Pattern Recognition (4005-759, 20092 RIT)
Exercise 1 Solution

Instructor: Prof. Richard Zanibbi

The following exercises are to help you review for the upcoming midterm examination on Thursday of Week 5 (January 14th). If things are unclear, please bring questions to class, visit Prof. Zanibbi during his office hours, or email Prof. Zanibbi to set up a meeting at another time.

Bayes: The Formula, the Rule, and the Error

1. Define Bayesian classification in terms of the canonical model of a classifier. Is Bayes' rule optimal, and if so, under what assumptions?

   Answer: In a canonical classifier, each class ω_i has a discriminant function, and the class with the highest discriminant function output for the input vector (features) is the one selected for output by the classifier. For a Bayesian classifier, the discriminant function for each class is the *posterior* probability of the class, as defined by Bayes formula:

   $$g_i(x) = p(\omega_i|x) = \frac{P(\omega_i)p(x|\omega_i)}{p(x)}$$

   where

   $$p(x) = \sum_{j=1}^{c} P(\omega_j)p(x|\omega_j)$$

   $p(x)$ does not change for each class (is a fixed scaling factor), and so can be removed from the discriminant functions. Bayes' rule is optimal if the true prior probabilities $P(\omega_i)$ and probability density functions $p(x|\omega_i)$ are known.

2. Define Bayes error mathematically. What significant property does the Bayes error posess? There is an important relationship between Bayes' error and the error made by a 1-Nearest Neighbor (1-NN) classifier when the number of training samples n approaches infinity: what is it?

   Answer: The Bayes' error is the smallest possible error rate, and is defined by:

   $$P_e(D^*) = 1 - \sum_{i=1}^{c} \int_{R_i*} P(\omega_i)p(x|\omega_i)dx$$

   where D^* is a (true) Bayesian classifier, R_i* is the region in feature space where class ω_i has the highest posterior probability (discriminant function value), and

$P(\omega_i)p(x|\omega_i)$ is the (un-normalized) discriminant function value (see p. 32 of Kuncheva).

The Bayes error is the smallest possible error for a given feature space and set of classes. For an infinite number of training samples, the 1-NN classifier has an error that is at most twice the Bayes error (see 2.45, page 57 of Kuncheva).

Classification and Pattern Recognition: Miscellaneous

3. In class it was mentioned that one can speed up a (sequential) 1-NN classifier by organizing the training samples within a search tree. Such a search tree is organized so that some series of 'entry points' appear below the root of the tree, each of which in turn have child points that partition their associated subspace. The subspaces are defined by the voronoi cells for the set of entry points: below the root this partitions the entire feature space, the children of an entry point partition the vornoi cell of the entry point, and so on recursively down the tree.

   Provide a counter-example demonstrating that this technique will not always locate the training point closest to a query point, i.e. will not always find the nearest neighbor.

   Answer: (From Duda, Hart, Stork, p. 185) Imagine that we store a large number of prototypes that are distributed uniformly in the unit square. Imagine that we prestructure [build a tree] with this set using four entry prototypes, at (0.25, 0.25), (0.25, 0.75), (0.75, 0.25) and (0.75, 0.75) - each fully linked only to points in it corresponding quadrant. When a test pattern appears, the closest of the four entry prototypes is found, and then the search is limited to prototypes within the quadrant. In this way, 3/4 of the prototypes need never be queried.

   Note that in this method we are no longer guaranteed to find the closest prototype. For instance, suppose the test point is a boundary of the quadrants, e.g. (0.49999, 0.49999). In this case only prototypes in the first quadrant will be searched, but the closest prototype may be in another quadrant, somewhere near (0.5, 0.5). More sophisticated search trees will have each point linked to a small number of others...Nevertheless, unless we query all training prototypes, we are not guaranteed that the nearest prototype will be found.

4. Explain the relationship between *discriminant functions* and *decision boundaries*.

   Answer: For a canonical classifier, decision boundaries are defined at the positions in feature space where two or more classes have the maximum discriminant function values.

