Machine Learning 10-701
http://alex.smola.org/teaching/cmu2013-10-701/ due on , 2013
Carnegie Mellon University

Homework 3

---

1 Choosing an SVM kernel [Leila; 25 pts]

In this problem, you will learn to use LIBSVM for classification and regression. Your goal will be to select the right kernel to build a model for each of the provided datasets. Although this question requires implementation, you will not be required to submit your code on Autolab; instead you should submit just the plots and explanations requested below to the coursework server.

The handout with the data for this question is at http://alex.smola.org/teaching/cmu2013-10-701x/assignments/assignment_3_handout.zip.

1.1 LIBSVM with Gaussian Kernels

1. Install LIBSVM from http://www.csie.ntu.edu.tw/~cjlin/libsvm/. You many use any language or IDE which has an interface to LIBSVM.
   If you want to use MATLAB you can either compile the mex files yourself and then add the files "svmtrain.mex..." and "svmpredict.mex..." to your working directory or if you find it problematic you can download the binary files for Mac 32-bit, Mac 64-bit, Linux 32-bit, Linux 64-bit and Windows 64-bit. To see the help type "svmtrain" or "svmpredict" in the command window (MATLAB has its own implementation of svmtrain, so typing "help svmtrain" will give you the wrong function.)
2. Using the data in `artificial.csv`, train a Gaussian (radial basis) Kernel for each of the following values of the parameter γ
   $$\gamma \in \{1, 10, 100, 1000, 10000\}$$
   The γ parameter of the Gaussian kernel in LIBSVM dictates how wide or narrow the kernel is. For instance $\gamma = 0.1$ represents a very smooth kernel, while $\gamma = 10000$ is a narrow one. A narrow kernel can better separate the training points, but it is more prone to overfitting.For each of the models obtained
   - Plot the decision boundary on top of a scatterplot of the data. You can create your own function (for which you might use MATLAB's `contour` function), or modify the function `svmtoy` if you like: http://home.caltech.edu/~htlin/program/libsvm/doc/svmtoy.m. Please make the boundary lines easy to see. The model structure returned by svmtrain also contains the support vectors, you should label the support vectors differently from the other points.
     See figures at end
   - State—based on the plot—whether the model overfits, underfits, or performs well.
     Kernels with $\gamma = 1$ and $\gamma = 10$ underfit and the kernel with $\gamma = 10000$ overfits. The other kernel ($\gamma = 100$ and $\gamma = 10000$) seem to behave appropriately.
3. Perform 10-fold cross validationto pick the appropriate γ for this dataset. Show how the testing and training errors averaged across folds change with γ. What's the value of γ you would choose and what are its corresponding test/training errors?

	1	10	100	1000	10000
train error	31.67	29.91	7.41	1.62	0.00
test error	31.75	30.15	9.60	6.30	10.40

The best value is $\gamma = 1000$ because it minimizes the test error (6.3%), it doesn't minimize the train error (1.62%). $\gamma = 10000$ clearly leads to overfitting because the train error is 0 and the test error is higher than when $\gamma = 1000$.

4. Let's assume you were submitting a paper on the usefulness of the Gaussian Kernel. Is it ok to report the test error you obtained after 10-fold cross validation as the test error of your model? Why or why not?
   No. The test errors we report here do not correspond to errors when the model is generalized to new data. The full dataset was chosen to optimize γ.

For 1.12 use *all* the data to train the model. For 1.13, split the data into 10 folds, and use 9 of the folds for training and the remaining fold for testing each time. Keep the data in the same order it appears in the file: i.e., the first 10% is the first fold, the next 10% is the next fold, etc. Do NOT randomly shuffle the data.

1.2 Unbiased Model Evaluation

Compare the following two ways of model evaluation on `dataset2.csv`. Use the same kernel described in 1.1, but try values

$$\gamma \in \{1, 5, 10, 50, 100, 500, 1000, 5000, 10000\}.$$

- (1-stage CV) Split your data into 2 sets. Do 10-fold CV separately on each set and average the two cross-validation test results. Report the best kernel and the test error obtained by averaging over the two sets.

	1	5	10	50	100	500	1000	5000	10000
set 1	35.90	31.00	24.90	17.30	14.00	9.20	9.90	14.60	22.70
set 2	36.70	32.70	26.20	19.70	14.80	9.70	9.20	18.50	27.60
average	36.30	31.85	25.55	18.50	14.40	9.45	9.55	16.55	25.15

- (2-stage CV) Split your data into 2 sets. Do 10-fold CV on Set_1 and pick the best kernel K_1. Compute the test error of K_1 on Set_2, thus obtaining the value E_1. Do 10-fold CV on Set_2; pick the best kernel K_2. Evaluate K_2 on Set_1, obtaining the test error E_2. Report $E = \frac{E_1 + E_2}{2}$.

Using 1–fold CV on Set1, we pick $\gamma = 500$, rerun the SVM on Set1 and get a classification error $E1 = 9.9\%$. We choose $\gamma = 1000$ with CV on Set2 and get a classification error $E2 = 8.8\%$. $E = \frac{E_1 + E_2}{2} = 9.35\%$.

Compare the two values. Which one do you think is a more accurate estimator of how your model would behave on hold-out data?

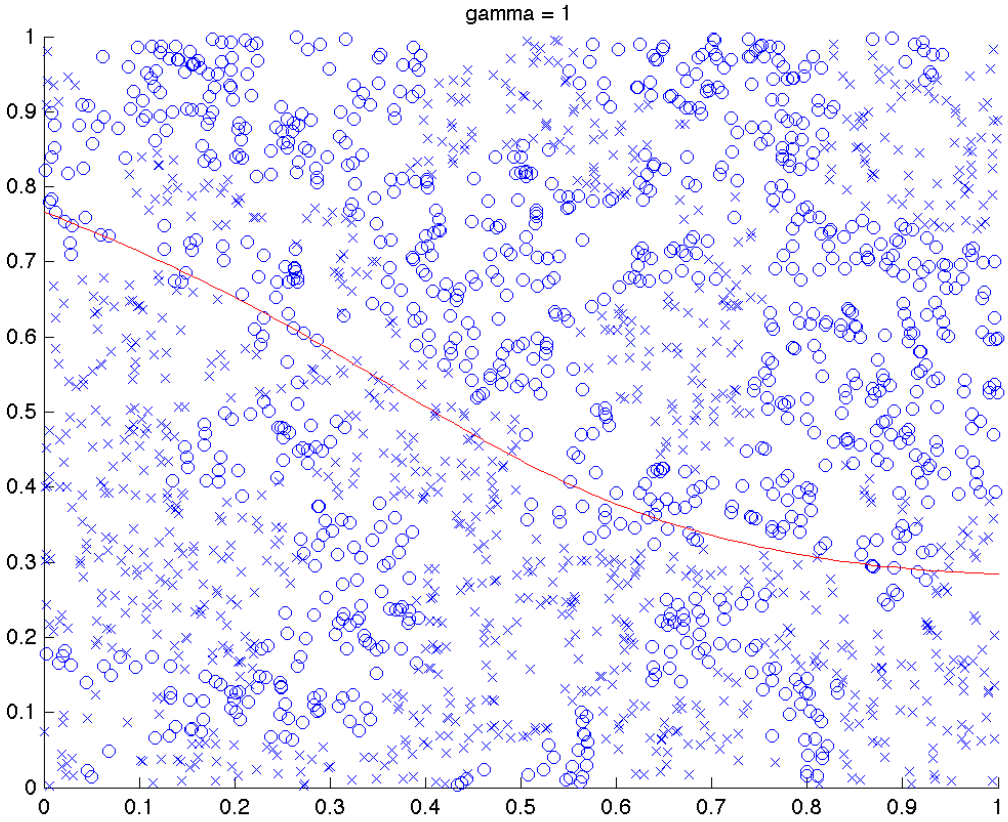
Do K_1 and K_2 match in the 2-stage CV?

Suppose that K_1 and K_2 don't match in the 2-stage CV. What conclusion can you draw? In this case, how would you report the results about error rate and kernel selection in a paper?

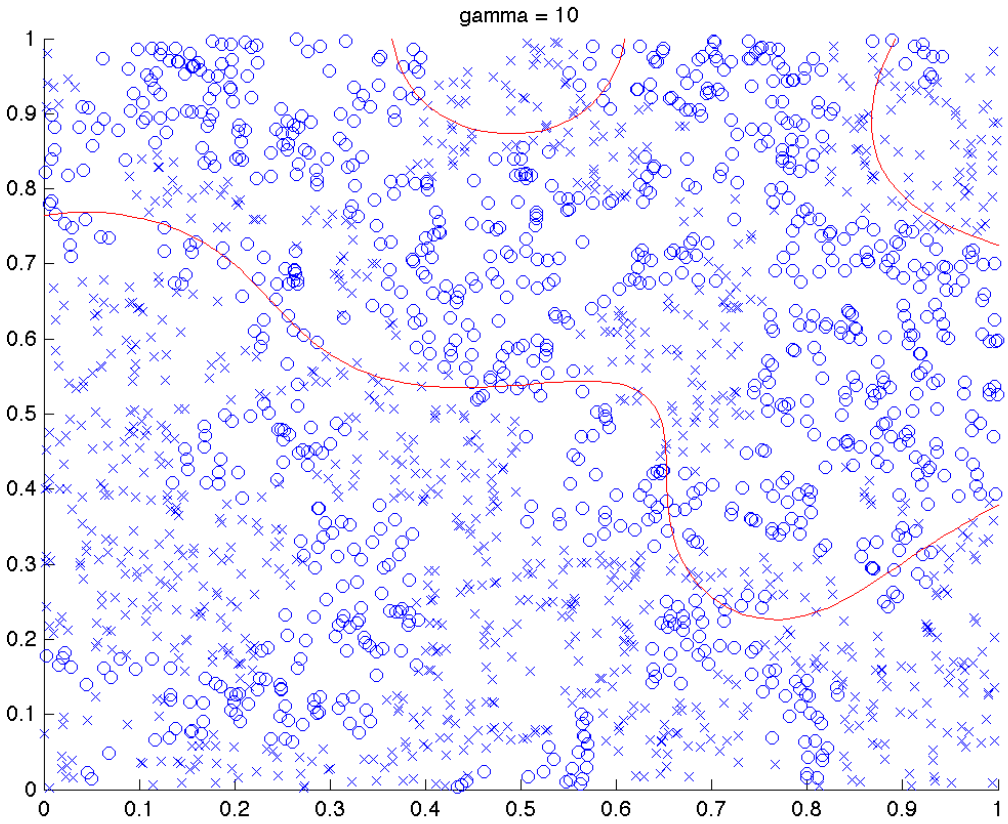
The 2-stage CV yields a more accurate estimate of the model's true error. It captures the sensitivity of your choice of parameters to the variability of the training data. In the 2-stage CV approach the key element is that the final test is done with the final parameter that is chosen on a separate dataset. The proportion of sets 1 and 2 can be varied. For example you could do a 2-stage cross validation by having two nested 10-fold CV: using 90% of the data, do 10-fold CV, pick a final parameter and get the test error on the held out 10%. Then repeat for the 10 folds.

$K1$ and $K2$ did not match. The mean error rate E from 2 stage CV should be reported in the paper as well as the fact that the chosen kernel is not always the same.

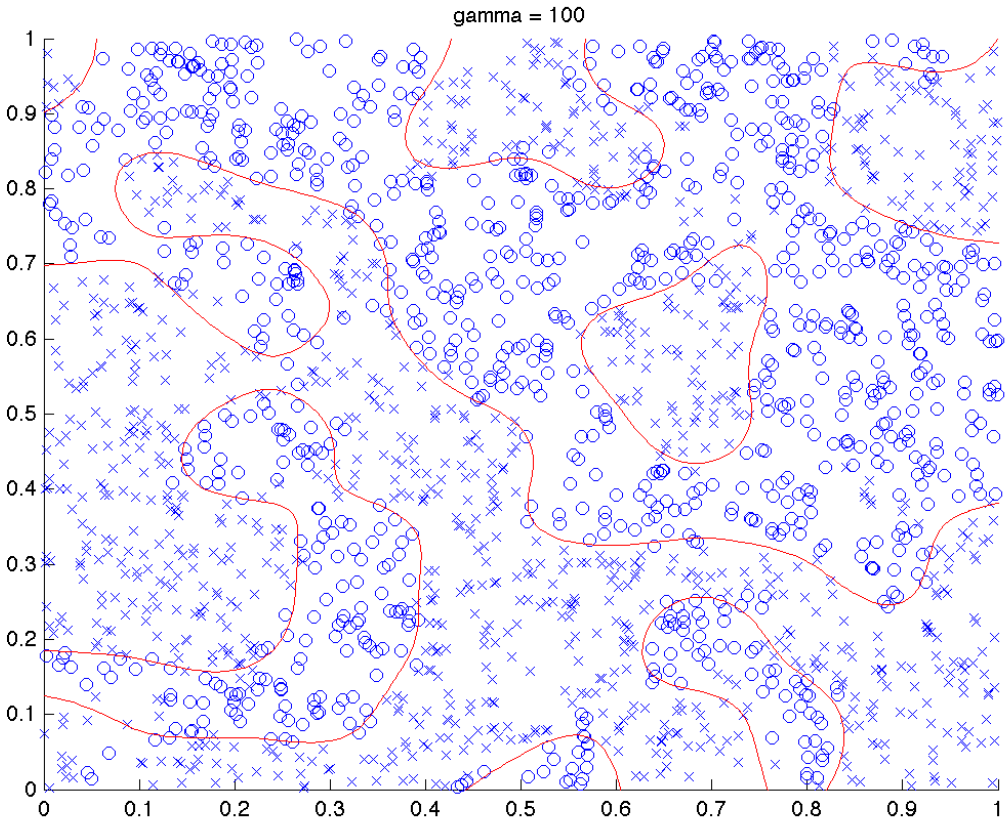
Homework 3



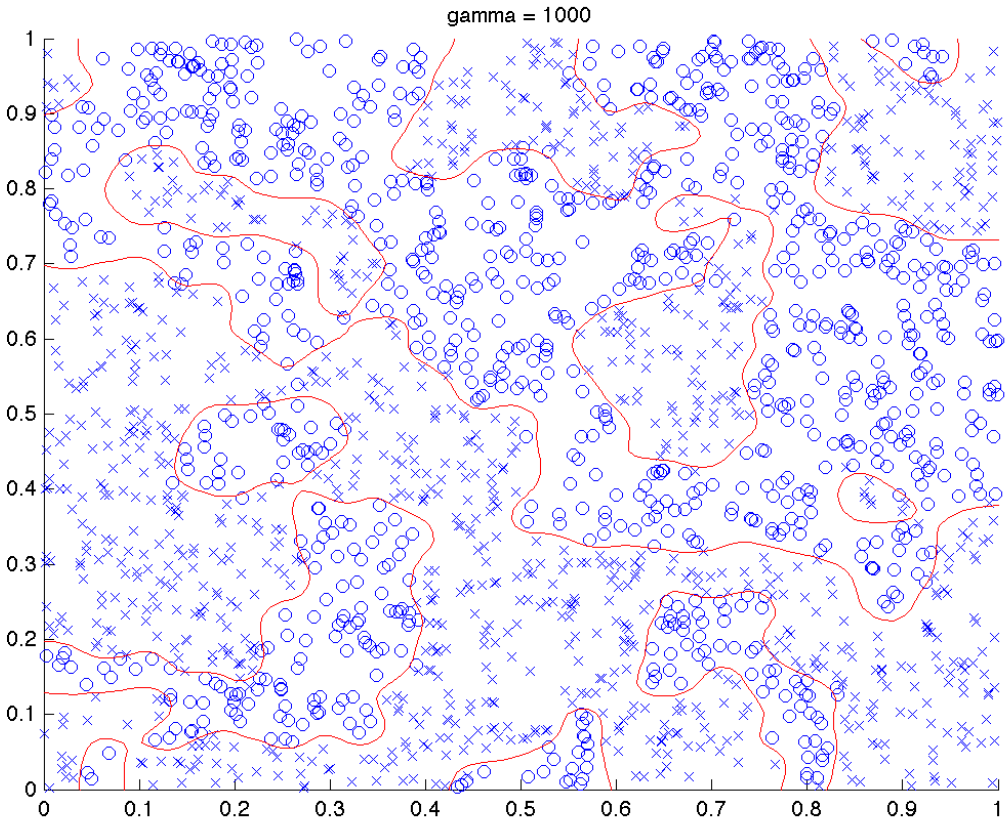
Homework 3



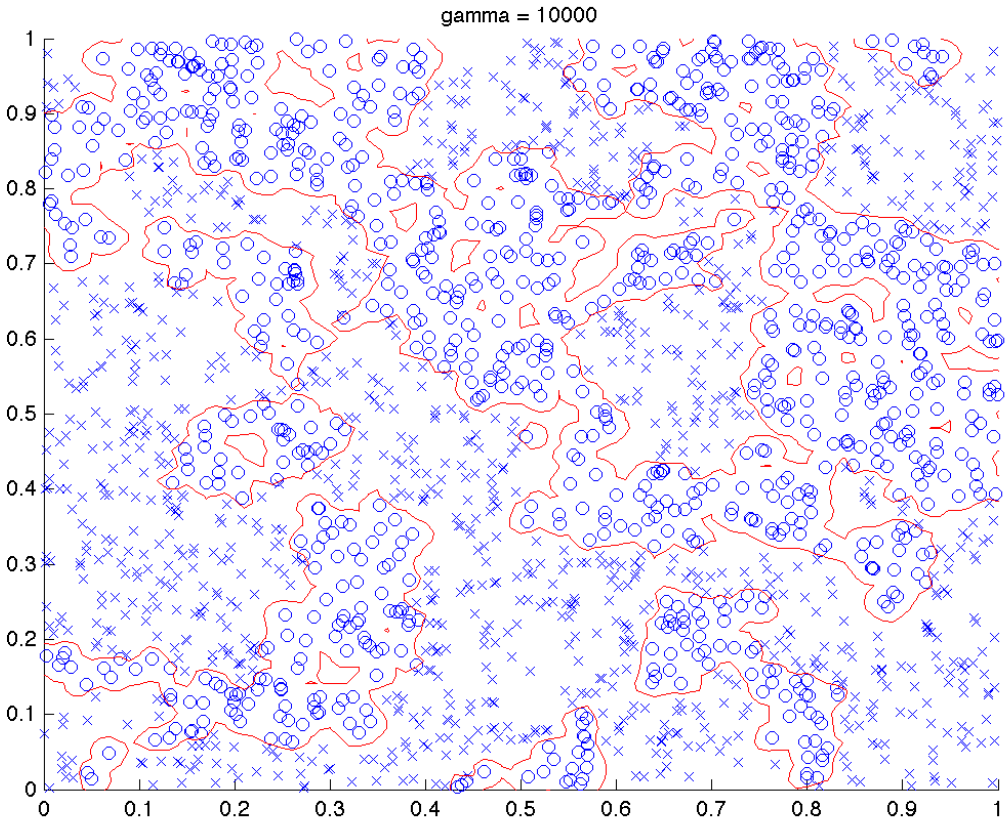
Homework 3



Homework 3



Homework 3


gamma = 10000
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1