Session 8: Classifier 1 - Support Vector Machine

Objectives

- Learn the basics of SVM based classifiers
- Train a SVM classifier and test its accuracy

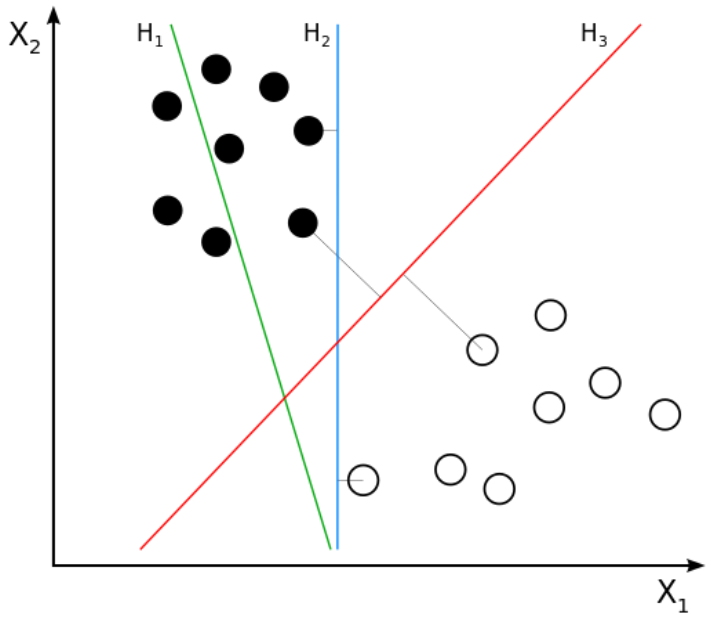
Classification

(ref: https://en.wikipedia.org/wiki/Support_vector_machine)

In machine learning and statistics, classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known.

Support Vector Machine

A support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class.

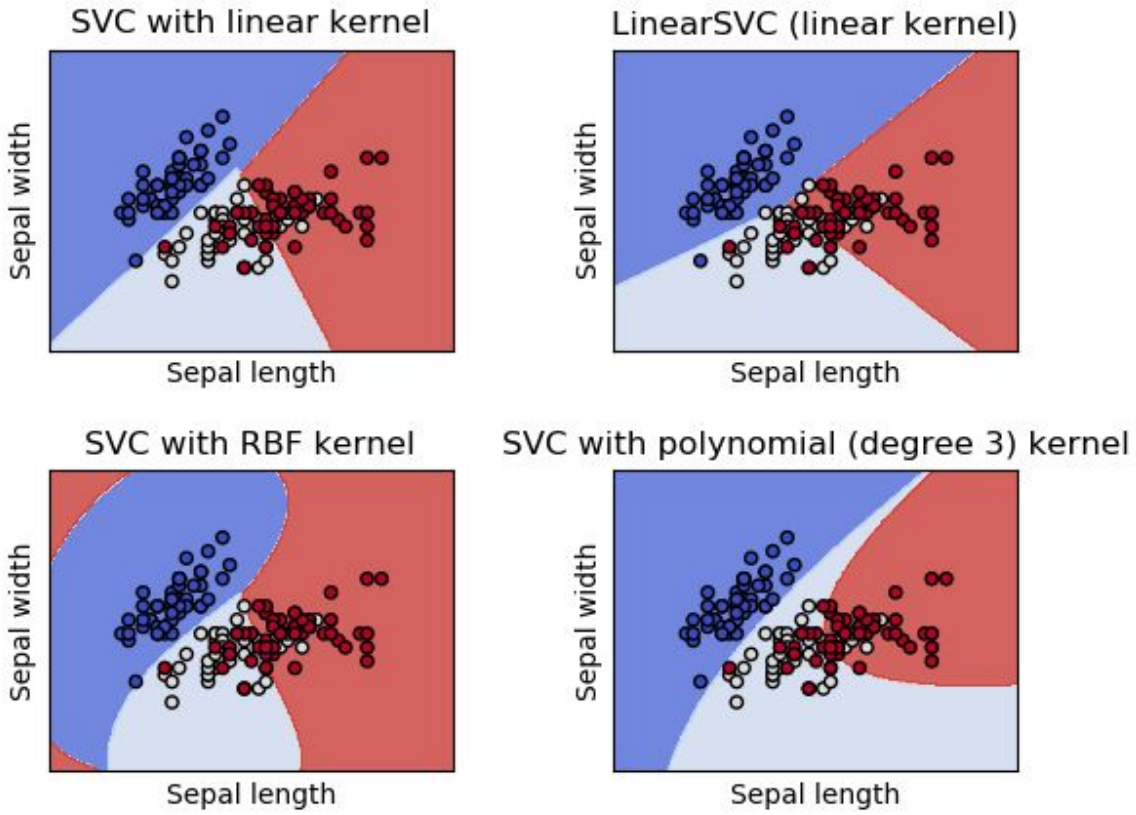


Feature Engineering

For our image classification application, X_1 and X_2 in the above illustration are features extracted from images, e.g. density of bright pixels, number of edge pixels etc. The vector (x_1,x_2) is a feature vector and the space they occupy is known as the feature space. Feature engineering refers to the process of choosing the features to extract/modify so that the feature space is separable.

Kernel options in sklearn SVM library

The kernel used in SVM determines the shape of the boundaries. In the sklearn SVM library, we have several choices for the kernels, see http://scikit-learn.org/stable/modules/svm.html.



Exercise 8-0: Prepare training and testing data

Go to folder ex-8-0-prepdata_classification. Here, you will find a set of pre-collected 50x50 pixel images that are sorted into three classes:

- No sign (class 0)
- Stop sign (class 1)
- Warn/Yield sign (class 2)

TO-DO:
Run `prepdata.py` to generate pickle file data.
`python prepdata.py`
This assembles all the images into two python pickle file `data.p` and `test.p`, which will be used by the main program `signs-svm.py` (and for the next session on Convolutional Neural Networks)

Exercise 8-1: Train a SVM classifier and verify its accuracy

Open the folder ex-8-1-SVM-Train.
TO-DO:
Copy the generated `data.p` and `test.p` files into working folder `scm-train-ex`.
`cp ../../ex-8-0-prepdata_classification/*.p .`

Study the code in `signs-svm.py`, which trains a SVM classifier and verifies the accuracy of the model. The main heavy lifting of SVM training is done by sklearn library:

```
clf = svm.SVC()
clf.fit(normFeatures,y_train)
```

Check out http://scikit-learn.org/stable/modules/svm.html to find out more about the various options of the sklearn SVM library.

TO-DO:
Choose and optimize a set of features in the function `getFeatures()` such that the training data can be effectively separately by the SVM classifier, and maximize the test accuracy.

Exercise 8-2: Video Pipeline

Open the folder ex-8-2-Pipeline.

In this exercise, we will implement a sliding window search algorithm that search for a traffic sign (and decide what it is) in each frame of a video file by performing classification on each 50x50 pixels window at a time. The main sliding window algorithm is in python class `SVMCLF` in the file `clf_svm.py`.

TO-DO:
Edit `getFeatures()` to use the same feature vector that was used for training.

TO-DO:
Edit `get_decision()` to experiment with different decision making rules based on the individual scores and thresholds defined in the constructor. Study the rest of the code to see how the scores are defined and change them if you want.

TO-DO:
Edit the search thresholds in the constructor `__init__()` to fine-tune the classification performance.

Check that the results are ok in the output video file '`result_test_video_frames.mp4`'.

Note: To play a video on the Raspberry Pi, navigate to the location of your video file in the terminal using cd, then type the following command:
`omxplayer example.mp4`