Lab Exercise 6 – Machine Learning

Objectives:

- Familiarization with the application of machine learning techniques to real data.

Introduction:

This practical introduces you to the application of machine learning techniques. The goal is to understand the basics of machine learning and apply this knowledge to sensor data.

```
We will be using the  scikit-learn (http://scikit-learn.org/) machine
learning library for python in this lab.

VM / MacOS users can install scikit-learn using:

$ pip install --user  scikit-learn

To install pip:
$ sudo apt-get install python-pip python-dev build-essential
$ sudo pip install –upgrade pip

For other installations (including Windows) refer:
http://scikit-learn.org/stable/install.html

Please note that Scikit-learn requires:
Python(>=2.7 or >=3.3)
NumPy (>=1.8.2)
SciPy(>=0.13.3)

Please make sure your system meet those requirements:
To install NumPy, please follow:
https://askubuntu.com/questions/509623/installation-procedure-for-
numpy-and-other-python-3-4-packages-on-ubuntu-14-04
Or:
https://docs.scipy.org/doc/numpy-1.10.1/user/install.html

To install SciPy, please follow:
https://www.scipy.org/install.html


Once installed test the installation in a python shell by importing:

>>> from sklearn.svm import SVC

which should not give any errors.
```

Machine Learning (3 Marks)

In this experiment, we will explore a few classification algorithms with supervised learning on a standard dataset. The sample python script 'standard-classification-sample.py' demonstrates a linear SVM based binary classifier. Here is a quick run through on what is implemented in the sample script.

- The function train_test_split() is used to split the dataset in to training (data1, target1) and testing (data2, target2) sets.
  http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.train_test_split.html

- In the tune_svm_lin() function, we use the GridSearchCV() class
  http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html
  to perform a grid search over the regularization parameter for values 'C': [0.1, 1, 10, 100].
  Invoking the fit() method performs the gridsearch and retains the best classifier which can will be used when the predict() method is called (in compute_accuracy() in the sample script). It also performs cross validation by default.
  Here we pass an SVC object with a linear kernel to GridSearchCV().
  http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Run the sample python script with the verbosity set to 3 or 4 and observe how the linear svm is tuned and performance is printed. You can set verbosity back to 1 later on.
Based on this and the scikit API documentation, modify a copy of the sample script to do the following.

NOTE: When you create classifier instances, make sure you pass random_state=rseed to produce repeatable results.

1. Complete the function stub tune_knn () to train a k-nearest neighbor classifier.
   It should perform a grid search to tune the following parameter(s) to and return the best classifier.

   'n_neighbors': [1, 2, 5, 10]

   (Hint: Look at the sample function tune_svm_lin() which has already been done for you)
   Refer:
   http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
   a. Uncomment line 103 to train a classifier clf_knn on the training set using this function.
   b. Complete the end of the script using the print_accuracy() function to print the classification accuracy of clf_knn on the test and training data.
   c. Is the accuracy higher on the test set or the training set? Why?

2. Complete the function stub tune_decision_tree () to train a DecisionTreeClassifier().
   http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
   It should perform a grid search to tune the following parameter(s) to and return the best classifier.

   'max_depth': [1, 10, 100],
   'max_features': ['auto', 1, 3, 30],

   (Hint: Look at the sample function tune_svm_lin() which has already been done for you)

   a. Use this function to train a classifier clf_decision_tree on the training data similar to line 103.
   b. Complete the end of the script using the print_accuracy() function to print the classification accuracy on the test and training data.
   c. Is the accuracy higher on the test set or the training set? Why?

3. Complete the function stub tune_random_forest() to train a RandomForestClassifier().
   http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
   It should perform a grid search to tune the following parameter(s) to and return the best classifier.

   ```
   'max_depth': [1, 10, 100],
   'max_features': ['auto', 1, 3, 30],
   'n_estimators': [2, 10, 100]
   ```

   (Hint: Look at the sample function tune_svm_lin() which has already been done for you)
   a. Use this function to train a classifier clf_random_forest on the training data similar to line 103 for the linear svm.
   b. Use the print_accuracy() function to print the classification accuracy on the test and training data.
   c. Is the accuracy higher on the test set or the training set? Why? How does having an ensemble of trees compare to a single decision tree in Q2?

Brightness-based Localization (2 Marks)

The brightness of an indoor environment varies with the location. The brightness signatures obtained from a light sensor can thus be used to identify and locate a device. In this section you will implement a localization model to locate your laptop (a bright location such as near your monitor) or other electronic devices (e.g. tablet) using the sensor tag's thermopile sensor.

1. Collect two sets of 600 samples from the light sensor (OPT3001) at 10Hz; one with the sensor near a bright location and the other set from a dark location. Save the samples in CSV format. Refer to the sensor-interface example in contiki-examples/humidity sensor. You may choose to transmit samples with either UDP directly or CoAP.

2. Train different classifiers on this data tuning parameter values as in the 'Machine Learning' question. The sample file 'sensortag_sample.py' shows how the features X and targets y can be build using readings from CSV files.

   a. Split the concatenated dataset (1,200 samples) in to training (data1, target1) and testing (data2, target2) sets.
   b. Train different classifiers on (data1, target1) and tune parameter values as in the 'Machine Learning' question.
   c. Use the tuned classifiers obtained in step b) to predict the target values based on data2, and print out the classification accuracy.

Marking Criteria (5 Marks)

You must demonstrate the following criteria and be able to answer questions.

1. **Machine Learning (3 Marks)**

   Implement and demonstrate tuning of each classifier using a gridsearch on the training set and printing performance on the test and training sets. (One mark for each classifier)

2. **Thermal Localization (2 Marks)**
    - Generate two reading files of 600 samples each for the bright and dark location (**1 mark)**
    - Training and tuning different classifiers (the three different classifiers you used in question 1) and demonstrate their accuracies on testing and training data. (**1 mark**)

References:

- http://scikit-learn.org/
- http://docs.python.org/2/library/json.html
- http://matplotlib.org/users/pyplot_tutorial.html

Note: All Python code should be run in the VM