代码

May 23, 2022

[97]:

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import torch
from    torch import  nn,optim
from    torch.nn import functional as F
```

[3]:

```
data=pd.read_csv('data.csv')
```

[4]:

```
data.x=data.x-1897
```

[5]:

```
x_2d=np.zeros([26,1])
x_2d[:,0]=data.x
y_1d=data.y
```

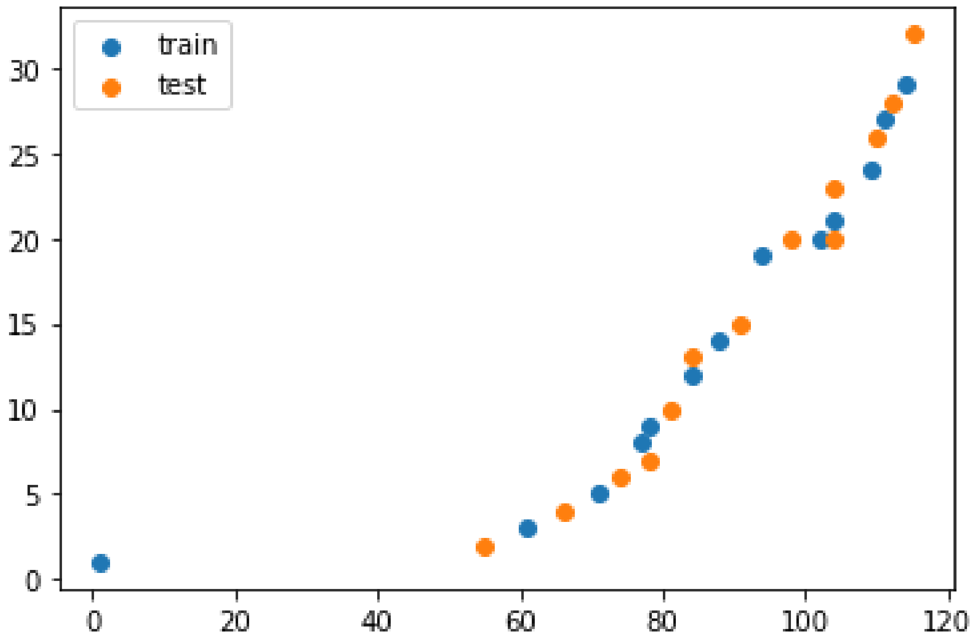
[176]:

```
train_x,test_x =np.zeros([13,1]),np.zeros([13,1])
train_y,test_y=np.zeros([13]),np.zeros([13])

train_x=x_2d[0:26:2,:]
train_y=y_1d[0:26:2].values
test_x=x_2d[1:26:2,:]
test_y=y_1d[1:26:2].values
```

[131]:

```
plt.scatter(train_x,train_y,label='train')
plt.scatter(test_x,test_y,label='test')
plt.legend()
plt.show()
```

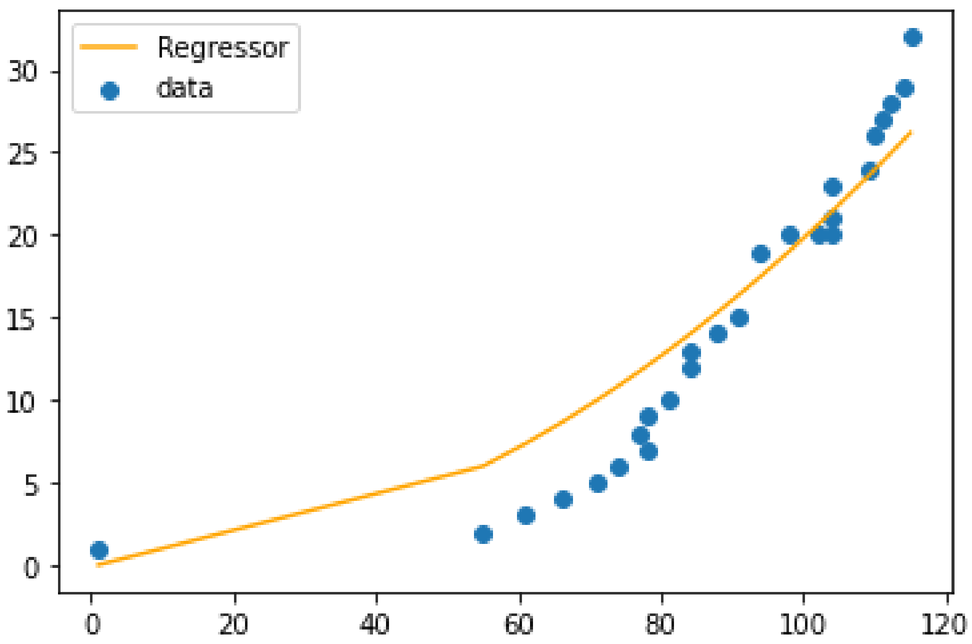


[8]:

