CSE 455/555 Spring 2010 Homework 3: Discriminants, SVM, and Nonparametric Density Estimation

Jason J. Corso
Computer Science and Engineering
SUNY at Buffalo
jcorso@buffalo.edu
Date Assigned 4 March 2011
Date Due 28 March 2011
Homework must be submitted in class. No late work will be accepted.

Remember, you are permitted to discuss this assignment with other students in the class (and not in the class), but you must write up your own work from scratch.
*I am sure the answers to some or all of these questions can be found on the internet. Copying from **any** another source is indeed cheating.*
This class has a zero tolerance policy toward cheaters and cheating. Don't do it.

This homework contains both written and computer questions. You must turn in the written questions (which may be parts of the computer questions) in class and you must submit the computer code via the CSE submit script. For the computer parts, on this homework in particular, it is highly recommended that you use Matlab (available on all University machines) or Python/NumPy/SciPy. However, you are free to choose your poison (C/C++ or Java). If you do so, I recommend you acquaint yourself with the CLAPACK (C Linear Algebra Package) or JAMA (Java Numerics `http://math.nist.gov/javanumerics/jama/doc`*).*

Problem 1: Kernel Density Estimation (20%)

This is problem 2 from chapter 4 DHS.

Consider a normal $p(x) \sim N(\mu, \sigma^2)$ and Parzen window function $\varphi(x) \sim N(0,1)$. Show that the Parzen window estimate

$$p_n(x) = \frac{1}{nh_n} \sum_{i=1}^{n} \varphi\left(\frac{x - x_i}{h_n}\right) \tag{1}$$

has the following properties:

1. $\overline{p}_n(x) \sim N(\mu, \sigma^2 + h_n^2)$
2. $\text{var}[p_n(x)] \simeq \frac{1}{2nh_n\sqrt{\pi}} p(x)$
3. $p(x) - \overline{p}_n(x) \simeq \frac{1}{2}\left(\frac{h_n}{\sigma}\right)^2 \left[1 - \left(\frac{x-\mu}{\sigma}\right)^2\right] p(x)$

for small h_n. (Note that if $h_n = h_1/\sqrt{n}$, this result implies that the error due to bias goes to zero as $1/n$, whereas the standard deviation of the noise only goes to zero as $\sqrt[4]{n}$.

Problem 2 and 3 Support Information

Download the files from the website:
`http://www.cse.buffalo.edu/~jcorso/t/CSE555/files/homework3-files.tar.gz`. 555hw3/problem2 contains just a set of a few images on which to proceed. 555hw3/problem3 contains a skeleton SVM implementation that you will need to complete.

Problem 2: Figure-Ground Segmentation (40%)

1. Implement the kernel-density estimation-based method for dynamically learning the pixel-color distribution for the foreground and background. The method is described in the Zhao and Davis paper we discussed in class (`http://www.cse.buffalo.edu/~jcorso/t/555pdf/ZhDaICPR2004.pdf`). Specific comments on the implementation:
   (a) Be sure to implement weighted kernel-density estimation (based on the current probability of foreground and probability of background).
   (b) Use the same exact color-based feature space that they do.
   (c) You do not need to implement the method of normalized KL-divergence for selecting the kernel scale for initialization. Just set it to some reasonable value manually.

(d) You do not need to use the method based on sample variance to set the kernel bandwidth. Just set it to something reasonable (0.1).

(e) You have the choice of either implementing a sampling-based version as Zhao and Davis have done (i.e., take 6% of the pixels each round), or you can simply process all of the pixels.

(f) Rather than implementing the Gaussian kernel as they have, use the Epanechnikov kernel:

$$K(u) = \frac{3}{4}(1 - u^2)\delta(|u| \leq 1) \tag{2}$$

(g) Have your system iterate for a fixed number of iterations (say 25).

2. Set the bandwidth to 0.1 and apply your implementation to the `flower.ppm` file and to the `butterfly.ppm` file. Output the initial probability maps (P_{fg} and P_{bg}), the initial classification, and the final maps and classification.

