In [8]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
import scipy.special
import matplotlib.pyplot as plt
%matplotlib inline
```

In [9]:
```
iris=load_iris()
irisdf=pd.DataFrame(iris.data,columns=iris.feature_names)
irisdf['target']=iris.target
irisdf
```

Out[9]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

In [10]:
```
# 데이터 슬라이싱
iris_group=irisdf.groupby('target')
iris_setosa=iris_group.get_group(0)
iris_versicolor=iris_group.get_group(1)
iris_virginica=iris_group.get_group(2)

list_setosa=list(range(50))
list_versicolor=list(range(50, 100))
list_virginica=list(range(100, 150))

np.random.shuffle(list_setosa)
np.random.shuffle(list_versicolor)
np.random.shuffle(list_virginica)


test_dataset=irisdf
train_dataset=irisdf

for i in range(50//5):
    s_setosa=list_setosa.pop()
    s_versicolor=list_versicolor.pop()
    s_virginica=list_virginica.pop()
    train_dataset=train_dataset.drop(s_setosa)
    train_dataset=train_dataset.drop(s_versicolor)
```

```
    train_dataset=train_dataset.drop(s_virginica)
test_dataset=test_dataset.drop(test_dataset.index[train_dataset.index])
train_dataset
```

Out[10]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
5	5.4	3.9	1.7	0.4	0
6	4.6	3.4	1.4	0.3	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

120 rows × 5 columns

In [11]:
```
test_dataset
```

Out[11]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
15	5.7	4.4	1.5	0.4	0
22	4.6	3.6	1.0	0.2	0
31	5.4	3.4	1.5	0.4	0
36	5.5	3.5	1.3	0.2	0
38	4.4	3.0	1.3	0.2	0
45	4.8	3.0	1.4	0.3	0
48	5.3	3.7	1.5	0.2	0
49	5.0	3.3	1.4	0.2	0
69	5.6	2.5	3.9	1.1	1
71	6.1	2.8	4.0	1.3	1
73	6.1	2.8	4.7	1.2	1
78	6.0	2.9	4.5	1.5	1
83	6.0	2.7	5.1	1.6	1
85	6.0	3.4	4.5	1.6	1
88	5.6	3.0	4.1	1.3	1
91	6.1	3.0	4.6	1.4	1

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
94	5.6	2.7	4.2	1.3	1
99	5.7	2.8	4.1	1.3	1
103	6.3	2.9	5.6	1.8	2
109	7.2	3.6	6.1	2.5	2
111	6.4	2.7	5.3	1.9	2
112	6.8	3.0	5.5	2.1	2
113	5.7	2.5	5.0	2.0	2
119	6.0	2.2	5.0	1.5	2
127	6.1	3.0	4.9	1.8	2
131	7.9	3.8	6.4	2.0	2
132	6.4	2.8	5.6	2.2	2
140	6.7	3.1	5.6	2.4	2

In [12]:
```
#파일입력
train_dataset.to_csv("./iris_train.csv")
test_dataset.to_csv("./iris_test.csv")
```

In [13]:
```
test_dataset=pd.read_csv("./iris_test.csv")
test_dataset=test_dataset.drop(["Unnamed: 0"],axis=1)
test_dataset
```

Out[13]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	5.7	4.4	1.5	0.4	0
3	4.6	3.6	1.0	0.2	0
4	5.4	3.4	1.5	0.4	0
5	5.5	3.5	1.3	0.2	0
6	4.4	3.0	1.3	0.2	0
7	4.8	3.0	1.4	0.3	0
8	5.3	3.7	1.5	0.2	0
9	5.0	3.3	1.4	0.2	0
10	5.6	2.5	3.9	1.1	1
11	6.1	2.8	4.0	1.3	1
12	6.1	2.8	4.7	1.2	1
13	6.0	2.9	4.5	1.5	1
14	6.0	2.7	5.1	1.6	1
15	6.0	3.4	4.5	1.6	1
16	5.6	3.0	4.1	1.3	1

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
17	6.1	3.0	4.6	1.4	1
18	5.6	2.7	4.2	1.3	1
19	5.7	2.8	4.1	1.3	1
20	6.3	2.9	5.6	1.8	2
21	7.2	3.6	6.1	2.5	2
22	6.4	2.7	5.3	1.9	2
23	6.8	3.0	5.5	2.1	2
24	5.7	2.5	5.0	2.0	2
25	6.0	2.2	5.0	1.5	2
26	6.1	3.0	4.9	1.8	2
27	7.9	3.8	6.4	2.0	2
28	6.4	2.8	5.6	2.2	2
29	6.7	3.1	5.6	2.4	2

In [14]:
```
train_dataset=pd.read_csv("./iris_train.csv")
train_dataset=train_dataset.drop(["Unnamed: 0"],axis=1)
train_dataset
```

Out[14]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	4.7	3.2	1.3	0.2	0
1	4.6	3.1	1.5	0.2	0
2	5.0	3.6	1.4	0.2	0
3	5.4	3.9	1.7	0.4	0
4	4.6	3.4	1.4	0.3	0
...	...	...	...	...	...
115	6.7	3.0	5.2	2.3	2
116	6.3	2.5	5.0	1.9	2
117	6.5	3.0	5.2	2.0	2
118	6.2	3.4	5.4	2.3	2
119	5.9	3.0	5.1	1.8	2

120 rows × 5 columns

In [24]:
```
class neuralNetwork:
    def __init__(self, inputnodes, hiddennodes,outputnodes, learningrate):

        self.inodes = inputnodes
        self.hnodes = hiddennodes
        self.onodes = outputnodes

        self.wih = np.random.normal(0.0,pow(self.hnodes, -0.5),(self.hnodes, self.ino
        self.who = np.random.normal(0.0,pow(self.onodes, -0.5),(self.onodes, self.hno
```

