1_iris

吴清柳

July 24, 2023

1 Iris classification

鸢尾花分类–使用全连接模型, Pytorch 模型，CPU 上训练

测试集上准确度：0.9666666666666667

[6]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import Dataset, DataLoader
from sklearn.model_selection import train_test_split

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[7]:
```
dataset = pd.read_csv("../dataset/iris.data")
```

[8]:
```
dataset.columns = [
    "sepal length(cm)",
    "sepal width(cm)",
    "petal length(cm)",
    "petal width(cm)",
    "species",
]
dataset.head()
```

[8]:
```
   sepal length(cm)  sepal width(cm)  petal length(cm)  petal width(cm)  \
0               4.9              3.0               1.4              0.2
1               4.7              3.2               1.3              0.2
2               4.6              3.1               1.5              0.2
3               5.0              3.6               1.4              0.2
4               5.4              3.9               1.7              0.4

       species
0  Iris-setosa
1  Iris-setosa
2  Iris-setosa
```

```
3    Iris-setosa
4    Iris-setosa
```

[9]:
```
# Transform species data to numeric values
mappings = {"Iris-setosa": 0, "Iris-versicolor": 1, "Iris-virginica": 2}
dataset["species"] = dataset["species"].apply(lambda x: mappings[x])
dataset.head()
```

[9]:
```
   sepal length(cm)  sepal width(cm)  petal length(cm)  petal width(cm)  \
0               4.9              3.0               1.4              0.2
1               4.7              3.2               1.3              0.2
2               4.6              3.1               1.5              0.2
3               5.0              3.6               1.4              0.2
4               5.4              3.9               1.7              0.4

   species
0        0
1        0
2        0
3        0
4        0
```

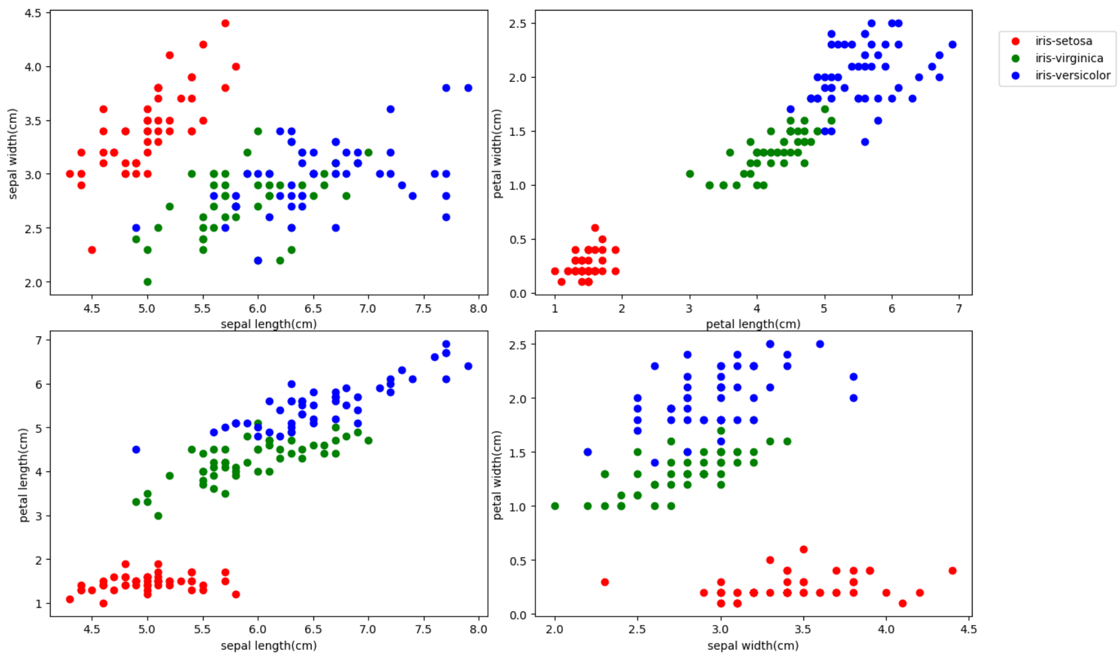
[14]:
```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 8))
fig.tight_layout()

plots = [(0, 1), (2, 3), (0, 2), (1, 3)]
colors = ["r", "g", "b"]
labels = ["iris-setosa", "iris-virginica", "iris-versicolor"]
print(dataset.columns)

for i, ax in enumerate(axes.flat):
    for j in range(3):
        x = dataset.columns[plots[i][0]]
        y = dataset.columns[plots[i][1]]
        ax.scatter(
            dataset[dataset["species"] == j][x],
            dataset[dataset["species"] == j][y],
            color=colors[j],
        )
        ax.set(xlabel=x, ylabel=y)

fig.legend(labels=labels, loc=3, bbox_to_anchor=(1.0, 0.85))
plt.show()
```

```
Index(['sepal length(cm)', 'sepal width(cm)', 'petal length(cm)',
       'petal width(cm)', 'species'],
      dtype='object')
```