```
a=0
b=0
for i in range(len(data.x)):
    a=a+data.x[i]**2 *data.y[i]
    b=b+data.x[i]**2
result=a/b
```

[111]:

```
t=data.x
y=result*t**2
plt.plot(t,y,c='orange',label='Regressor')
plt.scatter(data.x,data.y,label='data')
plt.legend()
plt.show()
```

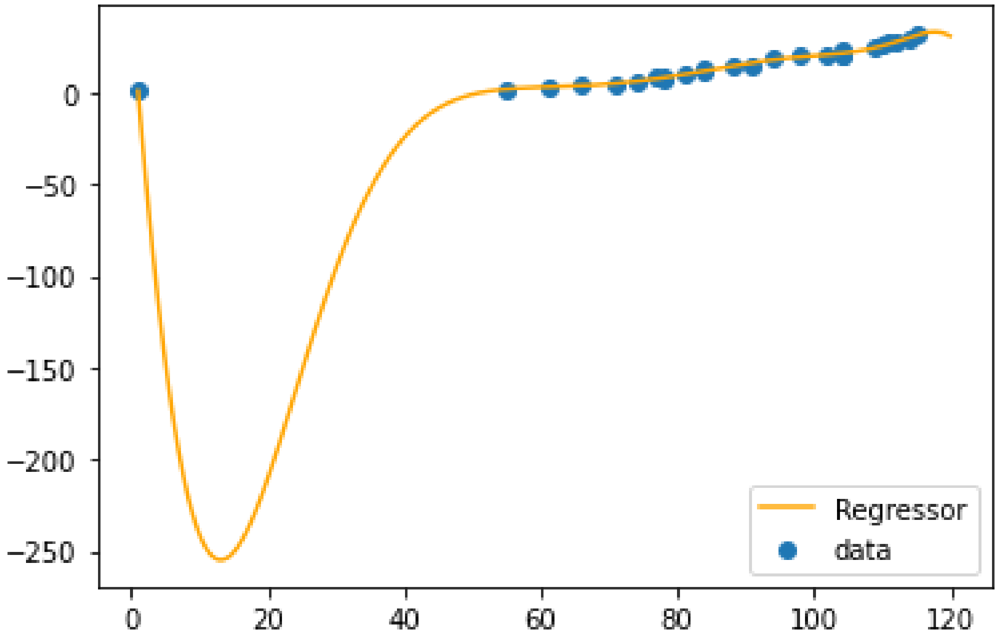


```
[10]: x=123
      y=- 1.639136814e-15*x**10 + 1.059936718e-12*x**9 - 0.0000000002960388319*x**8 +␣
       ↪0.00000004652752046*x**7 - 0.000004472787601*x**6 + 0.0002650337924*x**5 - 0.
       ↪00900509148*x**4 + 0.126851489*x**3 + 1.240266286*x**2 - 47.08797047*x + 46.
       ↪72959737
```

