
613 0.001319102942943573
614 0.0013132605236023664
615 0.001307456404902041
616 0.0013016941957175732

617 0.0012959727318957448
618 0.0012902908492833376
619 0.0012846485478803515
620 0.0012790453620254993
621 0.0012734810588881373
622 0.0012679528445005417
623 0.001262466423213482
624 0.0012570128310471773
625 0.001251598703674972
626 0.0012462228769436479
627 0.0012408801121637225
628 0.00123557576444
629 0.0012303078547120094
630 0.0012250728905200958
631 0.0012198751792311668
632 0.0012147114612162113
633 0.001209581270813942
634 0.0012044851901009679
635 0.0011994217056781054
636 0.001194393029436469
637 0.0011893983464688063
638 0.0011844345135614276
639 0.0011795032769441605
640 0.0011746061500161886
641 0.0011697394074872136
642 0.00116490398067981
643 0.0011601010337471962
644 0.0011553275398910046
645 0.0011505853617563844
646 0.0011458719382062554
647 0.0011411906452849507
648 0.0011365377577021718
649 0.0011319146724417806
650 0.0011273209238424897
651 0.0011227559298276901
652 0.0011182205053046346
653 0.0011137123219668865
654 0.0011092325439676642
655 0.0011047801235690713
656 0.0011003577383235097
657 0.0010959610808640718
658 0.0010915916645899415
659 0.0010872490238398314
660 0.0010829332750290632
661 0.0010786427883431315
662 0.001074380474165082
663 0.0010701430728659034
664 0.001065931748598814
665 0.0010617449879646301
666 0.0010575851192697883
667 0.001053449115715921
668 0.0010493365116417408
669 0.001045250566676259
670 0.0010411911644041538
671 0.0010371519019827247
672 0.0010331387165933847

673 0.0010291484650224447
674 0.0010251845233142376
675 0.0010212399065494537
676 0.0010173199698328972
677 0.0010134231997653842
678 0.0010095498291775584
679 0.001005699159577489
680 0.0010018697939813137
681 0.0009980619652196765
682 0.000994277885183692
683 0.0009905148763209581
684 0.0009867732878774405
685 0.0009830538183450699
686 0.0009793546050786972
687 0.0009756791405379772
688 0.0009720223024487495
689 0.0009683865355327725
690 0.0009647714323364198
691 0.0009611774003133178
692 0.0009576036827638745
693 0.0009540487080812454
694 0.0009505160851404071
695 0.0009470003424212337
696 0.0009435055544599891
697 0.0009400293929502368
698 0.0009365741861984134
699 0.000933137140236795
700 0.0009297197684645653
701 0.0009263200918212533
702 0.0009229405550286174
703 0.0009195791208185256
704 0.0009162358473986387
705 0.000912911375053227
706 0.000909603841137141
707 0.0009063163306564093
708 0.0009030450019054115
709 0.0008997921831905842
710 0.0008965561282821
711 0.0008933393401093781
712 0.00089013681281358
713 0.0008869547164067626
714 0.0008837876957841218
715 0.0008806370897218585
716 0.0008775062160566449
717 0.0008743891958147287
718 0.0008712901035323739
719 0.0008682089974172413
720 0.0008651412790641189
721 0.0008620910230092704
722 0.0008590580546297133
723 0.0008560391725040972
724 0.0008530366467311978
725 0.0008500517578795552
726 0.00084708072245121
727 0.0008441247628070414
728 0.000841184810269624

729 0.0008382608066312969
730 0.00083535211160779
731 0.0008324570371769369
732 0.0008295790757983923
733 0.0008267141529358923
734 0.0008238658774644136
735 0.0008210301166400313
736 0.0008182104793377221
737 0.0008154046954587102
738 0.0008126129978336394
739 0.0008098373073153198
740 0.0008070762851275504
741 0.0008043276611715555
742 0.0008015910862013698
743 0.0007988719153217971
744 0.00079616520088166
745 0.0007934737368486822
746 0.0007907961262390018
747 0.0007881298079155385
748 0.0007854783907532692
749 0.0007828391971997917
750 0.0007802151958458126
751 0.0007776033598929644
752 0.0007750049117021263
753 0.0007724189199507236
754 0.000769846432376653
755 0.0007672879728488624
756 0.0007647404563613236
757 0.0007622063276357949
758 0.000759685761295259
759 0.0007571766618639231
760 0.0007546815904788673
761 0.0007521981606259942
762 0.0007497263723053038
763 0.0007472668658010662
764 0.0007448201067745686
765 0.0007423864444717765
766 0.0007399623864330351
767 0.0007375526474788785
768 0.0007351537933573127
769 0.0007327680941671133
770 0.0007303932216018438
771 0.0007280294084921479
772 0.0007256775861606002
773 0.0007233371143229306
774 0.000721008749678731
775 0.0007186905131675303
776 0.0007163851405493915
777 0.0007140903035178781
778 0.0007118071080185473
779 0.0007095357868820429
780 0.0007072743028402328
781 0.0007050245767459273
782 0.0007027852698229253
783 0.0007005555089563131
784 0.0006983369239605963