[15]:
```
# loading dataset
X=dataset.drop("species",axis=1).values
y=dataset["species"].values

# split dataset to training and test set by ratio 8:2
# Using scikit-learn's random train and test split function
X_train,X_test, y_train,y_test=train_test_split(X,y,test_size=0.2)
X_train = torch.FloatTensor(X_train)
X_test=torch.FloatTensor(X_test)
y_train=torch.LongTensor(y_train)
y_test=torch.LongTensor(y_test)
```

1.0.1 Creating Model

- FC 4 input, 25 output
- FC 25 input, 30 output
- FC 30 input, 3 output

ReLU as activation function

[16]:
```
class Model(nn.Module):
    def __init__(self, input_feats=4, hidden_layer1=25, hidden_layer2=30,␣
 ↪output_feats=3) -> None:
        super().__init__()
        self.fc1 = nn.Linear(input_feats, hidden_layer1)
        self.fc2=nn.Linear(hidden_layer1, hidden_layer2)
```

```python
        self.out=nn.Linear(hidden_layer2, output_feats)

    def forward(self, x):
        x=F.relu(self.fc1(x))
        x=F.relu(self.fc2(x))
        x=self.out(x)

        return x
```

[17]:
```python
model = Model()
model
```

[17]:
```
Model(
  (fc1): Linear(in_features=4, out_features=25, bias=True)
  (fc2): Linear(in_features=25, out_features=30, bias=True)
  (out): Linear(in_features=30, out_features=3, bias=True)
)
```

[18]:
```python
# Adam optimizer, learning rate=0.01
criterion=nn.CrossEntropyLoss()
optimizer=torch.optim.Adam(model.parameters(),lr=0.01)
```

1.0.2 Training

select epoch as 100

[19]:
```python
epochs=100
losses=[]
for i in range(epochs):
    y_pred=model.forward(X_train)
    loss=criterion(y_pred,y_train)
    losses.append(loss)
    print(f'epoch: {i:2} loss: {loss.item():10.8f}')

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