```
[11]: y
```

```
[11]: 13.78173572648101
```

```
[112]: x=np.arange(1,120,0.1)
       y=- 1.639136814e-15*x**10 + 1.059936718e-12*x**9 - 0.0000000002960388319*x**8 +␣
        ↪0.00000004652752046*x**7 - 0.000004472787601*x**6 + 0.0002650337924*x**5 - 0.
        ↪00900509148*x**4 + 0.126851489*x**3 + 1.240266286*x**2 - 47.08797047*x + 46.
        ↪72959737
       plt.plot(x,y,c='orange',label='Regressor')
       plt.scatter(data.x,data.y,label='data')
       plt.legend()
       plt.show()
```



[13]:

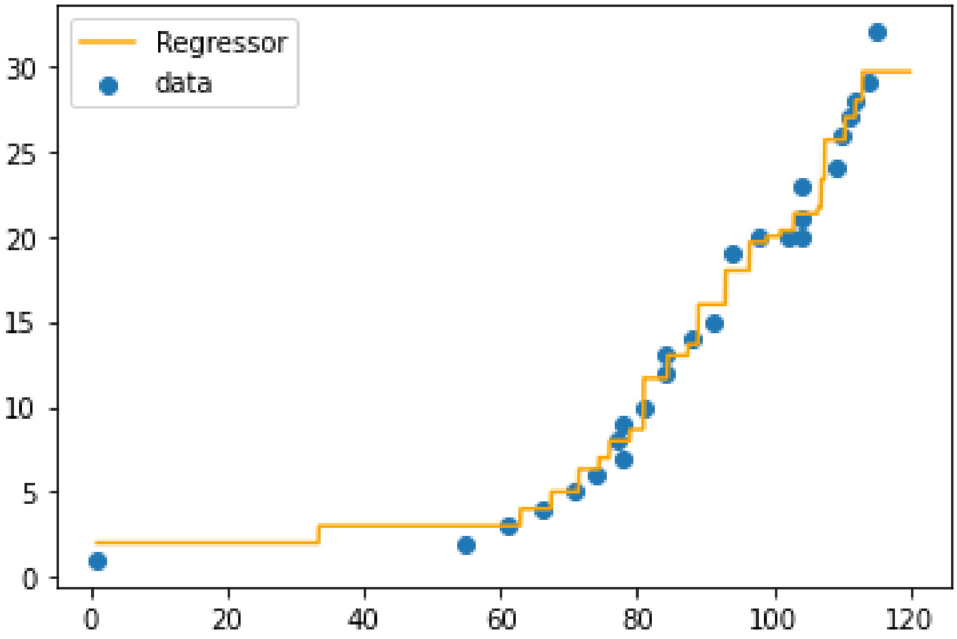
```
from sklearn.neighbors import KNeighborsRegressor
neigh = KNeighborsRegressor(n_neighbors=3)
neigh.fit(x_2d, y_1d)
neigh.predict([[123]])
```

[13]:

```
array([29.66666667])
```

[114]:

```
t=np.arange(1,120,0.1)
y=np.arange(1,120,0.1)
for i in range(len(t)):
    y[i]=neigh.predict([[t[i]]])
plt.plot(t,y,c='orange',label='Regressor')
plt.scatter(data.x,data.y,label='data')
plt.legend()
plt.show()
```



[72]:
```
class ResNet18(nn.Module):

    def __init__(self):
        super(ResNet18, self).__init__()

        self.conv1 = nn.Sequential(
            nn.Linear(2, 5)
        )
        self.conv2 = nn.Sequential(
           nn.Linear(5, 10)
        )
        self.conv3 = nn.Sequential(
           nn.Linear(10, 5)
        )
        self.conv4 = nn.Sequential(
           nn.Linear(1, 5)
        )

        self.outlayer = nn.Linear(5, 1)

    def forward(self, x):
        """

        :param x:
        :return:
```

```
        """

        x = F.relu(self.conv1(x))
        x = x@x.T
        x = F.relu(self.conv4(x))
        x = F.relu(self.conv2(x))
        x = F.relu(self.conv3(x))
        x = self.outlayer(x)
        return x
```

[73]:

```
model=ResNet18()
optimizer = optim.Adam(model.parameters(), lr=1e-3)
train=torch.from_numpy(data.x.values).to(torch.float32)
test=torch.from_numpy(data.y.values).to(torch.float32)
for i in range(2000):
    for j in range(25):
        x=torch.zeros(1,2)
        x[0]=train[j]
        model.train()
        optimizer.zero_grad()
        logits = model(x)
        loss_fn=torch.nn.MSELoss(reduce=False, size_average=False)
        loss= loss_fn(logits, test[j])
        loss.backward()
        optimizer.step()
```

```
C:\Users\Administrator\Anaconda3\lib\site-packages\torch\nn\_reduction.py:42:
UserWarning: size_average and reduce args will be deprecated, please use
reduction='none' instead.
  warnings.warn(warning.format(ret))
C:\Users\Administrator\Anaconda3\lib\site-packages\torch\nn\modules\loss.py:520:
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)
```

[74]:

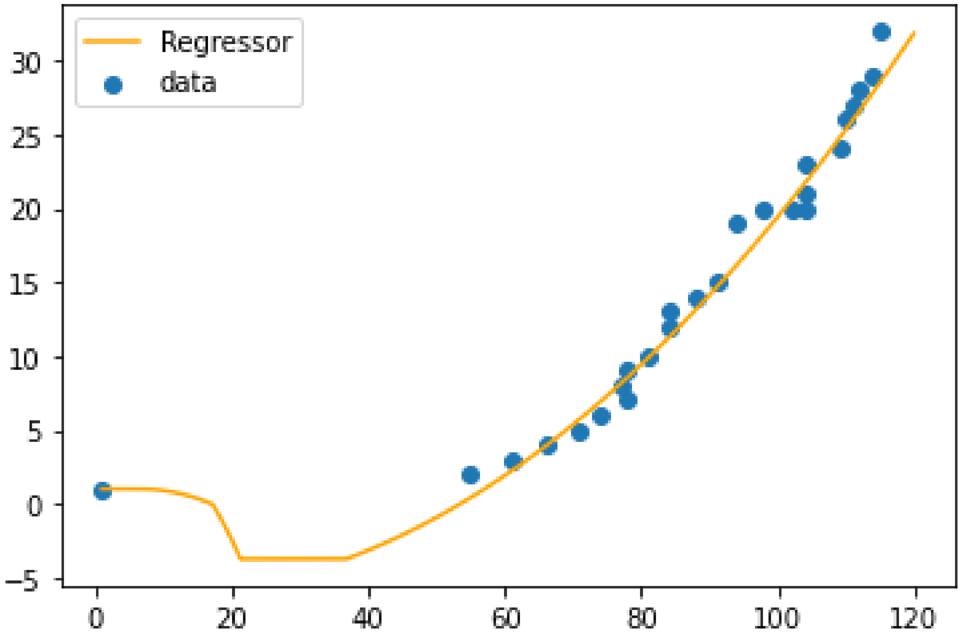
```
x=torch.zeros(1,2)
x[0]=123
model(x)
```

[74]:

```
tensor([[33.9991]], grad_fn=<AddmmBackward0>)
```

[115]:

```
t=np.arange(1,120,0.1)
y=np.arange(1,120,0.1)
for i in range(len(t)):
    x=torch.zeros(1,2)
    x[0]=t[i]
    y[i]=model(x)
plt.plot(t,y,c='orange',label='Regressor')
plt.scatter(data.x,data.y,label='data')
plt.legend()
plt.show()
```



[76]:

```
(34.00+29.67+29.97)/3
```

[76]:

```
31.213333333333335
```

[178]:

