Отчет по домашней работе.

1. Постановка задачи

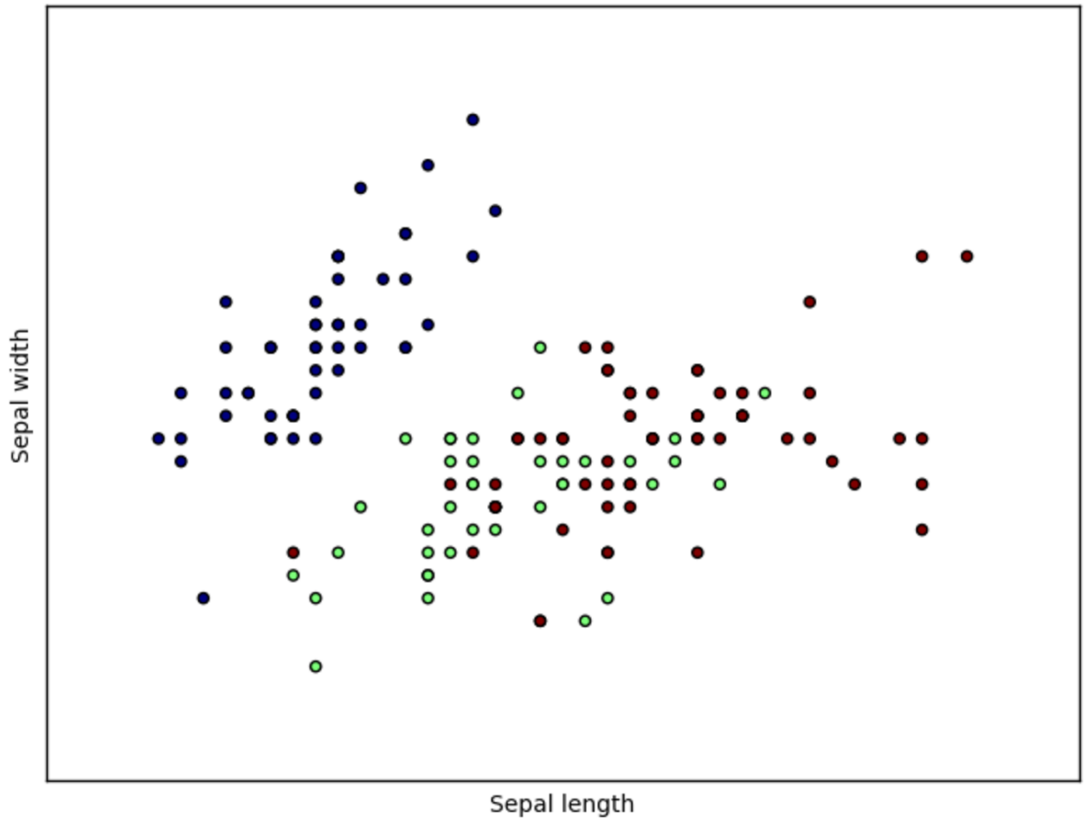
В данной необходимо решить задачу классифицирования вида цветка ириса (Iris-Setosa, Iris-Versicolour, Iris-Virginica) по следующим признакам – длина чашелистиков, ширина чашеристиков, длина лепестка и ширина лепестка, посредствам использования нейронной сети.

1.1. **Данные.** Данный датасет можно найти в питоновской библиотеке – sklearn.datasets с помощью функции load_iris. Датасет состоит за 150, по 50 на каждый из трех типов цветка, каждый пример – массив из четырех чисел и ответ — число от 0 до 2.

```
from sklearn                    import datasets

orig_data = datasets.load_iris()
target = orig_data.target
data   = orig_data.data
```