```
epoch:  0 loss: 1.07880127
epoch:  1 loss: 0.99530464
epoch:  2 loss: 0.93395382
epoch:  3 loss: 0.87158877
epoch:  4 loss: 0.80697405
epoch:  5 loss: 0.73825341
epoch:  6 loss: 0.66753036
epoch:  7 loss: 0.60140502
epoch:  8 loss: 0.54559052
epoch:  9 loss: 0.49762520
```

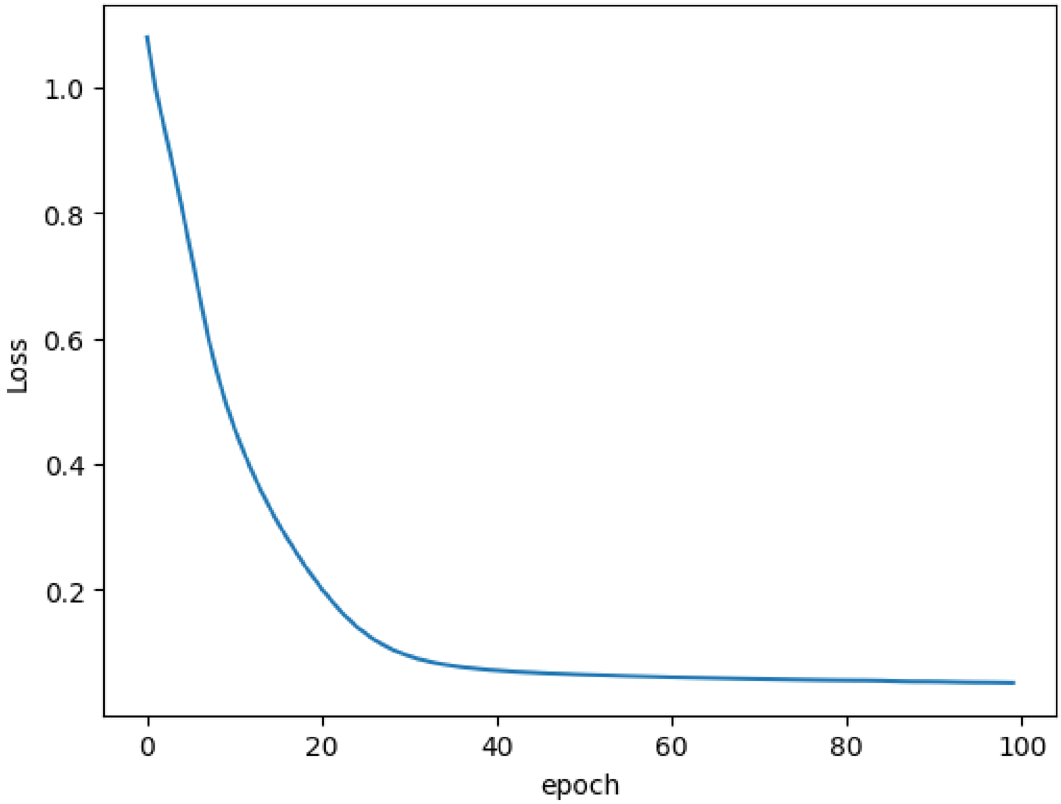
```
epoch: 10 loss: 0.45678517
epoch: 11 loss: 0.42075697
epoch: 12 loss: 0.38821664
epoch: 13 loss: 0.35862112
epoch: 14 loss: 0.33107480
epoch: 15 loss: 0.30519435
epoch: 16 loss: 0.28274354
epoch: 17 loss: 0.26062825
epoch: 18 loss: 0.23899034
epoch: 19 loss: 0.22006868
epoch: 20 loss: 0.20076425
epoch: 21 loss: 0.18431266
epoch: 22 loss: 0.16772504
epoch: 23 loss: 0.15393740
epoch: 24 loss: 0.14070341
epoch: 25 loss: 0.13031510
epoch: 26 loss: 0.12001187
epoch: 27 loss: 0.11237890
epoch: 28 loss: 0.10497826
epoch: 29 loss: 0.09937707
epoch: 30 loss: 0.09430575
epoch: 31 loss: 0.08988739
epoch: 32 loss: 0.08669277
epoch: 33 loss: 0.08330768
epoch: 34 loss: 0.08091813
epoch: 35 loss: 0.07882623
epoch: 36 loss: 0.07669885
epoch: 37 loss: 0.07531521
epoch: 38 loss: 0.07389495
epoch: 39 loss: 0.07250240
epoch: 40 loss: 0.07159806
epoch: 41 loss: 0.07063865
epoch: 42 loss: 0.06962064
epoch: 43 loss: 0.06892148
epoch: 44 loss: 0.06828201
epoch: 45 loss: 0.06750734
epoch: 46 loss: 0.06684444
epoch: 47 loss: 0.06635882
epoch: 48 loss: 0.06583561
epoch: 49 loss: 0.06523165
epoch: 50 loss: 0.06470645
epoch: 51 loss: 0.06429570
epoch: 52 loss: 0.06388523
epoch: 53 loss: 0.06341837
epoch: 54 loss: 0.06294944
epoch: 55 loss: 0.06254649
epoch: 56 loss: 0.06219992
epoch: 57 loss: 0.06185585
```

