PART 1: Neural Network Warm Up Problem

In [1]:
```
import numpy as np
import pandas as pd
import random
import math
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report
```

In [2]:
```
def generate_samples(size):
    columns = ['label', 'x1', 'x2']
    data = []
    count = 0
    while (count < size):
        # generate random x1, x2
        x1 = random.uniform(-1,1)
        x2 = random.uniform(-1,1)
        label = 0
        ignore = False
        count = count+1

        # assign the labels
        if(0.4 < (abs(x1) + abs(x2)) < 0.7):
            label = 0
        elif(math.sqrt(x1*x1 + x2*x2) < 0.3):
            label = 1
        elif(math.sin(10*x2) < 0):
            label = 2
        elif(math.sin(5*x2) > 0):
            label = 3
        else:
            count = count-1
            ignore = True

        if(ignore != True):
            data.append([label, x1, x2])

    return pd.DataFrame(data = data, columns = columns)
```

Generate samples for train, validation and test

In [8]:
```
df_train = generate_samples(500)
df_validation = generate_samples(500)
df_test = generate_samples(1000)
```

Releative fequency of each class in test data

In [21]:
```
df_tmp = df_train.label.value_counts()
print('Relative Frequency in training data\n',df_tmp / len(df_train.label),
```

```
Relative Frequency in training data
 2    0.546
0    0.238
3    0.148
1    0.068
Name: label, dtype: float64
```

Training a neural network with one hidden layer

In [5]:
```
X_train = df_train[["x1","x2"]]
y_train = df_train.label
X_validation = df_validation[["x1","x2"]]
y_validation = df_validation.label

alphas = np.logspace(-6, -2, 5)

# collection of classifiers based on parameter selection
classifiers = []

for alpha in alphas:
    classifiers.append(MLPClassifier(solver='adam', alpha=alpha, hidden_lay
                                     max_iter=3000, random_state=1))
    classifiers.append(MLPClassifier(solver='lbfgs', alpha=alpha, hidden_la
                                     max_iter=3000, random_state=1))

# Iterate over classifiers to find the score using validation data
best_clf = classifiers[0]
best_score = 0
for clf in classifiers:
    clf.fit(X_train, y_train)
    score = clf.score(X_validation, y_validation)
    if(score > best_score):
        best_score = score
        best_clf = clf
    print(clf, 'score=', score)
```

```
MLPClassifier(alpha=1e-06, hidden_layer_sizes=[500], max_iter=3000,
              random_state=1) score= 0.94
MLPClassifier(alpha=1e-06, hidden_layer_sizes=[500], max_iter=3000,
              random_state=1, solver='lbfgs') score= 0.94
MLPClassifier(alpha=1e-05, hidden_layer_sizes=[500], max_iter=3000,
              random_state=1) score= 0.94
MLPClassifier(alpha=1e-05, hidden_layer_sizes=[500], max_iter=3000,
              random_state=1, solver='lbfgs') score= 0.956
MLPClassifier(hidden_layer_sizes=[500], max_iter=3000, random_state=1) sc
ore= 0.942
MLPClassifier(hidden_layer_sizes=[500], max_iter=3000, random_state=1,
              solver='lbfgs') score= 0.938
MLPClassifier(alpha=0.001, hidden_layer_sizes=[500], max_iter=3000,
              random_state=1) score= 0.94
MLPClassifier(alpha=0.001, hidden_layer_sizes=[500], max_iter=3000,
              random_state=1, solver='lbfgs') score= 0.942
MLPClassifier(alpha=0.01, hidden_layer_sizes=[500], max_iter=3000,
              random_state=1) score= 0.936
MLPClassifier(alpha=0.01, hidden_layer_sizes=[500], max_iter=3000,
              random_state=1, solver='lbfgs') score= 0.942
```

In [10]:
```
best_clf
```

Out[10]:
```
MLPClassifier(alpha=1e-05, hidden_layer_sizes=[500], max_iter=3000,
              random_state=1, solver='lbfgs')
```

