Assignment 3: Classifiers, Clustering

10-701/15-781: Machine Learning (Fall 2004)

Out: Oct. 14th, 2004
Due: Oct. 28th 2004, Thursday, Start of class,

a *This assignment has four problems to test your understanding about classifiers and regression. Each of problems 1, 2, 3, and 5 are worth 15%, with problem 4 worth 40%.*

b *For the questions requiring programming, please use Matlab. You need to submit your code to the TAs (we'll provide instructions on how soon).*

c *For questions and clarifications, contact Max (questions 1 to 3) (maxim+@cs.cmu.edu) or Dave (questions 4 and 5) (dif+781@cmu.edu).*

d *Policy on collaboration:*

Homeworks will be done individually: each student must hand in their own answers. It is acceptable, however, for students to collaborate in figuring out answers and helping each other solve the problems. We will be assuming that, as participants in a graduate course, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.

d *Policy on late homework:*

Homework is worth full credit at the beginning of class on the due date. It is worth half credit for the next 48 hours. It is worth zero credit after that. You must turn in all of the 4 homeworks, even if for zero credit, in order to pass the course.

Question 1. Naive and Joint Bayes Classifiers (15 pts)

1.1 (5 pts) Suppose A and B are independent binary random variables, each having a 50% chance of being 0. Construct a boolean function $y = f(A, B)$ where A is not independent of B given y but for which a naive Bayes classifier will have a 0% error rate (assuming infinite training data). Prove that the classifier will have 0% error rate.

Answer: Consider the function $y = A \vee B$. Since the training data is infinite, the estimated probabilities will get estimated as follows: $P(y = 0) = \frac{1}{4}$, $P(y = 1) = \frac{3}{4}$;
$P(A = 0|y = 0) = P(B = 0|y = 0) = 1$, $P(A = 0|y = 1) = P(B = 0|y = 1) = \frac{1}{3}$;

For any inpute $A = a, B = b$, the naive classifier predicts y that maximizes $P(A = a|y)P(B = b|y)P(y)$. Hence, the classifier will predict y-values as follows:
$A = 0, B = 0$: $P(A = 0|y = 0)P(B = 0|y = 0)P(y = 0) = 1/4$, $P(A = 0|y = 1)P(B = 0|y = 1)P(y = 1) = 1/12$, prediction: $y = 0$.
$A = 0, B = 1$: $P(A = 0|y = 0)P(B = 1|y = 0)P(y = 0) = 0$, $P(A = 0|y = 1)P(B = 1|y = 1)P(y = 1) = 1/6$, $y = 1$ is predicted.
$A = 1, B = 0$: $P(A = 1|y = 0)P(B = 0|y = 0)P(y = 0) = 0$, $P(A = 1|y = 1)P(B = 0|y = 1)P(y = 1) = 1/6$, $y = 1$ is predicted.
$A = 1, B = 1$: $P(A = 1|y = 0)P(B = 1|y = 0)P(y = 0) = 0$, $P(A = 1|y = 1)P(B = 1|y = 1)P(y = 1) = 1/3$, $y = 1$ is predicted.
Thus, the classifier has zero error rate.

1.2 (10 pts) Suppose we have a function $y = (A \wedge B) \vee \neg(B \vee C)$, where A, B, C are again independent binary random variables, each having a 50% chance of being 0.
a) (3 pts) How many parameters a naive Bayes classifier needs to estimate (without counting $P(\neg x)$ as a parameter if $P(x)$ is already counted as an estimated parameter)? What will be the error rate of the naive Bayes classifier (assuming infinite training data)?

Answer: A naive classifier will need to estimate the probability of class $y = 0$: $P(y = 0)$, and the probabilities of input values within each class: $P(A = 0|y = 0)$, $P(B = 0|y = 0)$, $P(C = 0|y = 0)$, $P(A = 0|y = 1)$, $P(B = 0|y = 1)$, $P(C = 0|y = 1)$.
Altogether it is 7 parameters.
In general, for m input binary variables it would have been: $2m + 1$.

To compute error rate we can construct a Boolean table of the function and use it to estimate probabilities (since we assume infinite training data).

A	B	C	y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

The estimated parameters are:
$P(y = 0) = 1/2$;
$P(A = 0|y = 0) = 3/4$, $P(B = 0|y = 0) = 1/2$, $P(C = 0|y = 0) = 1/4$
$P(A = 0|y = 1) = 1/4$, $P(B = 0|y = 1) = 1/2$, $P(C = 0|y = 1) = 3/4$

For any input $A = a, B = b, C = c$ the classifier predicts y that maximizes $P(A = a|y)P(B = b|y)P(C = c|y)P(y)$.
The predictions of the classifier are then as follows (assuming that in case of a tie it always predicts 1):
$A = 0, B = 0, C = 0$: $y_{pred} = 1$
$A = 0, B = 0, C = 1$: $y_{pred} = 0$
$A = 0, B = 1, C = 0$: $y_{pred} = 1$
$A = 0, B = 1, C = 1$: $y_{pred} = 0$
$A = 1, B = 0, C = 0$: $y_{pred} = 1$
$A = 1, B = 0, C = 1$: $y_{pred} = 1$
$A = 1, B = 1, C = 0$: $y_{pred} = 1$
$A = 1, B = 1, C = 1$: $y_{pred} = 1$

From this, we can compute the error rate as the number of mistakes over the number of possible inputs (since each input is equally likely): error rate $= 2/8 = 0.25$.

b) (3 pts) How many parameters a joint Bayes classifier needs to estimate? What will be the error rate of the joint Bayes classifier (assuming infinite training data)?