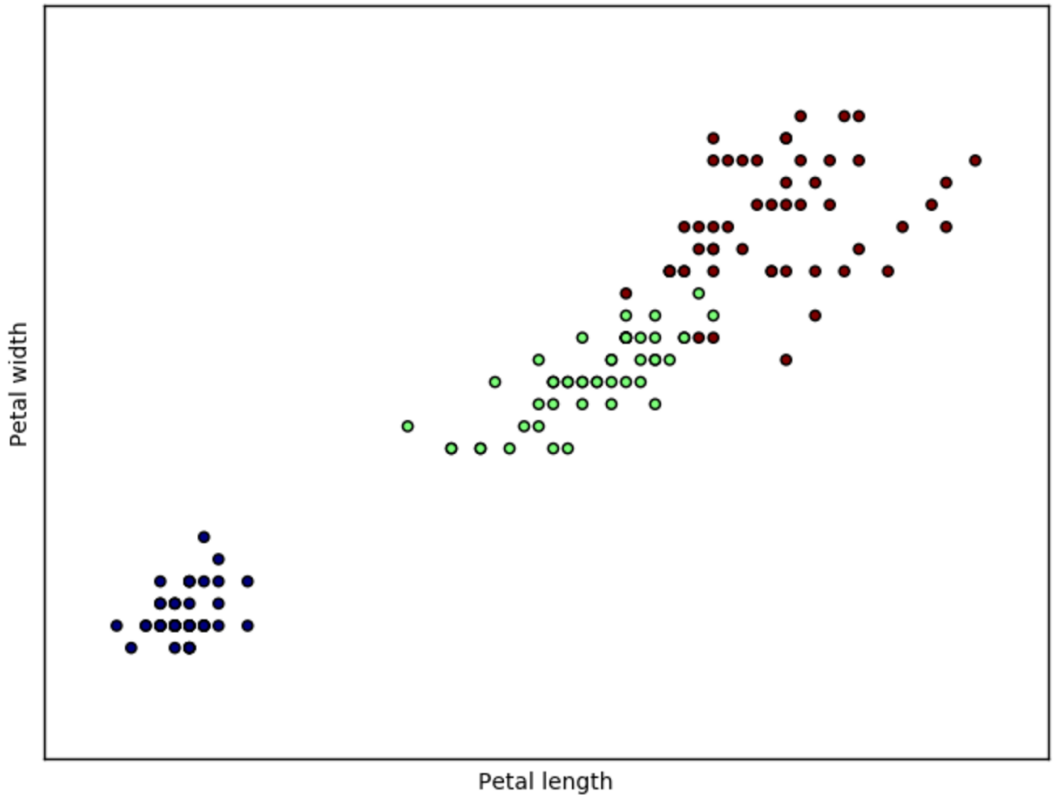
Визуализируем данные:

```
import matplotlib.pyplot as plt
from sklearn import datasets

iris = datasets.load_iris()
X = iris.data[:, :3]
Y = iris.target

x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5

plt.figure(2, figsize=(8, 6))
plt.clf()

plt.scatter(X[:, 0], X[:, 1], c=Y + 1)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')

plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())

plt.show()
```




```
import matplotlib.pyplot as plt
from sklearn import datasets

iris = datasets.load_iris()
X = iris.data[:, 2:]
Y = iris.target

x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5

plt.figure(2, figsize=(8, 6))
plt.clf()

plt.scatter(X[:, 0], X[:, 1], c=Y + 1)
plt.xlabel('Petal length')
plt.ylabel('Petal width')

plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
```

```
plt.yticks(())

plt.show()
```



Видно что по некоторым параметрам ответы датасета линейно разделимы – значит нам хватит сети с одним скрытым слоем и не обязательно делать ее сильно нелинейной.

2. Решение задачи

2.1. **Нейронная сеть.** Для решения задачи было решено использовать библиотеку pybrain. В качестве нейронной сети была выбрана feedforward neural network (сеть прямого распространения сигнала) с одним полносвязным скрытым слоем.

```
from pybrain.datasets         import ClassificationDataSet
from pybrain.tools.shortcuts  import buildNetwork
from pybrain.supervised.trainers import BackpropTrainer
from pybrain.structure.modules import SoftmaxLayer
from pybrain.structure.modules import LinearLayer
from pybrain.structure.modules import SigmoidLayer
from pybrain.structure        import FullConnection
```

```
from sklearn.cross_validation import train_test_split

from sklearn.model_selection  import cross_val_score
from sklearn                  import datasets

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

def convert_supervised_to_classification(data, target):
    num_tr = data.shape[0]

    traindata = ClassificationDataSet(4,1,nb_classes=3)
    for i in range(num_tr):
        traindata.addSample(data[i], target[i])

    traindata._convertToOneOfMany()
    return traindata

def net_model_pybrain():
    network = FeedForwardNetwork()
    inLayer = LinearLayer(4)
    hiddenLayer = SigmoidLayer(1)
    outLayer = SigmoidLayer(3)

    network.addInputModule(inLayer)
    network.addModule(hiddenLayer)
    network.addOutputModule(outLayer)

    in_to_hidden = FullConnection(inLayer , hiddenLayer)
    hidden_to_out = FullConnection(hiddenLayer , outLayer)

    network.addConnection(in_to_hidden)
    network.addConnection(hidden_to_out)

    network.sortModules()
```