```
epoch: 58 loss: 0.06148662
epoch: 59 loss: 0.06110198
epoch: 60 loss: 0.06073532
epoch: 61 loss: 0.06040291
epoch: 62 loss: 0.06009902
epoch: 63 loss: 0.05980722
epoch: 64 loss: 0.05951262
epoch: 65 loss: 0.05921249
epoch: 66 loss: 0.05890696
epoch: 67 loss: 0.05860454
epoch: 68 loss: 0.05830955
epoch: 69 loss: 0.05802541
epoch: 70 loss: 0.05775234
epoch: 71 loss: 0.05748898
epoch: 72 loss: 0.05723372
epoch: 73 loss: 0.05698566
epoch: 74 loss: 0.05674486
epoch: 75 loss: 0.05651178
epoch: 76 loss: 0.05629017
epoch: 77 loss: 0.05608093
epoch: 78 loss: 0.05589624
epoch: 79 loss: 0.05573207
epoch: 80 loss: 0.05561866
epoch: 81 loss: 0.05550870
epoch: 82 loss: 0.05545048
epoch: 83 loss: 0.05525275
epoch: 84 loss: 0.05500262
epoch: 85 loss: 0.05455121
epoch: 86 loss: 0.05413490
epoch: 87 loss: 0.05384509
epoch: 88 loss: 0.05371707
epoch: 89 loss: 0.05366949
epoch: 90 loss: 0.05356144
epoch: 91 loss: 0.05336954
epoch: 92 loss: 0.05305876
epoch: 93 loss: 0.05276483
epoch: 94 loss: 0.05255179
epoch: 95 loss: 0.05242549
epoch: 96 loss: 0.05233123
epoch: 97 loss: 0.05219265
epoch: 98 loss: 0.05200486
epoch: 99 loss: 0.05176898
```

[25]:
```python
detached_loss = [x.detach().numpy() for x in losses]
# detached_loss = losses
plt.plot(range(epochs), detached_loss)
plt.ylabel('Loss')
```

```
plt.xlabel('epoch')
```

[25]: Text(0.5, 0, 'epoch')



1.0.3 Validating and testing the model

[26]:
```
preds=[]
with torch.no_grad():
    for val in X_test:
        y_hat=model.forward(val)
        preds.append(y_hat.argmax().item())
```

[27]:
```
df=pd.DataFrame({'Y': y_test, 'Y_hat':preds})
df['Correct']=[1 if corr==pred else 0 for corr, pred in zip(df['Y'],
 ↪df['Y_hat'])]
df
```

[27]:

	Y	Y_hat	Correct
0	2	2	1
1	0	0	1
2	0	0	1

```
3   2      2       1
4   2      2       1
5   0      0       1
6   2      2       1
7   1      1       1
8   1      1       1
9   1      1       1
10  1      1       1
11  1      1       1
12  2      2       1
13  2      2       1
14  2      1       0
15  2      2       1
16  1      1       1
17  1      1       1
18  2      2       1
19  2      2       1
20  0      0       1
21  0      0       1
22  0      0       1
23  0      0       1
24  0      0       1
25  0      0       1
26  1      1       1
27  2      2       1
28  2      2       1
29  2      2       1
```

[28]:
```python
# Accuracy

