1-experiment-1-dlvs-lab

August 29, 2024

1 ****1. DESIGN SINGLE UNIT PERCEPTRON FOR CLASSIFICATION OF IRIS DATASET WITHOUT USING PRE-DEFINED MODELS*

[45]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that␣
↪gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved␣
↪outside of the current session
```

```
/kaggle/input/iris-dataset/iris.csv
```

[46]:
```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
```

[47]:
```
iris=pd.read_csv("/kaggle/input/iris-dataset/iris.csv")

iris
```

[47]:

```
     sepal_length  sepal_width  petal_length  petal_width    species
0             5.1          3.5           1.4          0.2     setosa
1             4.9          3.0           1.4          0.2     setosa
2             4.7          3.2           1.3          0.2     setosa
3             4.6          3.1           1.5          0.2     setosa
4             5.0          3.6           1.4          0.2     setosa
..            …            …             …            …        …
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica

[150 rows x 5 columns]
```

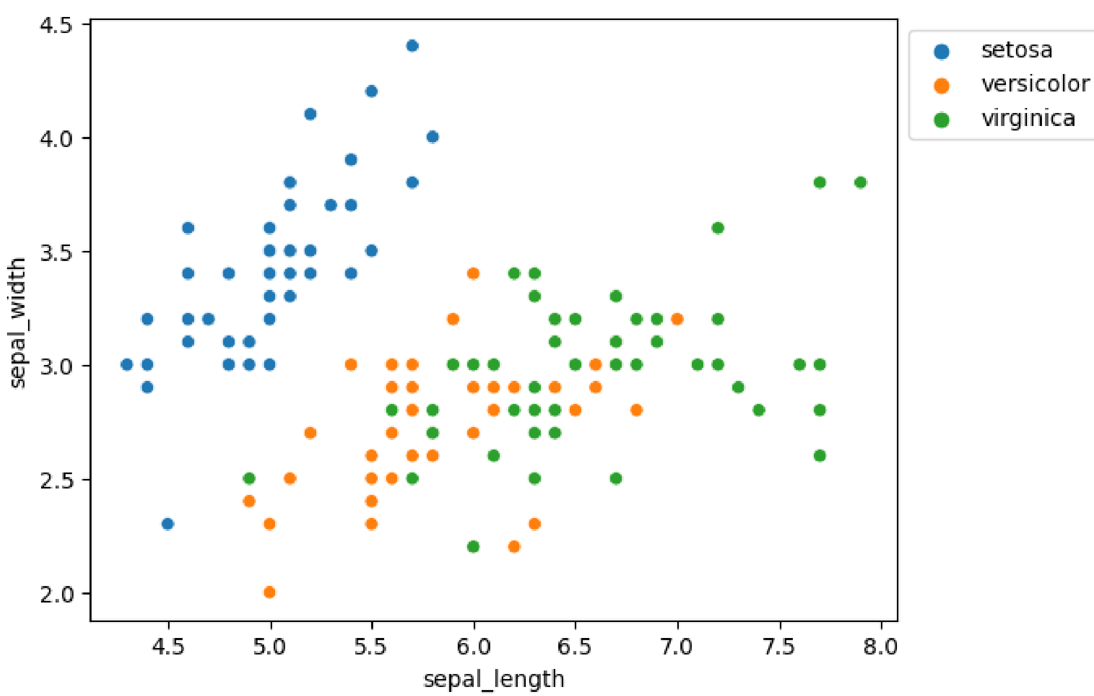
[48]:

```
import seaborn as sns
import matplotlib.pyplot as plt


sns.scatterplot(x='sepal_length', y='sepal_width', hue='species', data=iris, )

# Placing Legend outside the Figure
plt.legend(bbox_to_anchor=(1, 1), loc=2)

plt.show()
```



[49]:
```
iris['species'].unique()
```

[49]:
```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

[50]:
```
iris.groupby('species').size()
```

[50]:
```
species
setosa        50
versicolor    50
virginica     50
dtype: int64
```

[51]:
```
#iris = load_iris()
iris = load_iris()
X = iris.data[:100, :2]  # Use only two features and two classes (Setosa and␣
↪Versicolor)
y = iris.target[:100]
```

[52]:
```
y
```

[52]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

[53]:
```
# Convert labels to -1 and 1
y = np.where(y == 0, -1, 1)

y
```

[53]:
```
array([-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1])
```

[54]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[55]:
```
# Initialize weights and bias
weights = np.zeros(X_train.shape[1])
bias = 0
learning_rate = 0.1
```

```python
epochs = 10
```

[56]:
```python
# Perceptron training
for epoch in range(epochs):
    for i in range(X_train.shape[0]):
        linear_output = np.dot(X_train[i], weights) + bias
        y_pred = np.where(linear_output > 0, 1, -1)

        # Update weights and bias
        if y_train[i] != y_pred:
            weights += learning_rate * y_train[i] * X_train[i]
            bias += learning_rate * y_train[i]
```

[57]:
```python
# Testing the perceptron
correct_predictions = 0
for i in range(X_test.shape[0]):
    linear_output = np.dot(X_test[i], weights) + bias
    y_pred = np.where(linear_output > 0, 1, -1)
    if y_pred == y_test[i]:
        correct_predictions += 1
```

[58]:
```python
accuracy = correct_predictions / X_test.shape[0]
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Accuracy: 100.00%
```