```
a=0
b=0
for i in range(len(train_x[:,0])):
    a=a+train_x[i,0]**2 *train_y[i]
    b=b+train_x[i,0]**4
result=a/b
```

[179]:

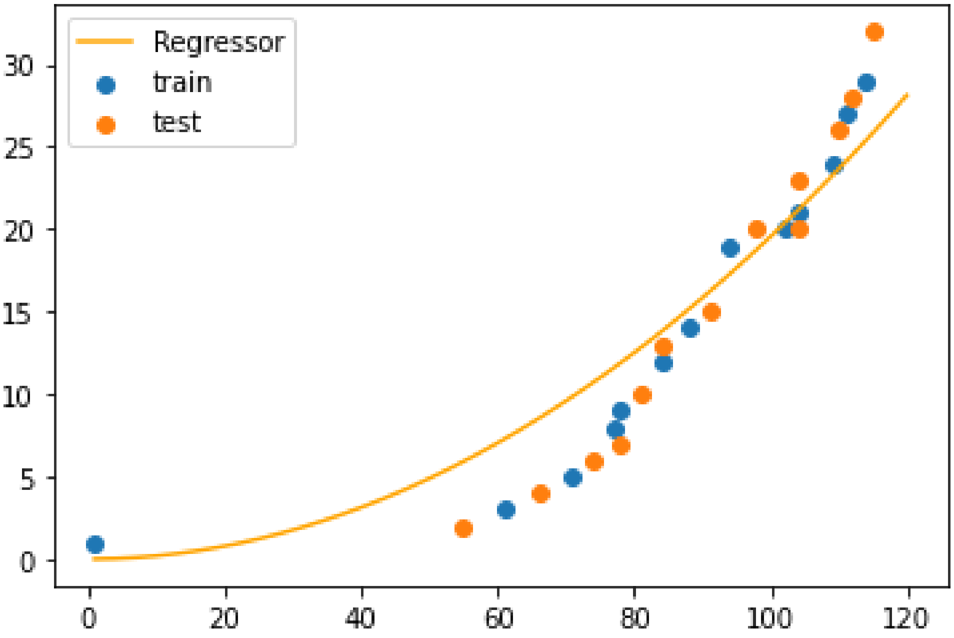
```
result
```

[179]:

```
0.001956146330816694
```

[180]:

