University of California
Department of Civil Engineering

Pozdnukhov, Instructor
Fall 2015

# CE 263 Problem #4

## Pattern Recognition

In this task you will build a classifier to recognize cars in the typical aerial images. Such pattern recognition systems have a wide scope of applications ranging from urban space management to emergency response, where large volumes of remotely sensed imagery data have to be processed automatically.

Python is the recommended programming language for this assignment. You are encouraged to use a Support Vector Machine implementation of Scikit-learn toolkit (http://scikit-learn.org/stable/modules/svm.html#classification), or any other implementation of the SVM.

**The data.**

**Training:** The dataset provided to you in this assignment problem is an image of a typical parking in the Bay Area in PNG format (**parking_train.png**). There are 4 bands in the image (R,G,B,T). The T band is a transparency mask that does not carry any information in this example. You are provided with a python script (**clicker.py**) to read the image and collect the dataset of car and non-car samples in order to train and tune the parameters of your classifier. The samples are collected interactively at a mouse click as image patches centered at the location of the click (left click is "car" and right click is "not a car" class. Mac users be aware of how the 'left' and 'right' click events are set up for your touchpad). No other feature selection is performed, and you may consider implementing a more sophisticated scheme, however, it is not necessary to complete this assignment. The samples are then saved to disk as NumPy arrays.

**Testing:** The performance of your classifier has to be demonstrated on the testing dataset (a set of samples that were not used for training nor tuning the hyper-parameters of the algorithm). A part of the testing image is provided to you (**parking_test_preview.png**) as well as an example set of locations (**test_locations_and_labels_preview.np**), however, the exact testing image and the sample locations for testing are not available prior to your submission and will be released only after the deadline.

**The Scoring System** (max available points: **112.5**)

You will be given 2 points for every car classified correctly and subtracted 0.5 points for every non-car classified by your method as a car (i.e. for every "false alarm") as well as for every car missed. Each correctly identified non-car brings you 0.25 points. There are 100 pre-set locations within the testing image (50 cars and 50 non-cars) that your method will be tested on.

**Submission requirements**

Your submission has to be a python script that reads a set of locations in the image, performs feature selection, and applies the classifier to the samples in the testing set. An example of such script is provided for your reference (**submission_YourName.py**). You may provide a short report describing your approach but it is not required.

**The Classifier.**

You can submit a saved (pickled) copy of your pre-train classifier, for example, with some dataset X and Y:

```
from sklearn import svm

clf = svm.SVC(C=100.0)
clf.fit(X, Y)

with  open('classifier_YourName.pickle','wb') as f:
  pickle.dump(clf, f)
```

Your classifier **must** have the predict function:

```
clf.predict(X[0])
```

The way it will be applied to the testing set is exactly as shown in lines 33-54 of (**submission_YourName.py**).

Happy Car Spotting!

May your final score be over 100!