10701/15781 Machine Learning, Spring 2007: Homework 3

Due: Monday, March 5, beginning of the class (no late days)

Instructions

There are three questions on this assignment. Refer to the webpage for policies regarding collaboration, due dates, and extensions. No late days will be allowed for this homework, so that we can post the solutions before the midterm exam.

1 [25 points] Error bound for 1-nearest neighbor classifier (Brian)

A nice result by Cover and Hart (1967) shows that, as the amount of training data approaches infinity, the error rate of the 1-nearest neighbor classifier is at most twice the Bayes-optimal error rate. In this problem, you will prove this result for the case of binary classification with real-valued inputs.

Let x_1, x_2, ... be the training examples and y_i be the corresponding binary class labels, $y_i \in \{0, 1\}$. You can think of x as points in some fixed d-dimensional Euclidean space.

Let $p_y(x) = p(X = x|Y = y)$ be the true conditional probability distribution for points in class y. We assume continuous and non-zero conditional probabilities: $0 < p_y(x) < 1$ for all x and y. Let also $\theta = p(Y = 1)$ be the probability that a random training example is in class 1. Again, assume $0 < \theta < 1$.

1. Given these expressions calculate the true probability $q(x) = p(Y = 1|X = x)$ that a data point x belongs to class 1. Express $q(x)$ in terms of $p_0(x)$, $p_1(x)$, and θ.
2. A Bayes-optimal classifier is a classifier that always assigns a data point x the most probable class $\arg\max_y P(Y = y|X = x)$. This means Bayes-optimal classifier is maximizing the probability of correct classification. Given some test data point x, what is the probability that example x will be misclassified using the Bayes-optimal classifier, in terms of $q(x)$?
3. Now the 1-nearest neighbor classifier assigns a test data point x the label of the closest training point x'. Given some test data point x and its nearest neighbor x', what is the expected error of the 1-nearest neighbor classifier, i.e., the probability that x will be misclassified, in terms of $q(x)$ and $q(x')$?
4. In the asymptotic case, the number of training examples of each class goes to infinity, and the training data fills the space in a dense fashion. Then the nearest neighbor x' of x has $q(x')$ converging to $q(x)$. By performing this substitution in the previous expression, give the asymptotic error for the 1-nearest neighbor classifier at point x, in terms of $q(x)$.
5. Show that the asymptotic error obtained in part 4 is less than twice the Bayes-optimal error obtained in part 2.
6. Why doesn't this asymptotic error bound hold in the non-asymptotic case, where the number of training examples is finite? What happens then?

2 [25 points] LOOCV (Jon and Purna)

2.1 Nearest Neighbor Classification: Two Classes

2.1.1 Mean Squared Error

Suppose you have 100 datapoints $\{(x^{(k)}, y^{(k)})\}_{k=1}^{100}$. Your dataset has one input and one output. The kth datapoint is generated as follows:

$$\begin{aligned} x^{(k)} &= k/100 \\ y^{(k)} &\sim \mathit{Bernoulli}(p) \end{aligned}$$

Note that all of the $y^{(k)}$'s are just noise, drawn independently of all other $y^{(k)}$'s. You will consider two learning algorithms:

- **Algorithm NN:** 1-nearest neighbor (with ties broken arbitrarily).
- **Algorithm Zero:** Always predict zero

Hint: Recall that the mean of $Bernoulli(p)$ is p and the variance is $p(1-p)$.

1. What is the expected Mean Squared Training Error for **Algorithm NN**?
2. What is the expected Mean Squared Training Error for **Algorithm Zero**?
3. What is the expected Mean Squared LOOCV error for **Algorithm NN**?
4. What is the expected Mean Squared LOOCV error for **Algorithm Zero**?

2.1.2 Some Pathological Examples

Now let's think about the pathology of LOOCV with respect to Nearest Neighbors. Let the dataset contain 100 1-dimensional non-overlapping points such that there are 50 positive examples and 50 negative examples.

1. Can you come up with an example of such a dataset for which the LOOCV accuracy with 1-nearest neighbor is 0%? Briefly explain.
2. Can you come up with an example of such a dataset for which the LOOCV accuracy with 1-nearest neighbor is 100%, but with 3-nearest neighbors is 0%? Briefly explain.

2.2 Support Vector Machines

1. (Linear Case) Consider training a linear SVM on linearly separable dataset consisting of m points. Let $|SV|$ be the number of support vectors obtained by training on the entire set. Show that the LOOCV error is bounded above by $\frac{|SV|}{m}$.