```
t=np.arange(1,120,0.1)
y=result*t**2
plt.plot(t,y,c='orange',label='Regressor')
plt.scatter(train_x,train_y,label='train')
plt.scatter(test_x,test_y,label='test')
plt.legend()
plt.show()
```

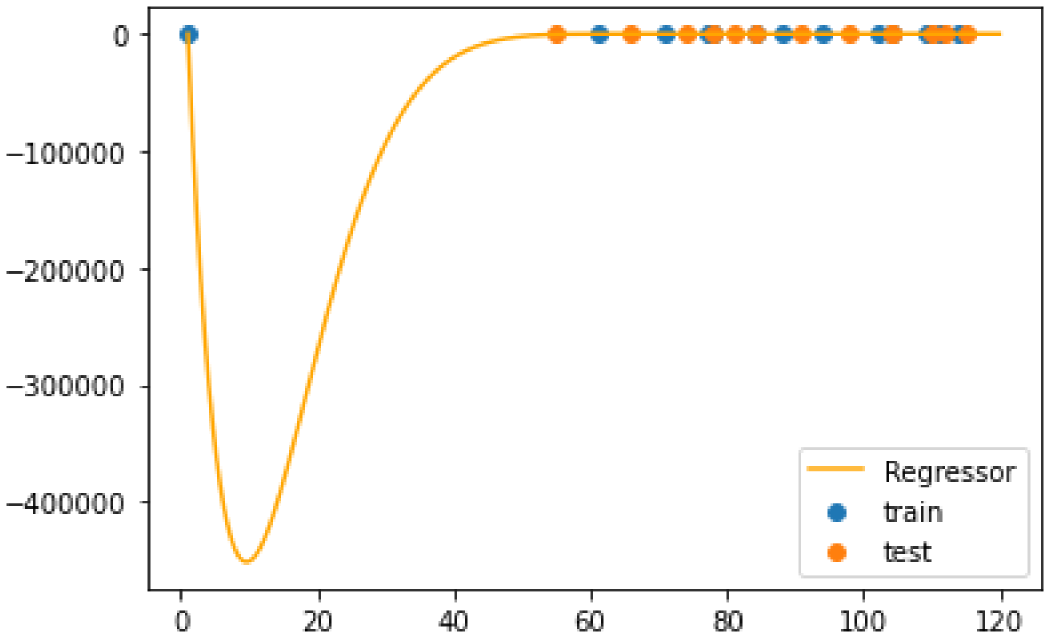


[260]:
```
((test_y-result*g[:,0])**2).sum()
```

[260]:
```
4395.533703057064
```

[191]:
```
x=123
y=3.975590632e-13*x**10 - 0.0000000003290771782*x**9 + 0.0000001204184953*x**8␣
 ↪- 0.00002556825203*x**7 + 0.003471943435*x**6 - 0.3127761976*x**5 + 18.
 ↪70628778*x**4 - 717.4565232*x**3 + 16097.80278*x**2 - 165200.0207*x + 149802.
 ↪2775
```

[192]:
```
x=np.arange(1,120,0.1)
y=3.975590632e-13*x**10 - 0.0000000003290771782*x**9 + 0.0000001204184953*x**8␣
 ↪- 0.00002556825203*x**7 + 0.003471943435*x**6 - 0.3127761976*x**5 + 18.
 ↪70628778*x**4 - 717.4565232*x**3 + 16097.80278*x**2 - 165200.0207*x + 149802.
 ↪2775
plt.plot(x,y,c='orange',label='Regressor')
plt.scatter(train_x,train_y,label='train')
plt.scatter(test_x,test_y,label='test')
plt.legend()
plt.show()
```



[196]:
```
x=test_x[:,0]
y=3.975590632e-13*x**10 - 0.0000000003290771782*x**9 + 0.0000001204184953*x**8␣
 ↪- 0.00002556825203*x**7 + 0.003471943435*x**6 - 0.3127761976*x**5 + 18.
 ↪70628778*x**4 - 717.4565232*x**3 + 16097.80278*x**2 - 165200.0207*x + 149802.
 ↪2775
```

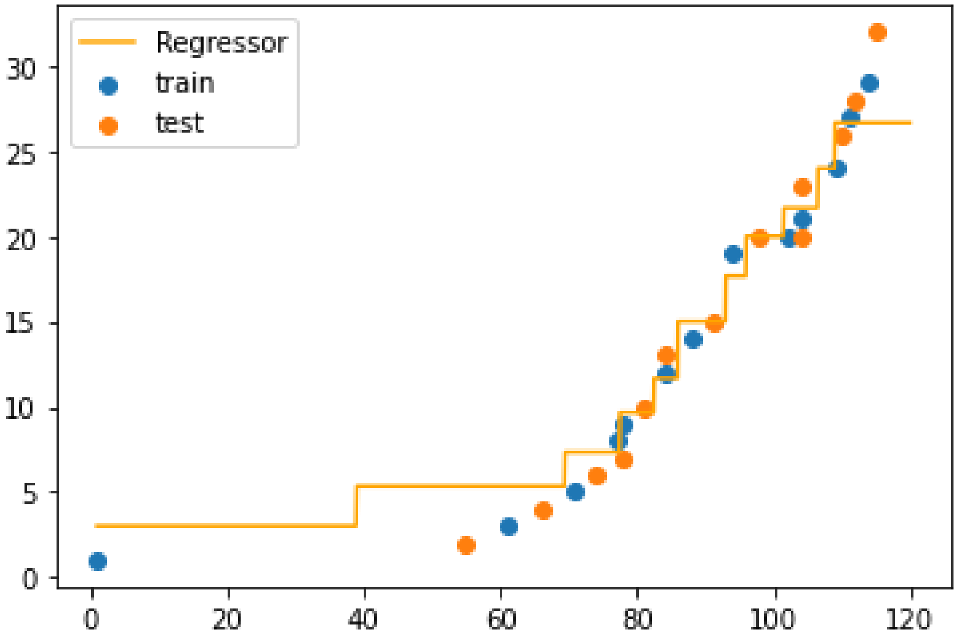
[200]:
```
((y-test_y)**2).sum()
```

[200]:
```
177174.24941002886
```

[202]:
```
from sklearn.neighbors import KNeighborsRegressor
neigh = KNeighborsRegressor(n_neighbors=3)
neigh.fit(train_x, train_y)
```

[202]:
```
KNeighborsRegressor(n_neighbors=3)
```

[203]:
```
t=np.arange(1,120,0.1)
y=np.arange(1,120,0.1)
for i in range(len(t)):
    y[i]=neigh.predict([[t[i]]])
plt.plot(t,y,c='orange',label='Regressor')
plt.scatter(train_x,train_y,label='train')
plt.scatter(test_x,test_y,label='test')
plt.legend()
plt.show()
```



[219]:

