Assignment2

October 13, 2024

[1]:
```
pip install ucimlrepo
```

```
Collecting ucimlrepo
  Downloading ucimlrepo-0.0.7-py3-none-any.whl.metadata (5.5 kB)
Requirement already satisfied: pandas>=1.0.0 in /usr/local/lib/python3.10/dist-
packages (from ucimlrepo) (2.2.2)
Requirement already satisfied: certifi>=2020.12.5 in
/usr/local/lib/python3.10/dist-packages (from ucimlrepo) (2024.8.30)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.0.0->ucimlrepo) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0->ucimlrepo) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.0.0->ucimlrepo) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.0.0->ucimlrepo) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.2->pandas>=1.0.0->ucimlrepo) (1.16.0)
Downloading ucimlrepo-0.0.7-py3-none-any.whl (8.0 kB)
Installing collected packages: ucimlrepo
Successfully installed ucimlrepo-0.0.7
```

[2]:
```python
import matplotlib.pyplot as plt
import random
import math

from ucimlrepo import fetch_ucirepo

# fetch dataset
iris = fetch_ucirepo(id=53)

# data (as pandas dataframes)
features = iris.data.features
targets = iris.data.targets
```

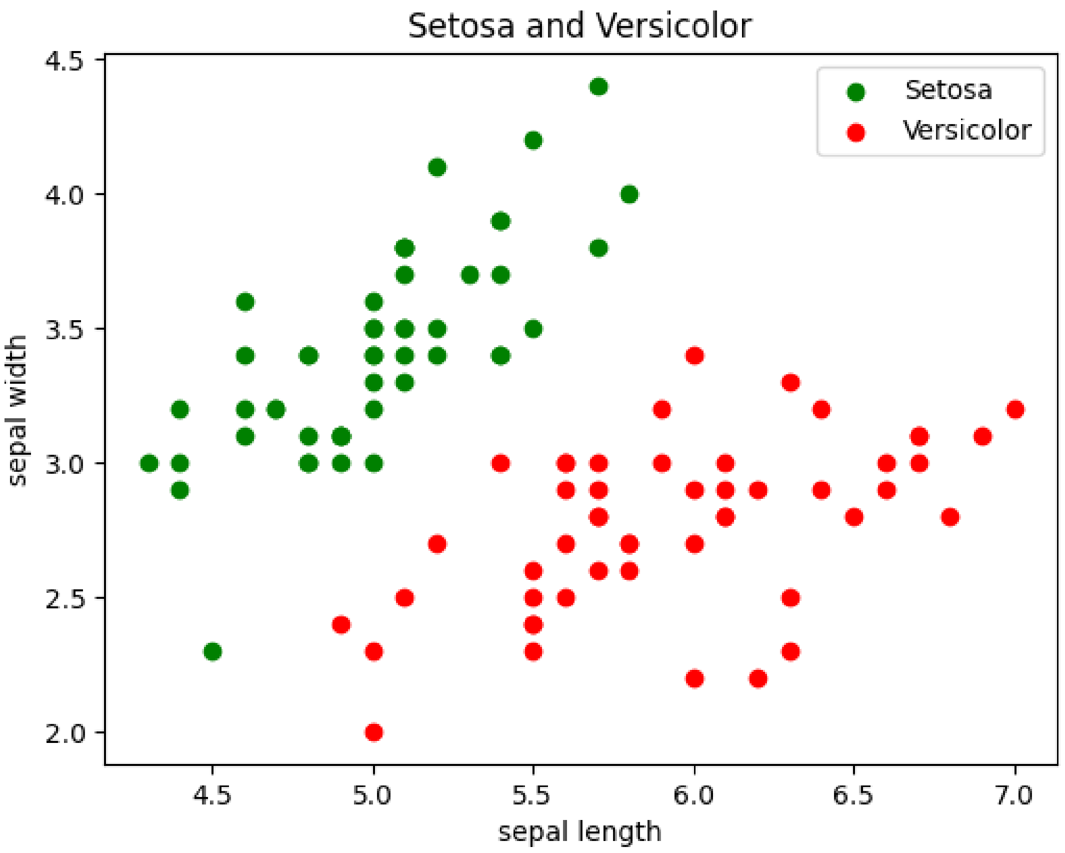
[3]:
```python
# Question 3
# extract sepal length and sepal width data from dataset for setosa and␣
↪versicolor
```

```
sepLSet = features['sepal length'].values[0:50]
sepWSet = features['sepal width'].values[0:50]
sepLVer = features['sepal length'].values[50:100]
sepWVer = features['sepal width'].values[50:100]

# create combined input lists
input1s = sepLSet.tolist() + sepLVer.tolist()
input2s = sepWSet.tolist() + sepWVer.tolist()

# plot sepal length/width graph
plt.title("Setosa and Versicolor")
plt.xlabel("sepal length")
plt.ylabel("sepal width")
setosa = plt.scatter(sepLSet, sepWSet, c='g', label="Setosa")
versicolor = plt.scatter(sepLVer, sepWVer, c='r', label="Versicolor")
plt.legend(handles=[setosa, versicolor])
```

```
[3]: <matplotlib.legend.Legend at 0x7d4b85e90610>
```



