HomeWork Assignment 5

Comp 590.133

8th April, 2014

Due: 11:59 pm, 22nd April 2014

Getting Started

What to submit: All the code you implemented plus descriptions of your implementation and experiments in a file called answers.pdf (preferably in pdf format but Word or ascii are ok too.) Please also provide a short description on how to run your code in a README file.

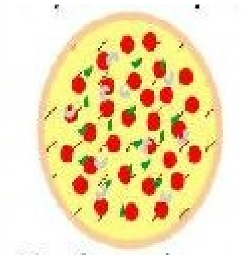
Where to submit: Upload the document(s) and all your python code files (along with a README) to */afs/cs.unc.edu/project/courses/comp590-133-s14/students/**<your-login>**/HW5* directory by logging into classroom/snapper nodes. Where **<your-login>** should be replaced by your CS login. (The same way you submitted homework 1,2,3,4.)

Evaluation: Your code will be tested and manually-graded for technical correctness. Written parts will also be manually checked.

Academic Dishonesty: We will be checking your code against other submissions in the class for logical redundancy. We trust you all to submit your own work only; *please* don't let us down. If you do, we will pursue the strongest consequences available to us.

Getting Help: You are not alone! If you find yourself stuck on something, contact the course staff for help. If you can't make our office hours, let us know and we will schedule more. We want these projects to be rewarding and instructional, not frustrating and demoralizing. But, we don't know when or how to help unless you ask. One more piece of advice: if you don't know what a variable does or what kind of values it takes, print it out. Also, we encourage you to work in a team of size 2.

Image Classification



In this assignment you will be implementing five-way Image Classification using SVMs and a simple "tiny image" representation.

We will be using the libSVM package (with python interface). First download and install libSVM from here: http://www.csie.ntu.edu.tw/~cjlin/libsvm/. Functions from this package that you will need to use are svm_train and svm_predict.

Data: images.tar.gz (source: Caltech 101)

To open this file under mac or linux, use the command: "tar zxvf images.tar".

The data contains 5 image classes: dollar bill (images in the dollar_bill directory), pizza (images in the pizza directory), soccer ball (images in the soccer_ball directory), dalmatian (images in dalmatian directory), sunflower (images in sunflower directory).

Divide the provided dataset into 2 parts. For each class, randomly select 70% of the images to use for training and 30% to use for testing.

a. Computing Features. (5 points)

As your image representation you will be using a simple "tiny image" representation. To form your representation for these RGB images, you should first resize each image to icon size 32x32x3 and then concatenate the image pixels into one long vector (length 3072).

You can use Python Imaging Library (PIL), available for python 2.7, to compute this representation. Instructions on how to download and install PIL are available here: http://www.pythonware.com/products/pil/

b. Training. (5 points)

You should train SVMs to recognize images from each of the given classes independently. To do this, train 5 one-vs-all SVMs (one SVM for each class). One-vs-all models perform binary classification to differentiate between examples from a class and examples that are not from that class. This means that for example, when training the "dalmatian" SVM, positive examples should come from the "dalmatian" training set and negative examples should come from the training sets of other classes. You should also use the "-b" option to obtain a probability from your SVM models.

You should also try training two kinds of SVMs, linear kernel SVMs and RBF kernel SVMs. See documentation on how to use different kernel types in libSVM.

As a part of the training process you should estimate good parameters of your models. You should implement this yourself rather than using any built in libSVM functionality for setting parameters. To find good parameter settings, split your training set for each category in half. Use one half of the training set to train an SVM with a particular setting of the parameters and the other half to evaluate this SVM under that parameter setting. For linear SVMs your only parameter is C (cost parameter). For RBF SVMs your parameters are C and g (gamma parameter of the gaussian). For each class, search over a reasonable range of values for each parameter and select the parameter values with highest performance on the evaluation set (you should experimentally determine what a reasonable range means for this data). Train your final model for each class using all samples from the training set using the pre-selected parameter values.

Report performance on the evaluation set for each class for the different parameter settings you tried.

c. Testing. (5 points)

We evaluate model performance using images in the test set. To predict a class for a test image you should evaluate it under each of the 5 trained models and take the argmax over classes (i.e. label the image as the class with highest predicted probability).

In your write-up, report classification performance for each class (percentage of images from that class that were correctly predicted to depict that class), overall classification performance (performance for each class averaged over the 5 classes), and a confusion matrix M (where M_{ij} is the number of images of class i that were classified as class j).

You should expect classification performance to be (significantly) above chance.

d. Writeup. (5 points)

Submit in a file called answers.pdf:

1) A description of your implementation of each of the above steps.

2) Descriptions of:

- Performance on the evaluation set and how this was used to select good parameter values for both linear and RBF SVMs.
- Classification accuracies achieved by your model on the test set for each category, overall performance across classes, and confusion matrix between classes for both linear and RBF SVMs.
- Also show some images where your method correctly or incorrectly predicted the class. Explain why you think this happened.
- Suggest some ways you could improve on this image classification idea, e.g. how could the representation, images, model be improved?

e. Mini Contest (extra credit)

Try implementing any of the other models or image representations described in class (or that you can find online). Enter your classification system into a class mini-contest. The best and/or most interesting models will win a prize and receive extra credit.