

10701/15781 Machine Learning, Spring 2005: Homework 3

Due: Wednesday, March 1, beginning of the class.

Please refer your questions to Jure (jure@cs) or Andreas (krausea@cs).

1 [30 points] Error bound for 1-nearest neighbor classifier

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-values inputs.

Let x_1, x_2, \dots 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, given 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 [20 points] SVMs

2.1 SVMs and feature maps

You are given a data set D with data from a single feature X_1 in \mathbb{R}^1 and corresponding label $Y \in \{+, -\}$, as in Figure 1. The data set contains two positive examples, $X_1 = 0$ and $X_1 = 2$ and one negative example $X_1 = 1$.

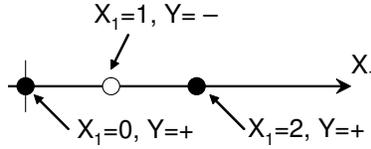


Figure 1: Dataset for SVM feature map task in Question 2.1.

1. Find a transformation (a *feature map*) of your data into \mathbb{R}^2 (i.e. a function ϕ that maps a data point X_1 from \mathbb{R}^1 to $(X'_1, X'_2) = \phi(X_1)$ in \mathbb{R}^2), such that the data becomes linearly separable.
2. Construct a maximum-margin separating hyperplane. This hyperplane will be a line in \mathbb{R}^2 , which can be parameterized by its normal equation, i.e. $w_1 X'_1 + w_2 X'_2 + c = 0$ for appropriate choices of w_1, w_2, c and all points X'_1, X'_2 on the plane. Here, $(X'_1, X'_2) = \phi(X_1)$ is the result from applying the feature map ϕ you developed in Question 2.1.1 to the original feature X_1 . Also, please explicitly compute the margin for your classification.

2.2 SVMs and the slack penalty C

You are given the data set presented in Figure 2. The slack penalty C will determine the location of the separating hyperplane. Please answer the following questions *qualitatively*, similarly to Question 1.2 in Homework 1. Give a one sentence answer/justification.

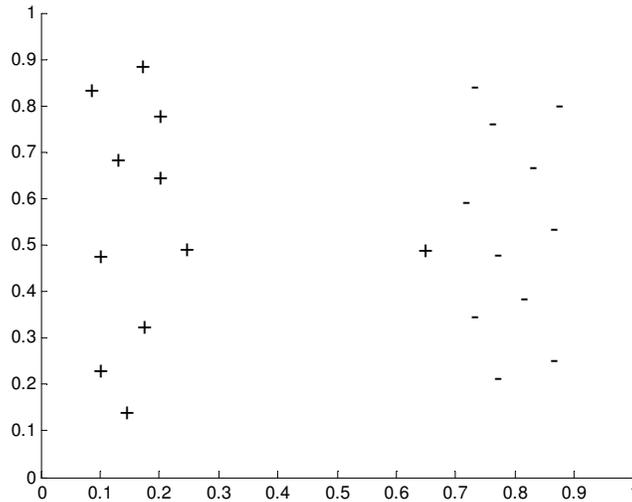


Figure 2: Dataset for SVM slack penalty selection task in Question 2.2.

1. For $C \approx 0$, indicate in Figure 2, where you would expect the decision boundary to be?
2. Where would the decision boundary be for very large values of C (i.e., $C \rightarrow \infty$)? Indicate it on Figure 2.
3. Which of the cases would you expect to work better in the classification task? Why?

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

In this question, you will explore how cross-validation can be used to fit magical 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 answers.

1. Download the file `hw3_matlab_files.zip` and unpack it. The `dataset.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 (number of times the word occurs divided by the number of words in the 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 (k NN) 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 k NN. 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 can be used in Windows only. 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/`

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