In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['KaiTi']
plt.rcParams['font.serif'] = ['KaiTi']
```

In [2]:

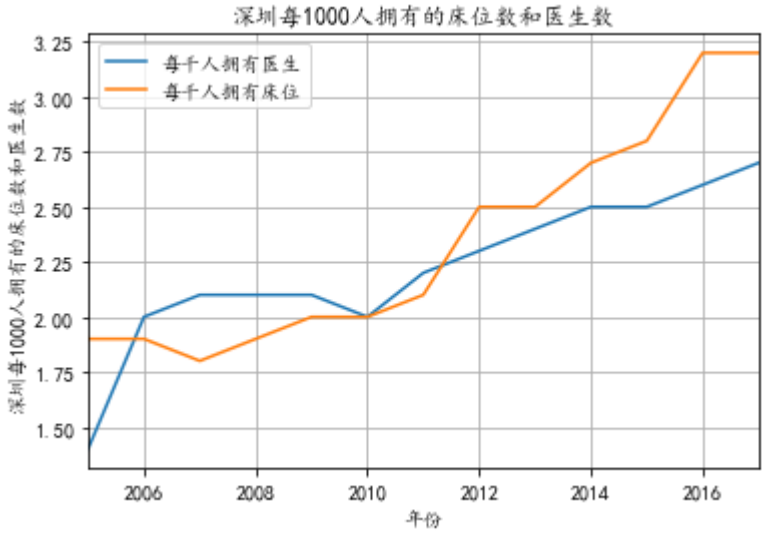
```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
```

In [3]:

```
import numpy as np
from scipy import interpolate
```

In [109]:

```
shen_yi = pd.read_excel('深圳医疗.xlsx')
shen_yi = shen_yi.iloc[26:39]
zhe = shen_yi[['年份','每千人拥有医生','每千人拥有床位']]
key = list(zhe.columns)
value = [round(i, 4) for i in list(zhe.describe().iloc[1].values)]
values = {i:j for i,j in zip(key,value)}
zhe = zhe.fillna(value=values)
zhe = zhe.set_index('年份')
zhe.plot()
plt.grid()
plt.xlabel('年份')
plt.ylabel('深圳每1000人拥有的床位数和医生数')
plt.title('深圳每1000人拥有的床位数和医生数')
plt.savefig('深圳每1000人拥有的床位数和医生数.png',dpi=300)
```



In [112]:

```
# shen_yang = pd.read_excel('深圳养老.xlsx')
# shen_yang = shen_yang.iloc[2:]
# shen_yang = shen_yang.set_index('年份')
# shen_yang
```

In [96]:

```
# guo_yi = pd.read_excel('全国医疗.xlsx')
# guo_yi
```

In [95]:

```
# guo_ren = pd.read_excel('全国医疗人员.xlsx')
# guo_ren = guo_ren[6:]
# guo_ren = guo_ren.set_index('年份')
# guo_ren
```

In [111]:

```
# beds_per_1000 = pd.read_excel('beds_per_1000.xlsx')
# beds_per_1000 = beds_per_1000[5:]
# beds_per_1000 = beds_per_1000.set_index('year')
# key = list(beds_per_1000.columns)
# value = [round(i, 4) for i in list(beds_per_1000.describe().iloc[1].values)]
# values = {i:j for i,j in zip(key,value)}
# beds_per_1000 = beds_per_1000.fillna(value=values)
# beds_per_1000.plot()
# plt.grid()
# plt.xlabel('年份')
# plt.ylabel('每1000人拥有的床位数')
# plt.title('四个国家每1000人拥有的床位数')
# plt.savefig('四个国家每1000人拥有的床位数.png',dpi=300)


# docters_per_1000 = pd.read_excel('docters_per_1000.xlsx')
# docters_per_1000 = docters_per_1000[5:]
# docters_per_1000 = docters_per_1000.set_index('year')
# key = list(docters_per_1000.columns)
# value = [round(i, 4) for i in list(docters_per_1000.describe().iloc[1].values)]
# values = {i:j for i,j in zip(key,value)}
# docters_per_1000 = docters_per_1000.fillna(value=values)
# docters_per_1000.plot()
# plt.grid()
# plt.xlabel('年份')
# plt.ylabel('每1000人拥有的医生数')
# plt.title('四个国家每1000人拥有的医生数')
# plt.savefig('四个国家每1000人拥有的医生数.png',dpi=300)