```
        self.lr = learningrate

        self.activation_function = lambda x: scipy.special.expit(x)

        pass

    def train(self, inputs_list, targets_list):
        inputs = np.array(inputs_list, ndmin = 2).T
        targets = np.array(targets_list, ndmin = 2).T

        hidden_inputs = np.dot(self.wih, inputs)
        hidden_outputs = self.activation_function(hidden_inputs)

        final_inputs = np.dot(self.who, hidden_outputs)
        final_outputs = self.activation_function(final_inputs)

        output_errors = targets - final_outputs

        hidden_errors = np.dot(self.who.T, output_errors)

        self.who += self.lr * np.dot((output_errors * final_outputs * (1.0 - final_ou
        self.wih += self.lr * np.dot((hidden_errors * hidden_outputs * (1.0 - hidden_

        pass

    def query(self, inputs_list):
        inputs = np.array(inputs_list, ndmin = 2).T

        hidden_inputs = np.dot(self.wih, inputs)
        hidden_outputs = self.activation_function(hidden_inputs)

        final_inputs = np.dot(self.who, hidden_outputs)
        final_outputs = self.activation_function(final_inputs)

        return final_outputs
```

In [25]:
```
training_data_file = open("./iris_train.csv",'r')
training_data_list = training_data_file.readlines()
training_data_file.close()

test_data_file = open("./iris_test.csv",'r')
test_data_list = test_data_file.readlines()
test_data_file.close()
```

In [26]:
```
#init
input_nodes=4
hidden_nodes=10
output_nodes=3

learning_rate=0.01

n=neuralNetwork(input_nodes, hidden_nodes, output_nodes, learning_rate)
```

In [27]:
```
epochs = 1000
for e in range(epochs):
    for record in training_data_list[1:]:
        all_values =record.split(',')
        inputs=all_values[1:5]
        inputs=list(map(float, inputs))
        targets = np.zeros(output_nodes) + 0.01
        targets[int(all_values[5])] = 0.99
        n.train(inputs, targets)
        pass
pass
```

In [28]:
```
scorecard = []

for record in test_data_list[1:]:
    all_values = record.split(',')
    correct_label = int(all_values[5])
    inputs=all_values[1:5]
    inputs=list(map(float, inputs))
    outputs =n.query(inputs)
    label = np.argmax(outputs)
    print("label is : ", label)
    print("correct label is : ",correct_label)
    if(label == correct_label):
        scorecard.append(1)
    else:
        scorecard.append(0)
        pass
    pass
```

```
label is :  0
correct label is :  0
label is :  0
correct label is :  0
label is :  0
correct label is :  0
label is :  0
correct label is :  0
label is :  0
correct label is :  0
label is :  0
correct label is :  0
label is :  0
correct label is :  0
label is :  0
correct label is :  0
label is :  0
correct label is :  0
label is :  0
correct label is :  0
label is :  1
correct label is :  1
label is :  1
correct label is :  1
label is :  1
correct label is :  1
label is :  1
correct label is :  1
label is :  2
correct label is :  1
label is :  1
correct label is :  1
label is :  1
correct label is :  1
label is :  1
correct label is :  1
label is :  1
correct label is :  1
label is :  1
correct label is :  1
label is :  1
correct label is :  1
label is :  2
correct label is :  2
label is :  2
correct label is :  2
label is :  2
correct label is :  2
label is :  2
correct label is :  2
label is :  2
```

```
correct label is :  2
label is :  2
correct label is :  2
label is :  2
correct label is :  2
label is :  2
correct label is :  2
label is :  2
correct label is :  2
label is :  2
correct label is :  2
```

In [29]:
```
scorecard_array=np.asarray(scorecard)
print("epochs = ", epochs,
      "hidden_nodes = ", hidden_nodes,
      "learning = ", learning_rate,
       "performance = ", scorecard_array.sum()/scorecard_array.size)
```

```
epochs =  1000 hidden_nodes =  10 learning =  0.01 performance =  0.9666666666666667
```

input_nodes의 개수와 output_nodes의 개수를 정한이유

붓꽃의 input의 종류가 꽃받침의 길이, 너비, 꽃잎의 길이와 너비로 4종류였고, output의 종류가 setosa, versicolor, virginica이기 때문에 input_nodes 4개, output_nodes 3개로 결정했다.

202001555 지은미