Assignment 4

Comp 560 Artificial Intelligence

Due: Nov 24, 2015, 11:59pm

In this assignment, you will implement image classification using SVMs.

You are free to use any high-level programming language you are comfortable with *as long as there is support for it in the libSVM package*. You may work in groups of up to 3 people.

Image Classification

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

We will be using the libSVM package which has support for multiple languages. 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: http://www.tamaraberg.com/teaching/Fall_15/HW/HW4/images.tar.gz
To open this file under mac or linux, use the command: "tar zxvf images.tar".

The data contains 4 image classes: hobo bags, clutch bags, flat shoes, and pumps as indicated by the file names.

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. (8 points)

You should implement and evaluate two different image representations:

1) A simple "tiny image" representation. To form your representation for these RGB images, your code should first resize each image to icon size 32x32x3 and then concatenate the image pixels into one long vector (length 3072).
2) A color histogram representation. To form your representation, your code should generate an 8x8x8 color histogram by binning the r,g,b channels for each image into 8 bins. If there are any non-color images in the data your code should handle these by duplicating the single image channel 3 times.

b. Training. (10 points)

You should train SVMs to recognize images from each of the given classes independently. To do this, train 4 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 "clutch" SVM, positive examples should come from the "clutch" training set and negative examples should be sampled 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 on each of the feature types computed in part a. 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 to form training and tuning sets. 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 tuning 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 selected parameter values.

Plot performance on the tuning set for each class and feature type for the different parameter settings you tried.

c. Testing. (3 points)

Evaluate model performance using images in the test set for each of the feature types you implemented. 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 create a confusion matrix M (where M_{ij} is the number of images of class i that were classified as class j).

d. Write-up. (4 points)

Your write-up should include:

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

2) Descriptions of:

- Performance for each feature type on the evaluation set and how this was used to select good parameter values for both linear and RBF SVMs.
- Classification accuracies for each feature type achieved by your models on the test set for each category, overall performance across classes, and confusion matrix between classes for both linear and RBF SVMs.
- Show 10 example images where your method correctly or incorrectly predicted the class. Explain why you think this happened for each case.
- Describe at least 3 ways you could improve on this image classification idea, e.g. how could the representation, images, model, or data 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.

Submission Instructions

HW should be uploaded to the classroom.cs.unc.edu server (one submission per group uploaded into the directory of one of the group members). Email Ric (poirson@cs.unc.edu) with the location of your HW submission, your group member names, and your PDF write-up. All submitted code must run on the classroom server for full credit. **Note**, if you edit any portion of your HW after the submission date that time-stamp will become your submission time and count into your late days.

Submissions should include:

1. A report in PDF format called HW4_lastname1_lastname2_lastname3.pdf with the last names of all group members. The report should describe your implemented solution and fully answer the questions for every part of the assignment. Your description should highlight the most "interesting" aspects of your solution, i.e., any non-obvious implementation choices and parameter settings, and what you have found to be especially important for getting good performance. Feel free to include pseudo-code or figures if they are needed to clarify your approach. Your report should be self-contained and it should (ideally) make it possible for us to understand your solution without having to run your source code.

All group reports need to include a brief statement of individual contribution, i.e., which group member(s) was/were responsible for which parts of the solution and submitted material.

2. Your source code and a README file indicating how to run your code. Code should be well commented and it should be easy to see the correspondence between what's in the code and what's in the report.

Late policy: Assignments are due at 11:59pm on the assigned date. Students have 5 free late days to use over the course of the semester (e.g. an assignment submitted at 2am after the deadline will count as 1 day late). After free late days are used, assignments will be accepted up to 1 week late at a penalty of 10% per day.