[4]:
```
# Question 5
# model function
def logistic(w, i):
    return 1/(1+math.exp(-(w[0]+w[1]*i[0]+w[2]*i[1])))

# update rule (stochastic)
def update(pred, i, y): # loop will calculate prediction, which is used to␣
 ↪update all 3 weights
    weights[0] = weights[0]+l_rate*(y-pred)*pred*(1-pred)*1
    weights[1] = weights[1]+l_rate*(y-pred)*pred*(1-pred)*i[0]
    weights[2] = weights[2]+l_rate*(y-pred)*pred*(1-pred)*i[1]

# iterate over all training samples
def epoch():
    success_count = 0   # setosa is success, probability > .5
    for ind in train_ind:
      inputs = [input1s[ind], input2s[ind]]
      it_pred = logistic(weights, inputs)
      success = 0
      if (it_pred < .5 and ind > 49): # first 50 elements are setosa (success)
        success_count += 1
      elif (it_pred >= .5 and ind <= 49):
        success_count += 1
      else:  # update weights if not a success
          update(it_pred, inputs, 0 if ind>49 else 1)

    return success_count/len(train_ind)

# run num_ep epochs and report accuracy rate
def train(num_ep):
    for i in range(num_ep):
        success_count_list.append(epoch())

# plot accuracy rate across all epochs
def report():
    print("Final weight vector:", weights)
    x = range(1, len(success_count_list)+1)
    plt.title("Training Accuracy")
    plt.xlabel("Epoch")
    plt.ylabel("% Correct")
    plt.xticks(x)
    plt.plot(x, [x * 100 for x in success_count_list])

# determine accuracy rate for test data
def test():
    test_success_count = 0
    for ind in test_ind:
        setosa = 1
        if (ind > 49): setosa = 0
        inputs = [input1s[ind], input2s[ind]]
        test_pred = logistic(weights, inputs)
        if (test_pred >= .5 and setosa == 1): test_success_count+=1
        if (test_pred < .5 and setosa == 0): test_success_count+=1
    return test_success_count/len(test_ind)
```

[5]:
```
# Question 4
# randomly assign indices of samples in training batch and test batch
train_ind = []
while len(train_ind) < 80:
  r = random.randint(0,99)
  if r not in train_ind:
    train_ind.append(r)

all_ind = list(range(0,100))
s = set(train_ind)
test_ind = [num for num in all_ind if num not in s]

print("Indices in training set:", train_ind)
print("Indices in test set:", test_ind)
```