```
t=test_x
y=np.zeros([13,1])
for i in range(len(t)):
    y[i]=neigh.predict([t[i,:]])
```

[221]:

```
((y-test_y)**2).sum()
```

[221]:

```
25636.222222222226
```

[225]:

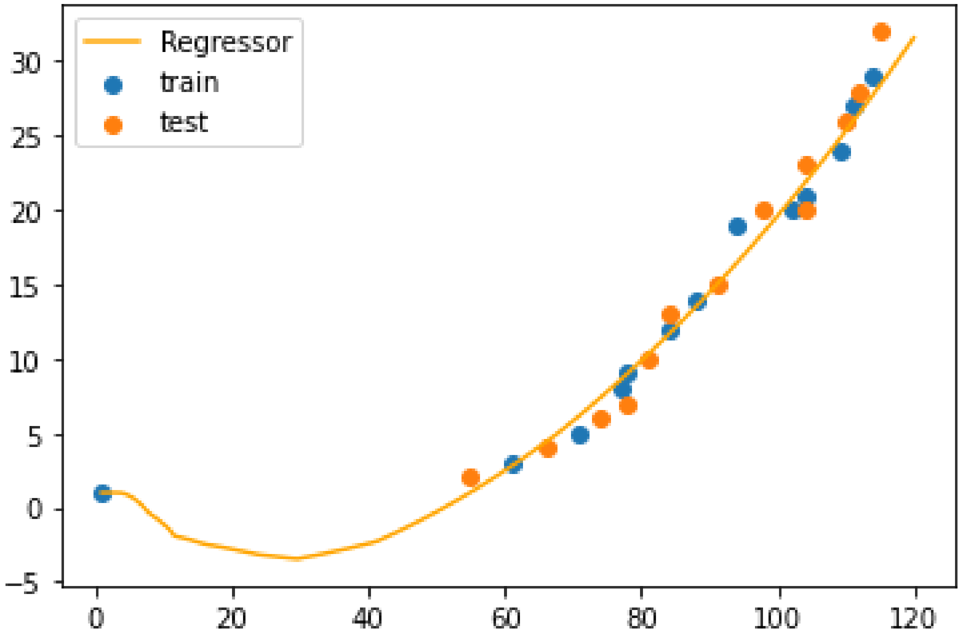
```
model=ResNet18()
optimizer = optim.Adam(model.parameters(), lr=1e-3)
train=torch.from_numpy(train_x).to(torch.float32)
test=torch.from_numpy(train_y).to(torch.float32)
for i in range(2000):
    for j in range(13):
        x=torch.zeros(1,2)
        x[0]=train[j]
        model.train()
        optimizer.zero_grad()
        logits = model(x)
        loss_fn=torch.nn.MSELoss(reduce=False, size_average=False)
        loss= loss_fn(logits, test[j])
        loss.backward()
        optimizer.step()
```

```
C:\Users\Administrator\Anaconda3\lib\site-packages\torch\nn\_reduction.py:42:
```

```
UserWarning: size_average and reduce args will be deprecated, please use
reduction='none' instead.
  warnings.warn(warning.format(ret))
C:\Users\Administrator\Anaconda3\lib\site-packages\torch\nn\modules\loss.py:520:
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)
```

[226]:

```
t=np.arange(1,120,0.1)
y=np.arange(1,120,0.1)
for i in range(len(t)):
    x=torch.zeros(1,2)
    x[0]=t[i]
    y[i]=model(x)
plt.plot(t,y,c='orange',label='Regressor')
plt.scatter(train_x,train_y,label='train')
plt.scatter(test_x,test_y,label='test')
plt.legend()
plt.show()
```



[248]:

```
y=torch.zeros([13,1])
x=torch.zeros(1,2,)
for i in range(len(t)):
    x[0]=test_x[i,0]
    y[i]=model(x)
```

```
[258]: ((y[:,0]-torch.from_numpy(test_y))**2).sum()
```

```
[258]: tensor(31.2013, grad_fn=<SumBackward0>)
```

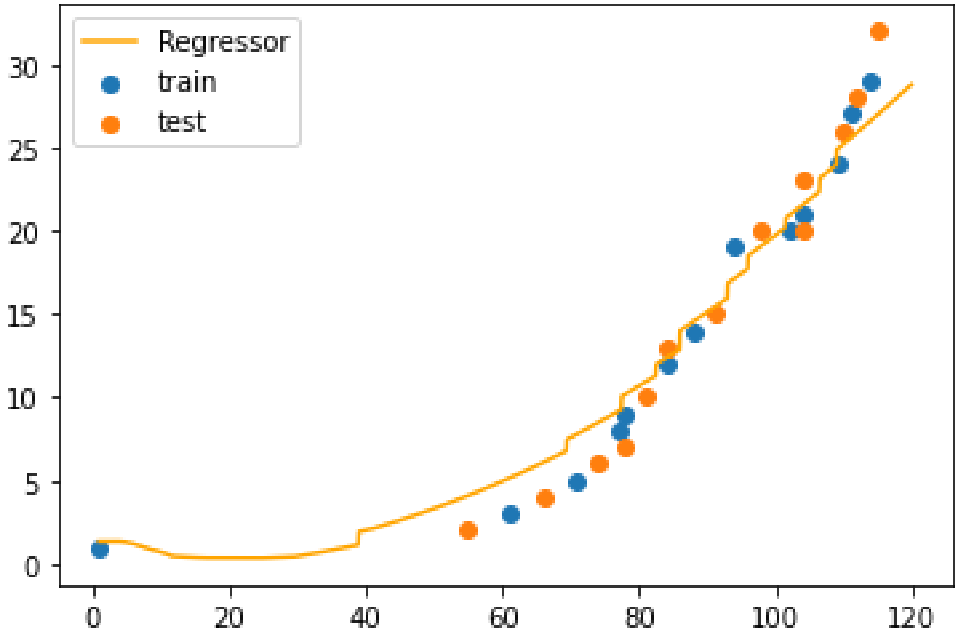
```
[262]: t=np.arange(1,120,0.1)
       y1=result*t**2
       y2=np.arange(1,120,0.1)
       for i in range(len(t)):
           y2[i]=neigh.predict([[t[i]]])

       y3=np.arange(1,120,0.1)

       for i in range(len(t)):
           x=torch.zeros(1,2)
           x[0]=t[i]
           y3[i]=model(x)

       y=(y1+y2+y3)/3

       plt.plot(t,y,c='orange',label='Regressor')
       plt.scatter(train_x,train_y,label='train')
       plt.scatter(test_x,test_y,label='test')
       plt.legend()
       plt.show()
```



[271]:

```
y1=result*test_x**2

t=test_x
y2=np.zeros([13,1])
for i in range(len(t)):
    y2[i]=neigh.predict([t[i,:]])

y3=torch.zeros([13,1])
y4=y3.numpy()
x=torch.zeros(1,2,)
for i in range(len(t)):
    x[0]=test_x[i,0]
    y3[i]=model(x)
```

[272]:

```
((((y1+y2+y4)/3)-test_y)**2).sum()
```

[272]:

```
24614.146525273478
```