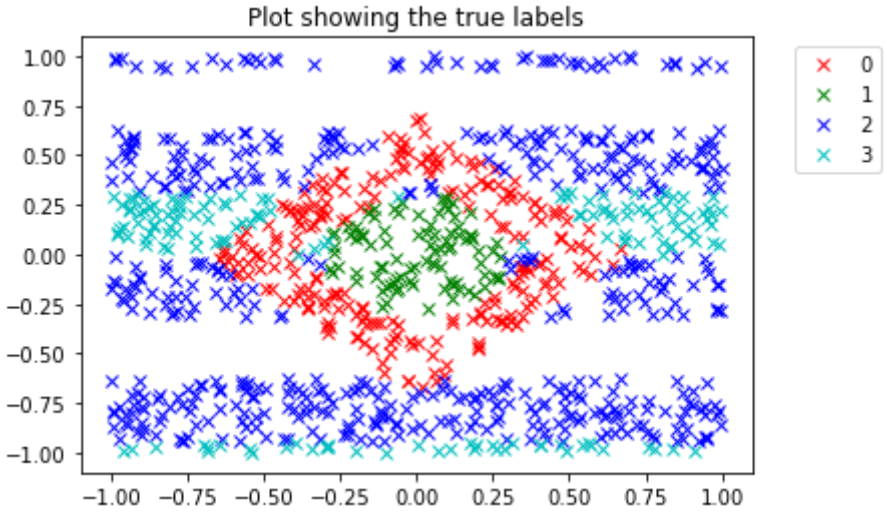
overall accuracy of the classifier on the test data

In [51]:
```
X_test = df_test[["x1","x2"]]
y_test = df_test.label
y_pred = best_clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

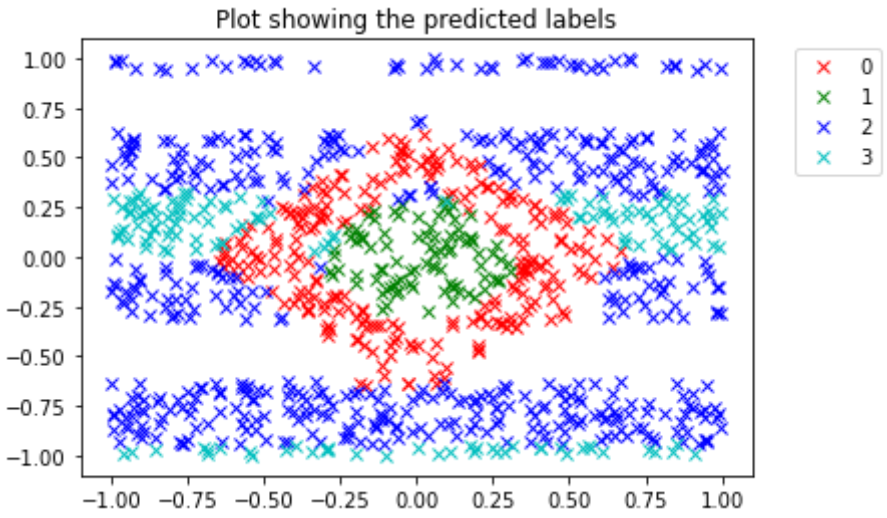
```
              precision    recall  f1-score   support

           0       0.93      0.97      0.95       232
           1       0.98      0.97      0.97        92
           2       0.98      0.97      0.97       522
           3       0.95      0.94      0.95       154

    accuracy                           0.96      1000
   macro avg       0.96      0.96      0.96      1000
weighted avg       0.96      0.96      0.96      1000
```

two plots of the test samples

In [56]:
```
PlotIt(X_test.to_numpy(), y_test, 'Plot showing the true labels')
```



In [57]:
```
PlotIt(X_test.to_numpy(), y_pred, 'Plot showing the predicted labels')
```



In [55]:
```
import matplotlib.pyplot as plt
def PlotIt(data, labels, title):
    xs = data[:, 0]
    ys = data[:, 1]
    colors = ['r', 'g', 'b', 'c']
    for i in range(4):
        idx = labels == i
        plt.plot(xs[idx], ys[idx], 'x', color=colors[i], label = i)
    plt.title(title)
    plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.show()
```