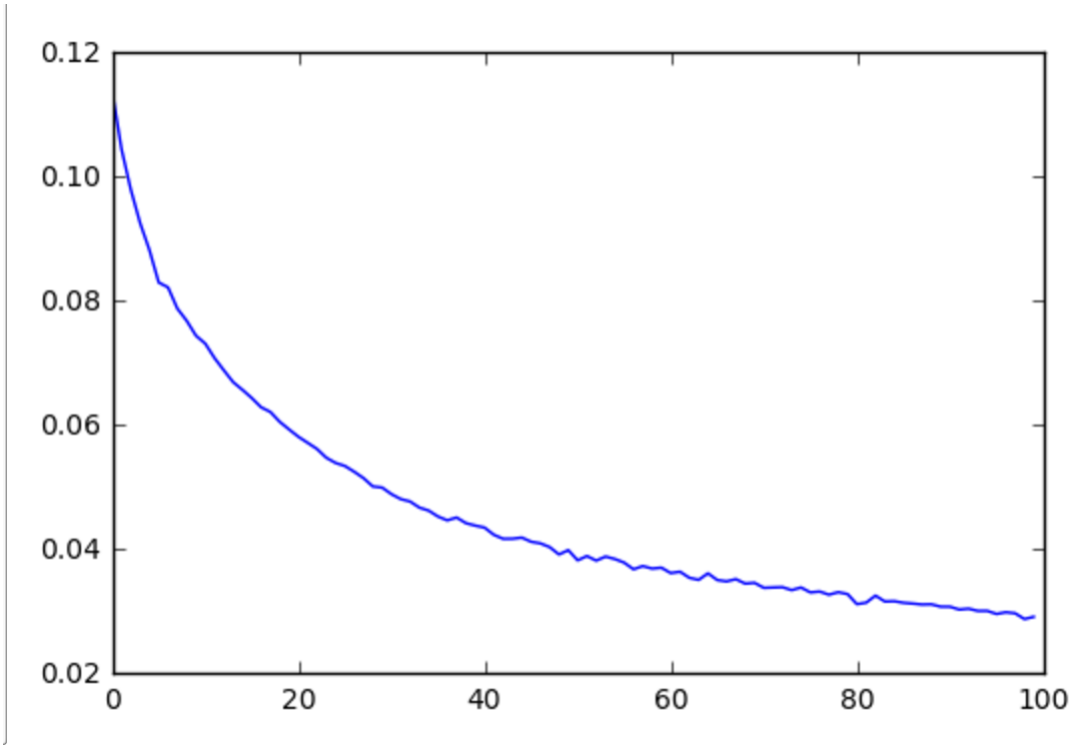
2.2. **Обучение сети.** Для начала создадим сеть

```
traindata = convert_supervised_to_classification(data, target)
network = buildNetwork(4,1,3,outclass=SoftmaxLayer)
trainer = BackpropTrainer(network, dataset=traindata, momentum=0.1, verbose=True)
```

Далее обучим ее на наших данных на 100 эпохах и построим график ошибки.

```
epochs = 100
```

```
from matplotlib import pyplot as plt
%matplotlib inline


x = []
y = []
for i in range(epochs):
    x += [i]
    y += [trainer.train()]

plt.plot(x, y)
```



Из графика видно, что ошибка нейронной сети на данном датасете оказалась, в конченом итоге, достаточно мала, учитывая, что данных было не так уж и много. Так же под конец ошибка начинает стабилизироваться, что говорит о том что алгоритму требуется не так уж и много эпох что бы придти к оптимальной модели.