df['Correct'].sum()/len(df)
```

[28]:
```
0.9666666666666667
```

1.0.4 Apply the model to classify new, unseen data

[29]:
```python
unknown_iris = torch.tensor([4.0,3.3,1.7,0.5])
```

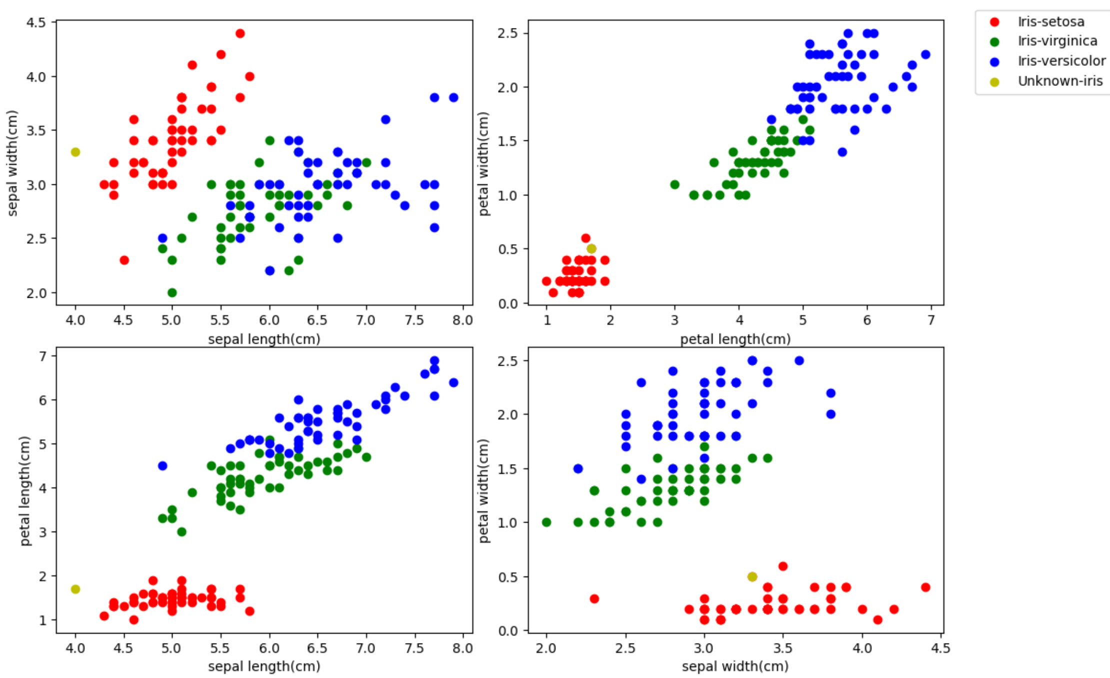
[30]:
```python
fig, axes=plt.subplots(nrows=2,ncols=2,figsize=(10,7))
fig.tight_layout()

plots=[(0,1),(2,3),(0,2),(1,3)]
colors=['r','g','b']
labels=['Iris-setosa','Iris-virginica','Iris-versicolor','Unknown-iris']
for i,ax in enumerate(axes.flat):
    for j in range(3):
        x=dataset.columns[plots[i][0]]
```

```
        y=dataset.columns[plots[i][1]]
        ax.scatter(dataset[dataset['species']==j][x],␣
↪dataset[dataset['species']==j][y],color=colors[j])
        ax.set(xlabel=x,ylabel=y)

    # Add a plot for unknown iris
    ax.scatter(unknown_iris[plots[i][0]], unknown_iris[plots[i][1]], color='y')

fig.legend(labels=labels,loc=3,bbox_to_anchor=(1.0,0.85))
plt.show()
```



Unknown-iris falls into red point group(Iris-setosa).

[31]:
```
with torch.no_grad():
    print(model(unknown_iris),'\n')
    print(labels[model(unknown_iris).argmax()])
```

```
tensor([ 11.3244,   5.7368, -17.0120])

Iris-setosa
```

[ ]:
```
```