   Hint: Consider two cases - (1) removing a support vector datapoint and (2) removing a non-support vector datapoint.
2. (General Case) Now consider the same problem as above - but instead of using a linear SVM, we'll use a general (Mercer) kernel. Assuming that the data is linearly separable in the high dimensional feature space corresponding to the kernel, does the bound in part (1) still hold? Explain why or why not.

3 [50 points] k-NN, SVM, classification and cross-validation (Andy)

In this question, you will explore how cross-validation can be used to fit "magic parameters." More specifically, you'll fit the constant k in the k-Nearest Neighbor algorithm, and the slack penalty C in the case of Support Vector Machines. For all implementation questions, please electronically submit your source code to

`/afs/andrew.cmu.edu/course/10/701/Submit/your_andrew_id/HW3/`

and supply pseudo-code in your writeup where requested.

1. Download the file `hw3_matlab.zip` and unpack it. The file `cvdataset.mat` contains the Matlab variables `traindata` (training data), `trainlabels` (training labels), `testdata` (test data), `testlabels` (test labels) and `evaldata` (evaluation data, needed later).

   This is a text classification task: given a document, you need to predict its topic. So, each row corresponds to a data point (a document). Each column is a feature, a word. The value of the feature is a relative frequency of the word in a document.

   The `cosineDistance.m` implements the *cosine distance*, a distance function commonly used for text data. It takes two feature vectors, and computes a nonnegative, symmetric distance between x and y. To check your data, compute the distance between the first training example from each class.

2. Implement the k-Nearest Neighbor (kNN) algorithm in Matlab. Hand in pseudo-code. *Hint: You might want to precompute the distances between all pairs of points, to speed up the cross-validation later.*

3. Implement n-fold cross validation for kNN. Your implementation should partition the training data and labels into n parts of approximately equal size. Hand in the source code.

4. Compute the 10-fold (i.e. $n = 10$) cross-validation error for the training data, for $k = 1, 2, \dots, 100$ and plot your results. Also plot the training and test error for the same choices of k. How do you interpret these plots? Does the value of k which minimizes the cross-validation error also minimize the test set error? Why or why not?

5. Now download *libsvm* using the link from the course website and unpack it to your working directory. It has a Matlab interface which includes binaries for Windows. It can be used on OS X or Unix but has to be compiled– see the `Readme` file. Download the files `testSVM.m` (an example demonstration script), `trainSVM.m` (for training) and `classifySVM.m` (for classification), which will show you how to use *libsvm* for training and classifying using an SVM. Run `testSVM`. This should report a test error of 0.3077. In order to train an SVM with slack penalty C on training set `data` with labels `labels`, call `svmModel = trainSVM(data, labels, C)`. In order to classify examples `test`, call `testLabels = classifySVM(svmModel, test)`. Train an SVM on the training data with $C = 4$, and report the error on the test set.

6. Now implement n-fold cross-validation for SVMs. Similarly to k-NN, split your training data into n roughly equal parts. Hand in the pseudo-code.

7. Compute the 10-fold (i.e. $n = 10$) cross-validation error for the training data, for $C = 1, 2, \dots, 100$ and plot your results. Also plot the training and test error for the same choices of C. How do you interpret these plots? Does the value of C which minimizes the cross-validation error also minimize the test set error? Why or why not?

8. Design your favorite classifier: You have to use either k-NN or SVM, but you are allowed to use arbitrary values for k or for C. For k-NN, you can invent different distance functions than the one we gave you or you can try to weigh the influence of training examples by their distance from the test point. If you want, you can do arbitrary feature selection, e.g. you can ignore columns. You can also perform any *linear* transformation of the features if you want. Whatever you do, please document it, and apply your algorithm to the `evaldata` data set. Output your class labels for this evaluation set,

one label per line, in the order of the examples from the evaluation set. Submit your labels as file `evallabels_yourid.txt` where *yourid* is your Andrew ID.

Submit the actual code and the predicted labels (in file `evallabels_yourid.txt`) to

```
/afs/andrew.cmu.edu/course/10/701/Submit/your_andrew_id/HW3/
```

You might find the Matlab `save` function helpful for saving your labels.

9. (Extra credit) Your labels will participate in a competition. The top submissions will receive great honor and some extra credit.