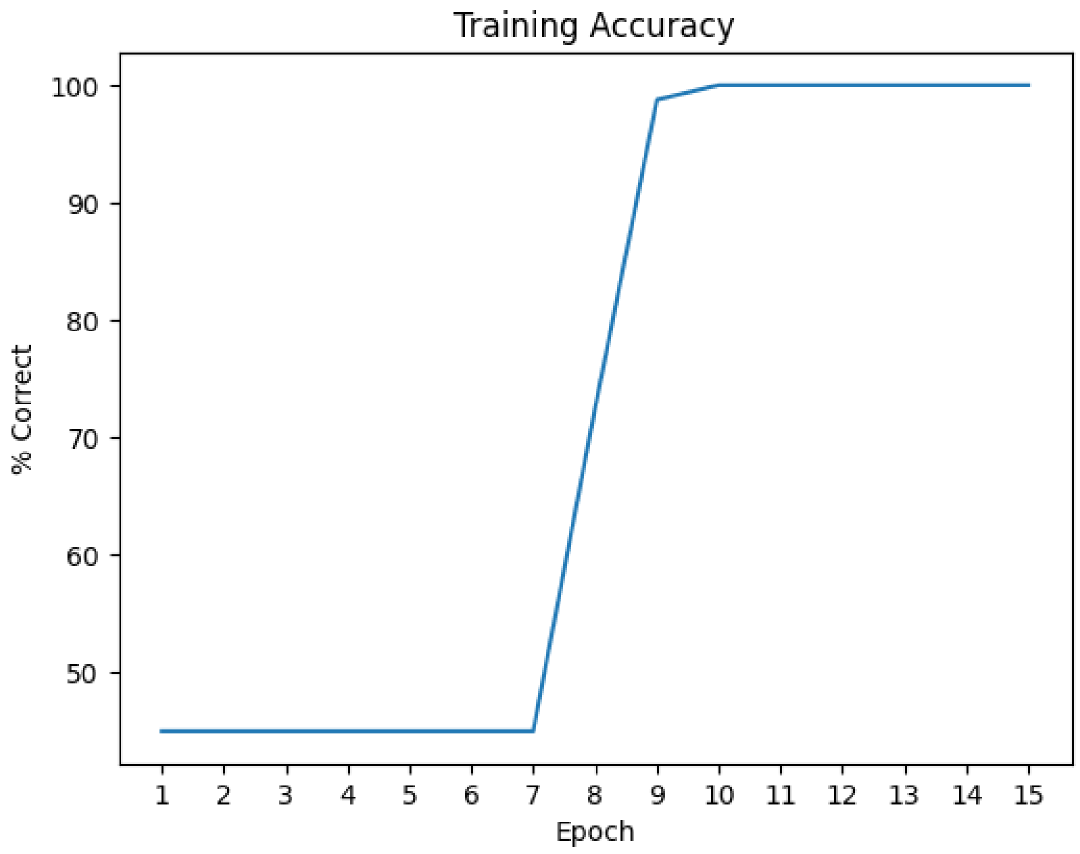
```
Indices in training set: [38, 70, 72, 13, 31, 36, 32, 91, 98, 11, 54, 60, 20,
84, 61, 39, 59, 87, 58, 40, 43, 56, 68, 62, 46, 14, 97, 45, 7, 6, 2, 80, 88, 86,
85, 17, 19, 49, 79, 22, 93, 53, 29, 74, 99, 5, 69, 92, 34, 51, 81, 63, 16, 64,
8, 78, 73, 25, 52, 75, 44, 50, 4, 15, 67, 94, 35, 42, 65, 90, 0, 76, 89, 23, 12,
82, 66, 37, 33, 96]
Indices in test set: [1, 3, 9, 10, 18, 21, 24, 26, 27, 28, 30, 41, 47, 48, 55,
57, 71, 77, 83, 95]
```

[6]:
```
# Question 6
# train and report accuracy across epochs

# initialization
weights = [.5,.5,.5]
l_rate = .01
success_count_list = []

# training
print("Initial weight vector", weights)
train(15)
report()
print("Test accuracy: " + str(test()*100) + "%")
```

```
Initial weight vector [0.5, 0.5, 0.5]
Final weight vector: [0.38392006603391177, -0.181159720638634,
0.18301812309706772]
Test accuracy: 95.0%
```



Summary: I trained my model to classify a sample as either Setosa or not Setosa (Versicolor). The features used as inputs were sepal length and sepal width. An output of 0.5 or greater from the logistic regression model was treated as a positive identification of a Setosa sample.