3. Repeat the previous step but vary the bandwidth from among (0.05, 0.1, 0.15, and 0.2). Explain the results.

4. Now, do a Google™ Image Search for *your favorite wild animal* (e.g., tiger, leopard, hawk) and take the first photographic result. Execute your automatic figure-ground segmenter on this image. Show the resulting separation and explain it. (Note, you probably want to make the image some reasonable size like 320x240 before processing it.)

Problem 3: Support-Vector Machine (40%)

You must use Matlab for this problem.

In this problem, you will explore a straightforward implementation of a support vector machine. I have provided a skeleton set of files in the zip 555h3/problem3. We have used some of these already in the class, namely the make_new_dataset.m and plot_dataset.m, but there are additional ones that contain SVM code. You will need to complete them, per the questions below.

Getting started. You can load a provided dataset easily:

```
D = read_datafile('test1.dat'); plot_dataset(D);
```

The following questions ask you to complete the SVM code by filling in the misses parts.

1. Complete line 40 in classify_grid.m, which applies a learned SVM classifier. You need this to test a classifier.

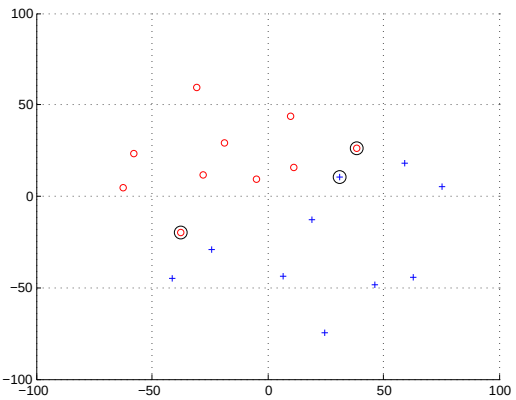
2. Complete line 48 in trainsvm.m.

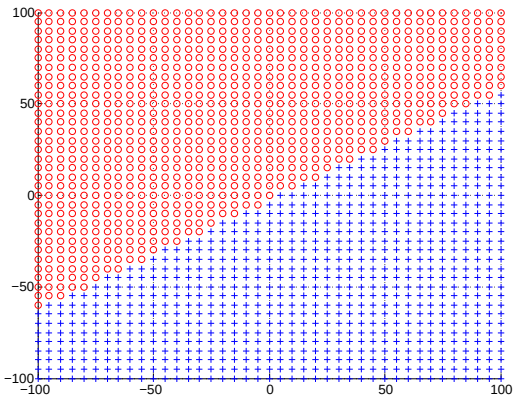
3. Complete line 78 in trainsvm.m. Use quadprog solver in matlab to actually solve for the SVM.

4. Load datafile test1.dat and run

   ```
   [SV,alpha,b] = trainsvm(D,inf,@kernel_linear,[]);
   ```

   You should ultimately see figures like the ones below popping up. The function returns `SV, alpha, b`, explain these parameters.





5. Explain what the circled data points are.

6. Explain the inf in the second parameter of trainsvm.

7. Write a new kernel function that implements a quadratic kernel. Load in nonlin1.dat and run the SVM. Plot the output and explain it.

8. Load noise1.dat. Run the code with the second parameter set to inf, 10, 1, 0.1 and plot the results. What's changing and why? What role does the second parameter play?

How and what to submit for the computer question(s): explains, figures and tables should be part of the "written" assignment due in class. The code (whether it is Matlab, Python, C or Java) should be tar'd and submitted via the CSE submit script:

1. Login to a department student Unix machine, hadar, metallica, nickelback, pollux, styx, timberlake (the code MUST work on the CSE machines).

2. Use "tar -cvf homework2.tar list-of-files-or-directories".

3. Then, type "submit_cse555 homework2.tar" if you are in 555 or "submit_cse455 homework2.tar" if you are in 455.

Include a README file to explain what is there, but only submit the code. If it is C/C++ code, you need to include a makefile that will compile on the CSE machines (it is thus encouraged to link to any libraries available there rather than submitting them). If it is Java code, you need to include an ant file that will compile on the CSE machines. In all cases, you need to include a script named "go" that will (once compiled) do everything requested in the assignment (in the case of C or Java, it is sufficient to generate image files of the plots and save them to reasonably named files). The script can be bash/tcsh for C/Java and should be a simple Matlab/Python script for Matlab/Python.