785 0.0006961305625736713
786 0.0006939339218661189
787 0.0006917479913681746
788 0.000689572945702821
789 0.0006874089012853801
790 0.0006852538790553808
791 0.0006831104401499033
792 0.0006809759070165455
793 0.0006788518512621522
794 0.0006767383893020451
795 0.0006746342987753451
796 0.0006725414423272014
797 0.0006704573752358556
798 0.000668383901938796
799 0.0006663206149823964
800 0.0006642660591751337
801 0.0006622213404625654
802 0.0006601864006370306
803 0.0006581624038517475
804 0.0006561462651006877
805 0.0006541404291056097
806 0.0006521425675600767
807 0.0006501557072624564
808 0.0006481768796220422
809 0.0006462088786065578
810 0.0006442476296797395
811 0.0006422967999242246
812 0.0006403560983017087
813 0.000638422672636807
814 0.0006364992586895823
815 0.0006345835281535983
816 0.0006326767615973949
817 0.0006307802395895123
818 0.000628891633823514
819 0.0006270110607147217
820 0.0006251403247006238
821 0.0006232773885130882
822 0.0006214226013980806
823 0.0006195770693011582
824 0.0006177397444844246
825 0.0006159100448712707
826 0.000614089542068541
827 0.0006122763734310865
828 0.0006104728672653437
829 0.0006086758221499622
830 0.0006068886141292751
831 0.0006051079835742712
832 0.0006033364334143698
833 0.0006015718099661171
834 0.000599816266912967
835 0.0005980685818940401
836 0.0005963270668871701
837 0.0005945944576524198
838 0.0005928684258833528
839 0.0005911524640396237
840 0.0005894425557926297

841 0.0005877402727492154
842 0.0005860449164174497
843 0.0005843568942509592
844 0.000582677370402962
845 0.0005810049478895962
846 0.0005793407326564193
847 0.0005776837933808565
848 0.0005760312778875232
849 0.0005743891233578324
850 0.0005727539537474513
851 0.0005711251287721097
852 0.0005695032887160778
853 0.0005678876768797636
854 0.0005662795156240463
855 0.0005646784557029605
856 0.0005630850791931152
857 0.0005614981637336314
858 0.0005599184660241008
859 0.0005583449965342879
860 0.0005567788612097502
861 0.0005552187212742865
862 0.0005536661483347416
863 0.0005521185230463743
864 0.0005505782319232821
865 0.0005490457988344133
866 0.0005475191865116358
867 0.0005459993844851851
868 0.0005444859270937741
869 0.0005429786397144198
870 0.0005414775805547833
871 0.00053998245857656
872 0.0005384943797253072
873 0.0005370121216401458
874 0.0005355364410206676
875 0.0005340669886209071
876 0.0005326040554791689
877 0.0005311472341418266
878 0.0005296967574395239
879 0.0005282511119730771
880 0.0005268118111416698
881 0.0005253797280602157
882 0.0005239525926299393
883 0.0005225312779657543
884 0.0005211164243519306
885 0.0005197069840505719
886 0.0005183040047995746
887 0.0005169049836695194
888 0.0005155144026502967
889 0.0005141283618286252
890 0.0005127467447891831
891 0.0005113726947456598
892 0.0005100046400912106
893 0.0005086408345960081
894 0.0005072815692983568
895 0.0005059307441115379
896 0.000504583993460983

897 0.0005032426561228931
898 0.00050190812908113
899 0.0005005766288377345
900 0.0004992514732293785
901 0.0004979329532943666
902 0.0004966194392181933
903 0.0004953098250553012
904 0.0004940067301504314
905 0.0004927075351588428
906 0.000491414568386972
907 0.0004901269567199051
908 0.000488843594212085
909 0.00048756611067801714
910 0.0004862935747951269
911 0.0004850257246289402
912 0.00048376372433267534
913 0.0004825061478186399
914 0.00048125386820174754
915 0.00048000714741647243
916 0.00047876511234790087
917 0.0004775275010615587
918 0.00047629541950300336
919 0.00047506787814199924
920 0.00047384429490193725
921 0.0004726261831820011
922 0.0004714153765235096
923 0.00047020570491440594
924 0.00046900182496756315
925 0.0004678026889450848
926 0.00046660995576530695
927 0.00046541952178813517
928 0.00046423578169196844
929 0.00046305457362905145
930 0.00046187909902073443
931 0.0004607092996593565
932 0.00045954258530400693
933 0.00045838006190024316
934 0.0004572233301587403
935 0.00045606851927004755
936 0.0004549217992462218
937 0.000453776417998597
938 0.00045263738138601184
939 0.0004515016917139292
940 0.00045037100790068507
941 0.00044924390385858715
942 0.00044812163105234504
943 0.00044700363650918007
944 0.0004458898038137704
945 0.0004447802493814379
946 0.00044367462396621704
947 0.00044257345143705606
948 0.0004414769646245986
949 0.0004403847560752183
950 0.0004392961272969842
951 0.00043821174767799675
952 0.00043713246122933924