5. Explain what is meant by *generalization*, and *overtraining*.

Answer: generalization refers to how well a classifier performs on unseen data (i.e. data not in the training set). Overtraining or *overfitting* occurs when a classifier has been fit too closely to the training set, leading to poor generalization.

Classification Methods

6. Briefly explain how each of the following may be understood in terms of the canonical model of a classifier. Be specific about how discriminant functions are defined.

   (a) Quadratic classifier
   (b) k-Nearest Neighbor (k-NN)
   (c) Mutli-layer perceptron
   (d) Decision trees

   Answer: In the canonical model of a classifier, we choose the class ω_i with the maximum discriminant function value for a feature vector, $g_i(x)$.

   (a) Quadratic: discriminant functions are defined by an estimate of the posterior probability of a class given a feature vector $P(\omega_i|x)$, using gaussians with mean vector μ_i and covariance matrix Σ_i to represent the distribution of features for each class i. The discriminant function is simplified by using the log probability (a monotonic, i.e. non-decreasing transformation), and removing terms that are independent of a specific class. See Kuncheva, p. 46.
   (b) k-NN: most simply defined, the discriminant function for each class is the number of k closest neighbors to a test point. We discussed in class how this may be understood as an estimate for the posterior probabilities for each class, $P(\omega_i|x)$ (see Kuncheva, p. 57).
   (c) Multi-layer perceptron: each neuron i in the output layer of an MLP produces a discriminant value for class i. As the number of training samples approaches infinity, these discriminant values become the posterior probability for each class, again $P(\omega_i|x)$.
   (d) Decision tree: For each test input, the tree has been constructed such that the leaf node reached represents a set of training samples that occupy one region in feature space; the class returned is the one with the largest number of points at that leaf. Thus, the discriminant can be understood as being similar to that for NN *after* we have traversed the tree to the appropriate leaf (NOTE: normally in a decision tree, we do not store the samples at each node in the final tree). For example, if for a two-class problem we split the root of our tree into two leaves, splitting on $x < 5, x \geq 5$, then for test sample t with $x = 1$, the left child can be used to estimate $P(\omega_1|x < 5)$ and $P(\omega_2|x < 5)$, in both cases dividing the number of points in class i by the total number of points at the leaf.

7. Draw a 2D feature distribution for two classes, for which the training data is better fit by a nearest neighbor classifier than a linear classifier. State briefly why you chose this distribution.

> **Answer:** There are many examples; one is the banana data set used in the Kuncheva text. The two linear discriminants produce a linear decision boundary, which cannot properly represent the shapes of the class boundaries. However, assuming that we have a representative training sample, with the nearest neighbor classifier we can capture the class boundaries more closely, as it depends only on the k samples closest to a point.

8. Draw a 2D feature distribution for three classes, for which the training data is better fit by a nearest neighbor classifier than a quadratic classifier. State briefly why you chose this distribution.

> **Answer:** Again, we can do this by choosing a distribution of features that significantly violates the assumption of the probability densities for features in each class being gaussian. Roughly speaking, this and the last question try to get at where non-parametric techniques (e.g. k-NN) can be more effective in representing complex class boundaries. This comes with added expense in computation at classification time: once the gaussian parameters for LDC and QDC are determined, they are quite fast, while plain k-NN requires comparison with all stored samples (editing and/or extracting samples using Wilson, Hart, Random, Bootstrapping or other techniques can be used to reduce the sample set size).
>
> One example: consider a distribution in R^2 where the classes are close (nearly overlapping) with a crescent-shaped distribution, similar to the banana data set. The estimated likelihood of belonging to a class diminshes as we approach the tails of the banana; if the center of a banana is situated at the each endpoint (making the center of that gaussian closer than that of the true class), there will be class confusions. With sufficient samples, k-NN will represent the boundaries of the classes more closely.

9. Look at Fig 1.11(a) in the Kuncheva textbook. Sketch how the decision boundaries move if the prior probability of class 2 (ω_2) increases relative to classes 1 and 3. Explain your answer.

> **Answer:** The left decision boundary for ω_2 will shift to the left, and the right decision boundary to the right. As the prior probability of class two increases, it becomes the most probable for a larger part of the feature space, in particular near points where the two most likely classes meet.