# gov_pays_per = pd.read_excel('gov_pays_per.xlsx')
# gov_pays_per = gov_pays_per[5:]
# gov_pays_per = gov_pays_per.set_index('year')
# key = list(gov_pays_per.columns)
# value = [round(i, 4) for i in list(gov_pays_per.describe().iloc[1].values)]
# values = {i:j for i,j in zip(key,value)}
# gov_pays_per = gov_pays_per.fillna(value=values)
# gov_pays_per.plot()
# plt.grid()
# plt.xlabel('年份')
# plt.ylabel('政府为每个人医疗支付')
# plt.title('四个国家政府为每个人医疗支付')
# plt.savefig('四个国家政府为每个人医疗支付.png',dpi=300)


# pays_per = pd.read_excel('pays_per.xlsx')
# pays_per = pays_per[5:]
# pays_per = pays_per.set_index('year')
# key = list(pays_per.columns)
# value = [round(i, 4) for i in list(pays_per.describe().iloc[1].values)]
# values = {i:j for i,j in zip(key,value)}
# pays_per = pays_per.fillna(value=values)
# pays_per.plot()
# plt.grid()
# plt.xlabel('年份')
# plt.ylabel('平均每个人医疗支付')
# plt.title('四个国家平均每个人医疗支付')
# plt.savefig('四个国家平均每个人医疗支付.png',dpi=300)
```

插值

In [203]:

```
matic = []
# ################ 读
beds_per_1000 = pd.read_excel('beds_per_1000.xlsx')
docters_per_1000 = pd.read_excel('docters_per_1000.xlsx')
gov_pays_per = pd.read_excel('gov_pays_per.xlsx')

for name in [beds_per_1000,docters_per_1000,gov_pays_per]:
    # ################ 删减

    name = name[:14]
    name = name.set_index('year')
    key = list(name.columns)
    value = [round(i, 4) for i in list(name.describe().iloc[1].values)]
    values = {i:j for i,j in zip(key,value)}
    name = name.fillna(value=values)

    x = np.linspace(0,10,name.shape[0])         # beds 10                beds   13
    y = name['United States'].values

    xnew=np.linspace(0,10,21)
#     plt.plot(x,y,"ro")
#     plt.xticks(x,[str(i) for i in beds_per_1000.index.to_list()])

    f=interpolate.interp1d(x,y,kind="cubic")
    ynew=f(xnew)
    matic.append(ynew)
    # plt.plot(xnew,ynew)
    # plt.grid()
    # plt.xlabel('年份')
    # plt.ylabel('政府为每个人医疗支付\美元')
    # plt.title('冰岛政府为每个人医疗支付三次插值')
    # plt.savefig('冰岛政府为每个人医疗支付三次插值.png',dpi=300)
US_cha = pd.DataFrame({'beds_US':matic[0],'docs_US':matic[1],'dols_US':matic[2]})
US_cha
```

Out[203]:

	beds_US	docs_US	dols_US
0	3.500000	2.594200	2015.092285
1	3.520295	2.248700	2147.358293
2	3.472571	2.207491	2275.682189
3	3.404685	2.293232	2395.923429
4	3.348454	2.373866	2507.018873
5	3.257163	2.540029	2622.955411
6	3.159221	2.712791	2742.678650
7	3.178397	2.580613	2847.121514
8	3.190374	2.404785	2971.693998
9	3.114072	2.418501	3125.853005
10	3.088331	2.440312	3248.792987
11	3.101739	2.435019	3353.919790
12	3.099660	2.438753	3476.024277

	beds_US	docs_US	dols_US
13	3.105980	2.446451	3610.802834
14	3.094678	2.445984	3736.084590
15	3.031269	2.435947	3831.859402
16	2.951377	2.442953	3904.267672
17	2.897702	2.465864	3970.724355
18	2.893226	2.487376	4048.103984
19	2.907621	2.514599	4134.657048
20	2.900000	2.559600	4224.915527

In [201]:

```
beds_per_1000.columns
```

Out[201]:

```
Index(['year', 'China', 'United States', 'Iceland', 'Japan'], dtype='object')
```

RNN

In [262]:

```
#参数配置
num_time_steps = 21
input_size = 1
hidden_size = 16
output_size = 1
lr = 0.03
```

In [263]:

```
#网络定义
class Net(nn.Module):
    def __init__(self, ):
        super(Net, self).__init__()
        self.rnn = nn.RNN(
            input_size=input_size,
            hidden_size=hidden_size,
            num_layers=1,
            batch_first=True,
        )
        for p in self.rnn.parameters():
            nn.init.normal_(p, mean=0.0, std=0.001)
        self.linear = nn.Linear(hidden_size, output_size)

    def forward(self, x, hidden_prev):
        out, hidden_prev = self.rnn(x, hidden_prev)
        # [b, seq, h]
        out = out.view(-1, hidden_size)
        out = self.linear(out)
        out = out.unsqueeze(dim=0)
        return out, hidden_prev
```

In [264]:

```
#常见模型
model = Net()
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr)
hidden_prev = torch.zeros(1, 1, hidden_size)
print(model)
```

```
Net(
  (rnn): RNN(1, 16, batch_first=True)
  (linear): Linear(in_features=16, out_features=1, bias=True)
)
```

In [301]:

```
#训练模型
for iter in range(1000):
    data = US_cha['dols_US'].values/1000
    data = data.reshape(num_time_steps, 1)
    x = torch.tensor(data[:-1]).float().view(1, num_time_steps - 1, 1)
    y = torch.tensor(data[1:]).float().view(1, num_time_steps - 1, 1)


    output, hidden_prev = model(x, hidden_prev)
    hidden_prev = hidden_prev.detach()

    loss = criterion(output, y)
    model.zero_grad()
    loss.backward()
    optimizer.step()

    if loss.item() < 0.0052:
        print("Iteration: {} loss {}".format(iter, loss.item()))
        break

    if iter % 100 == 0:
        print("Iteration: {} loss {}".format(iter, loss.item()))
```

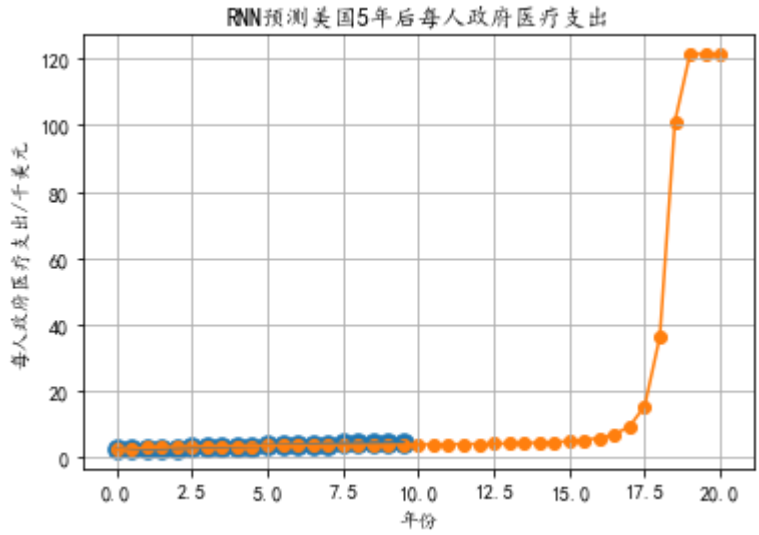
```
Iteration: 0 loss 12.707064628601074
Iteration: 100 loss 0.0065187979489564896
Iteration: 200 loss 0.0061891465447843075
Iteration: 300 loss 0.005901883356273174
Iteration: 400 loss 0.009620944038033485
Iteration: 500 loss 0.006029903888702393
Iteration: 600 loss 0.016757184639573097
Iteration: 700 loss 0.006810381077229977
Iteration: 800 loss 2.931955099105835
Iteration: 900 loss 0.007951842620968819
```

In [302]:

```
#使用模型预测
# start = np.random.randint(3, size=1)[0]
time_steps = np.linspace(0, 0 + 10, num_time_steps)
time_s = np.array(list(np.linspace(0, 0 + 10, num_time_steps))+(list(np.linspace(10, 10 + 10, num_t
# data = np.sin(time_steps)
# data = beds_per_1000['China'].values
data = US_cha['dols_US'].values/1000
data = data.reshape(num_time_steps, 1)
x = torch.tensor(data[:-1]).float().view(1, num_time_steps - 1, 1)
y = torch.tensor(data[1:]).float().view(1, num_time_steps - 1, 1)

predictions = []
inp = x[:, 0, :]
for _ in range(time_s.shape[0]):
    inp = inp.view(1, 1, 1)
    (pred, hidden_prev) = model(inp, hidden_prev)
    inp = pred
    predictions.append(pred.detach().numpy().ravel()[0])

x = x.data.numpy().ravel()
y = y.data.numpy()
plt.scatter(time_steps[:-1], x.ravel(), s=90)
plt.plot(time_steps[:-1], x.ravel())
plt.xlabel('年份')
plt.ylabel('每人政府医疗支出/千美元')
plt.title('RNN预测美国5年后每人政府医疗支出')
plt.scatter(time_s, predictions)
plt.plot(time_s, predictions)
plt.grid()
plt.savefig('RNN预测美国5年后每人政府医疗支出.png',dpi=300)
plt.show()
```



In [241]:

```
# plt.plot(range(len(los)),los)
# plt.xlabel('iterations')
# plt.ylabel('loss')
# plt.title('冰岛每千人拥有床位数训练迭代损失')
# plt.savefig('beds_Iceland_loss.png',dpi=300)
```

In [365]:

```
# x = torch.tensor(data[:-10]).float().view(1, num_time_steps - 10, 1)
# x.shape[1]
# predictions  21
```

Out[365]:

```
array([ 5. ,  5.5,  6. ,  6.5,  7. ,  7.5,  8. ,  8.5,  9. ,  9.5, 10. ,
       10.5, 11. , 11.5, 12. , 12.5, 13. , 13.5, 14. , 14.5, 15. ])
```

In [369]:

```
# y1 = torch.tensor(data[1:]).float().view(1, num_time_steps - 1, 1)
# y2 = torch.tensor(data[1:]).float().view(1, num_time_steps - 1, 1)
# y3 = torch.tensor(data[1:]).float().view(1, num_time_steps - 1, 1)
# torch.cat([y1,y2,y3],dim=2)
```

In [252]:

```
data = beds_per_1000['United States'].values
data
```

Out[252]:

```
array([3.2   , 3.1   , 3.1   , 3.1   , 3.1   , 3.    , 2.9   , 2.9   ,
       2.9   , 3.0333])
```

In [186]:

```
IceLand_cha['dols_Iceland'].values/1000
```

Out[186]:

```
array([2.14866309, 2.19608322, 2.35182612, 2.52175285, 2.63027156,
       2.69480605, 2.74072943, 2.76819179, 2.79934135, 2.85226483,
       2.92358281, 3.01650038, 3.11506522, 3.15323258, 3.0379418 ,
       2.77628432, 2.64171719, 2.65865618, 2.68619378, 2.74386285,
       2.88402612])
```

数据

In [305]:

```
# beds_china = ynew
# beds_US = ynew
# beds_Iceland = ynew
# beds_Japan = ynew

# docs_china = ynew
# docs_US = ynew
# docs_Iceland = ynew
# docs_Japan = ynew

# dols_Japan = ynew
# dols_Iceland = ynew
# dols_US = ynew
# dols_china = ynew
```

In [313]:

```
data = beds_china
data = data.reshape(num_time_steps, 1)
data
```

Out[313]:

```
array([[2.45      ],
       [1.91006522],
       [2.11996317],
       [2.69760214],
       [3.26089042],
       [3.45304332],
       [3.37158811],
       [3.50048838],
       [3.94478667],
       [4.20235471],
       [3.95842095],
       [3.58744526],
       [3.50877933],
       [3.70374496],
       [3.99962036],
       [4.20301139],
       [4.1140961 ],
       [3.73938309],
       [3.32417584],
       [3.1270797 ],
       [3.4067    ]])
```

In [ ]: