

CS 410/510
Machine Learning
Winter, 2006

Homework 6:
Computational Learning Theory and Support Vector Machines

Due Thursday, March 2.

Part A: Written Problems

1. Recall that a bound on the sample complexity for finite hypothesis spaces is given by

$$m \geq \frac{1}{\epsilon} (\ln |H| + \ln \frac{1}{\delta}).$$

Reiterate the derivation for this formula that was given in class.

2. (a) Let X be the set of all 4-bit strings (i.e., strings that consist of 1s and 0s). Let $C = H$ be the set of “schemas” over 4-bit strings, where a schema consists of either the null set \emptyset or four-symbol strings consisting of symbols 1, 0, and *, where * matches either 1 or 0. For example, $h = 1***$ returns “true” for any 4-bit string whose leftmost bit is a 1, and returns “false” for all other 4-bit strings. Likewise, $h = ****$ is the hypothesis that returns “true” for all 4-bit strings. $h = \emptyset$ returns false for all strings.

Use the inequality given in problem 1 above to give a lower bound on the number of training examples needed for a consistent learner using H to, with probability greater than or equal to 0.95, learn a target schema with error less than or equal to 0.01. (Show your work.)

- (b) Is X shattered by H ? Prove your answer.

- (c) What is the VC dimension of H ? Prove your answer.

3. Let instance space $X = \mathfrak{R}$. Consider the (infinitely large) set of target functions $\{t_a : a \in \mathfrak{R}\}$, such that $t_a(x) = 1$ if $x > a$; otherwise $t_a(x) = -1$. That is, each possible target function represents a “threshold” a and returns 1 if instance x is greater than the threshold; -1 otherwise.

Let hypothesis space $H = \{b \in \mathfrak{R}\}$. For a given target function t_a , hypothesis h_b guesses that $b = a$.

What is $VC(H)$? Prove your answer.

Part B: Experimenting with a Support Vector Machine

In this part of the assignment you will experiment with using a support vector machine on the UCI spam classification data set.

The support vector machine system you will use is called SVM-Light. It was written by Thorston Joachims. You can download the code from <http://svmlight.joachims.org/> This web site also gives instructions on how to compile and run the code.

Here is what you need to do for this assignment:

1. Download the SVM_light code from the URL given above, and read the instructions on how to run it. You can simply use all the default parameters, which means you don't need to specify any options (except for the experiments below on varying the kernel type.)
2. Download the training ("spam.svm.train") and test ("spam.svm.test") files from the class web page. These are again examples from the UCI spam data set, but now in the format required by SVM_light (note that in each example, zero-valued features are skipped). The training file now has 960 instances and the test file has 3681 instances, each with about 39.5% spam and 59.5% non-spam. (I made the training file short here because SVM_light is fairly slow to train on large data sets. On my Linux machine, it took about 5 minutes or so to train SVM_light on this training data.)
3. Run `svm_learn` on the training data (`spam.data`). Then run `svm_classify` on the test data. Report the following (you can get all this information from the output of the SVM code and from the "model" file it creates):
 - The classification accuracy (i.e., fraction of correct classifications) on the training data.
 - The classification accuracy (i.e., fraction of correct classifications) on the test data.
 - The precision and the recall on the test data.
 - The number of support vectors that were used in the model created by the SVM.
 - The estimated VC-Dimension of the model (or an upper bound on it).

Repeat a few times, using different sizes of training sets (you can choose the sizes—if you make the training set larger, simply move some examples from the test set to the training set).

4. Now go back to the original UCI spam data that you used for the assignments on decision trees and on naive Bayes classifiers. (This is still available on the class web site.) Using the original test data (960 instances) as the new training data and the original training data as the new test data, run your C4.5 code and your naive Bayes classifier code on this data, and compare the classification accuracy of these on the (new) test set with your support vector

machine results from step 3. (You only need to make this comparison for the training set of size 960.)

5. In step 3 you used the default *linear* kernel. Repeat step 3 with the original-size training and test data using two other kernels (specified with the `-t` option): *polynomial* (1) and *radial basis function* (2). Report the same results for each of these that you reported in step 3.

What to hand in

Hand in your neatly written or formatted answers for Part A. For Part B, summarize your results and answer these questions (for Part B, in a computer-formatted—not handwritten) write-up:

- How does the SVM's generalization performance compare with that of naive Bayes and decision trees?
- How does the size of the training set affect the SVM results?
- How do the SVM results vary with number of support vectors in the model?
- Take a look at the support vectors in one of the models that was created by the SVM. Each support vector is one training example. How do you interpret the fact that so many support vectors had to be used in this problem?
- Which kernel gives you the best generalization performance?
- Do you see evidence of the structural risk minimization principle? That is, to minimize generalization error, you must minimize error on the training set at the same time as minimizing VC dimension. Discuss any tradeoffs that you see between these two values.
- Do you see any evidence of over-fitting? Explain.
- Define precision and recall. Why do you think the precision is often (perhaps always) larger than the recall in the results of SVMs on this problem?
- Why is the classification accuracy on the test data sometimes higher than on the training data?