10. Explain the relationship between a Bayesian classifier, and the linear and quadratic classifiers that we have studied in class.

Answer: The linear and quadratic classifiers assume that the probability density functions for the feature distributions in each class are normally distributed (gaussian). For the linear classifier, it is assumed that all feature distributions have the same covariance matrix; this is not assumed for the quadratic classifier. Both the LDC and QDC are simply Bayesian classifiers (using the posterior probability for the discriminant function), for which the stated assumptions about the feature distributions hold (see Kuncheva, pp. 45-48).

In practice, LDC and QDC are often used by estimating the mean vector and covariance matrix from a training set. In this case, they are not true Bayesian classifiers, but are simple to define and are still often effective in practice.

11. What is different about the types of features that may be used for decision tree classification, as opposed to the other classification techniques that we have studied?

Answer: Decision trees may be used with qualitative (nominal and ordinal sets of values such as names and ranks, respectively) as well as quantitative (in particular, real-valued) features.

Comparing Classifiers

12. In a paper that you have been reading while working on the course project, two classifiers, a neural network and a support vector machine, are run on the MNIST data set. The neural net produces a 1% error rate, while the support vector machine produces a 0.82% error rate. The authors conclude that their support vector machine is better suited to handwritten digit classification than the neural network.

Is this a valid conclusion? If so, state why, and if not, explain anything further needed to properly support the authors' claim.

Answer: This is not a valid conclusion. Even if the SVM performs better on MNIST, this does not guarantee better of performance than the neural net on handwritten digits in general. In addition, the results may be an artifact of the training set selected (sample bias), and the difference in performance may not be statistically significant; a hypothesis test comparing the two error distributions (such as McNemar test) should be made.

13. Explain how data is used to train and test classifiers in each of the following approaches. Also, identify which method is generally avoided, and why.
    - Resubstitution
    - Hold out
    - Cross-validation (including 'leave-one-out')

Answer: Resubstitution is the process of testing the classifier on the data used to train it. As a testing method it is generally avoided, as it is positively biased (in particular where a classifier has been overfit to the training set).

In the Hold out method, data is randomly split randomly into two or three sets (three if a validation set is needed); in a shuffle, the random split is produced L times, and the average error is reported. In cross-validation, the data set is split into k disjoint sets, with each set being used once as a test set, with the remaining $k-1$ sets being used for training. The results are then averaged across the k test sets. 'Leave-one-out' occurs when we treat $k = N$ (the number of samples), in which case every sample takes a turn as a single test point, with the remaining data used for training.

14. For a statistical test comparing two distributions, what is the *null hypothesis*?

Answer: that the two distributions are the same (that there is no difference between them).

Programming (MATLAB/PRTools)

To get started, I suggest that you look at the PRTools examples that have been posted, and the information on getting the MNIST data set under the 'Assignments' link on the course web page.

15. The PRTools toolbox contains a function named `rejectc`, which will create a modified version of any classifier, by 'padding' decision boundaries with a reject region grown uniformly on either side until a given threshold of the training data lies within the rejection region. Run the example shown when you type `'help rejectc'`.

Write an m-file (MATLAB program) that modifies the example to show boundaries produced when we reject up to a maximum of 5%, 20%, and 50% of the training data, with each boundary in a different color (don't forget to include the original classification boundaries).

Then produce a new data set b, again using `'gendatb'`. Compute the confusion matrix for each classifier, along with the expected classification errors for **both** the training and test sets. Briefly summarize the effect produced as the size of rejection region increases.

16. Try running 1-NN and 9-NN on the MNIST data, using the built-in `knnc` provided by PRTools (note that these will take awhile to run!). Use the training and test sets provided. Look at the confusion matrices and expected classification errors for each classifier when run on the MNIST test set.

Comment on the recognition accuracy produced by each classifier. In terms of speed, in what way would decision tree and neural network classifiers be faster than nearest neigh-

bor? In what way would they be slower?

Using any method that you like, modify the MNIST data so that the feature vectors are much smaller. Run the classifiers again, and compare the new confusion matrices and error rates produced on the test set. Try to explain briefly (in a few sentences), why you saw the result that you did.