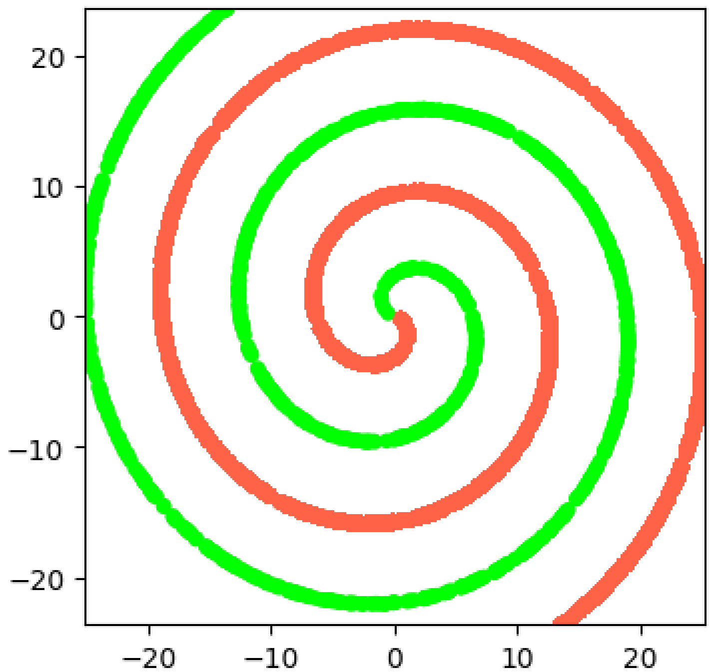
```
953 0.00043605404789559543
954 0.000434983434388414
955 0.00043391494546085596
956 0.00043285085121169686
957 0.00043179080239497125
958 0.00043073485721834004
959 0.00042968092020601034
960 0.00042863472481258214
961 0.00042758966446854174
962 0.0004265500174369663
963 0.0004255112144164741
964 0.00042447986197657883
965 0.0004234512452967465
966 0.0004224247531965375
967 0.0004214039072394371
968 0.00042038553510792553
969 0.00041937243076972663
970 0.0004183637211099267
971 0.0004173557390458882
972 0.00041635450907051563
973 0.0004153554909862578
974 0.0004143599944654852
975 0.0004133683687541634
976 0.00041238032281398773
977 0.00041139646782539785
978 0.00041041598888114095
979 0.0004094376927241683
980 0.00040846329648047686
981 0.0004074940807186067
982 0.00040652777533978224
983 0.00040556458407081664
984 0.00040460482705384493
985 0.0004036497266497463
986 0.0004026966926176101
987 0.0004017468309029937
988 0.0004008013056591153
989 0.00039985895273275673
990 0.0003989199176430702
991 0.0003979845205321908
992 0.0003970518591813743
993 0.00039612234104424715
994 0.0003951969265472144
995 0.00039427465526387095
996 0.0003933551488444209
997 0.0003924389893654734
998 0.0003915277775377035
999 0.00039061764255166054
1000 0.0003897115821018815
```

In [20]:
```
with torch.no_grad():
    fig = plt.figure(figsize=(4,4))
    axes = []
    axes.append(fig.add_subplot(1,1,1))#,sharex=True,sharey=True))
    xs, ys = X, y#sample_gaussian(n_samples=200) ; s = torch.linspace(0, 1,
    y_pred = model(xs)
    print(y_pred.shape)
```

```
    label = (y_pred[:,0] >= 0.5).long()
    print(label)
    colors = ['lime','tomato']
    for i in range(1024):
        axes[0].scatter(xs[i,0], xs[i,1], c=colors[label[i]], edgecolor='nor
        axes[0].scatter(xs[i+1024:,0], xs[i+1024:,1], c=colors[label[i+1024]
    axes[0].set_xlim(xs[:,0].min(), xs[:,0].max()) ; axes[0].set_ylim(xs[:,1
    plt.show()

    print(xs.shape)
    err = torch.sum(torch.abs(label - y))
    print(err)
```

```
torch.Size([2048, 1])
tensor([0, 0, 0,  ..., 1, 1, 1])
```




```
torch.Size([2048, 2])
tensor(0)
```

In [ ]:

```

```