Problem Set 6
Due on Friday, November 21, 2014 at 11:55 pm

How to Submit

Create one zip file (.zip) of your code and submit it via `ilearn.ucr.edu`. This zip file should have a single script named `ps6.m` that runs all of the examples. It should also contain the code necessary to run these samples. Finally, it should contain a single pdf file (ps6.pdf) that has your plots.

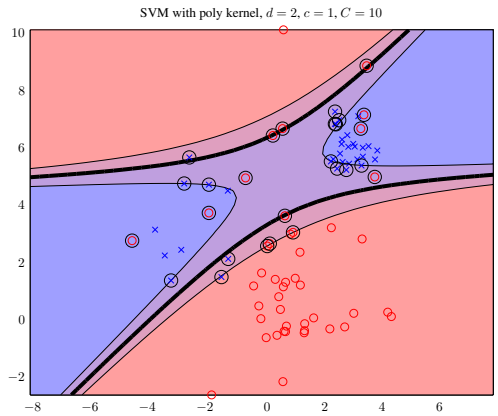
Do not supply any directories in your zip file. Each file should (in comments) list

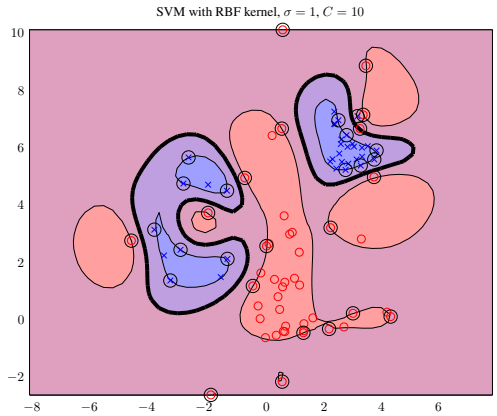
- Your name
- Your UCR student ID number
- The date
- The course (CS 229)
- The assignment number (PS 6)

Support Vector Machines part a.
Use the `solvesvm` function supplied to train a support vector machine. Plot the decision surface (as we have been doing for a few assignments on the same `class2d.ascii` data) for a polynomial kernel with $d = 2$ and $c = 1$ and the SVM penalty term $C = 10$.

Now make the same plot for a radial basis function kernel with $\sigma = 1$ and $C = 10$.

To check, my plots look like the following.





Cross Validation part b.
Selecting C is difficult to do by hand. Implement a procedure to select C by cross-validation.

In particular, use n fold cross validation k times. That is, k different times, randomly permute the data. Then for that permutation, run cross validation with n folds for every C value you are considering.

The score of a particular C is the average number of points gotten wrong across all n folds and all k permutations. Select the best C value.

Then, you can train an SVM on all of the data using this selected C.

For this data, $n = 10$ and $k = 5$ works reasonably. It is still a little noisy, but not too bad. Select C to be from the list (in matlab notation) `10.^(-4:0.5:2)`.

For each of the following kernels, plot both the cross-validation curve (the score as a function of C) on a semi-log scale, and the decision surface (as above):

- polynomial kernel with $d = 1$ and $c = 1$
- polynomial kernel with $d = 2$ and $c = 1$
- polynomial kernel with $d = 3$ and $c = 1$
- Gaussian kernel with $\sigma = 0.5$
- Gaussian kernel with $\sigma = 1$
- Gaussian kernel with $\sigma = 5$

There is no real way to avoid loops in the cross validation. However, your testing code should not need any loops. On my machine, it takes about 10 to 15 seconds to run one cross-validation experiment. Of course, make things easy on yourself. Write a function that does cross validation. Write a function that plots a SVM surface. Write a function that puts these together. Then, just call it six times with different parameters.

The plot for the second case above should look like the following