Answer: A joint classifier will need to estimate a probability of one of the two classes, say $P(y = 0)$, and probabilities for all possible inputs within each class (minus one within each class since the sum of all input value probabilities sums to one), that is: $P(A = 0, B = 0, C = 0|y = 0), P(A = 0, B = 0, C = 0|y = 1), P(A = 0, B = 0, C = 1|y = 0), P(A = 0, B = 0, C = 1|y = 1), \ldots P(A = 1, B = 1, C = 0|y = 0), P(A = 1, B = 1, C = 0|y = 1)$.

Thus, it is $1 + 2 * (2^3 - 1) = 15$.
In general, for m input binary random variables it would have been $1 + 2 * (2^m - 1)$.

The error rate of a joint Bayes classifier is zero (assuming infinite training data) since it can model an arbitrary complex Boolean function.

c) (4 pts) Suggest a Bayes classifier that will need to estimate less number of parameters than a joint Bayes classifier but will still have a zero error rate (assuming infinite training data). Show that the classifier has this error rate.

Answer: Consider a Bayes classifier that assumes that A is independent of C when conditioned on B and on y (unlike a naive Bayes classifier that assumes that A, B, C are all independent of each other when conditioned on y).

Then $P(A, B, C|y) = P(A, C|B, y)P(B|y) = P(A|B, y)P(C|B, y)P(B|y)$.
For any input $A = a, B = b, C = c$ the classifier then predicts y so as to maximize $P(A = a|B = b, y)P(C = c|B = b, y)P(B = b|y)P(y)$.

The parameters it needs to estimate are: $P(y = 0)$, $P(B = 0|y = 0)$, $P(B = 0|y = 1)$,
$P(A = 0|B = 0, y = 0)$, $P(A = 0|B = 0, y = 1)$, $P(A = 0|B = 1, y = 0)$, $P(A = 0|B = 1, y = 1)$,
$P(C = 0|B = 0, y = 0)$, $P(C = 0|B = 0, y = 1)$, $P(C = 0|B = 1, y = 0)$, $P(C = 0|B = 1, y = 1)$
Thus, there are 11 parameters to estimate.

Using the Boolean table above, we can estimate the parameters as follows: $P(y = 0) = 1/2$
$P(B = 0|y = 0) = 1/2$, $P(B = 0|y = 1) = 1/2$,
$P(A = 0|B = 0, y = 0) = 1/2$, $P(A = 0|B = 0, y = 1) = 1/2$, $P(A = 0|B = 1, y = 0) = 1$, $P(A = 0|B = 1, y = 1) = 0$,
$P(C = 0|B = 0, y = 0) = 0$, $P(C = 0|B = 0, y = 1) = 1$, $P(C = 0|B = 1, y = 0) = 1/2$, $P(C = 0|B = 1, y = 1) = 1/2$

The predictions of the classifier are then as follows (there are no ties):
$A = 0, B = 0, C = 0$: $y_{pred} = 1$
$A = 0, B = 0, C = 1$: $y_{pred} = 0$
$A = 0, B = 1, C = 0$: $y_{pred} = 0$
$A = 0, B = 1, C = 1$: $y_{pred} = 0$
$A = 1, B = 0, C = 0$: $y_{pred} = 1$
$A = 1, B = 0, C = 1$: $y_{pred} = 0$
$A = 1, B = 1, C = 0$: $y_{pred} = 1$
$A = 1, B = 1, C = 1$: $y_{pred} = 1$

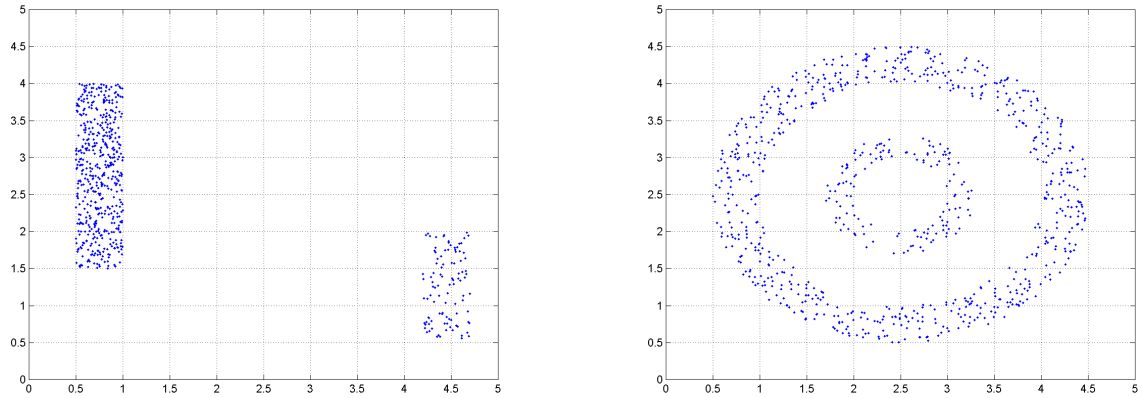
This corresponds to zero error rate.

Question 2. Spectral Clustering I (15 pts)

In this problem we will analyze the operation of one of the variants of spectral clustering methods on two datasets shown in Figure 1. For each of the datasets (unless directed otherwise) please answer the following questions.

2.1 (3 pts) The first step is to build an affinity matrix. The matrix defines the degree of similarity between points.
a) Suppose we use the L2 norm to construct the following affinity matrix (let x_i denote an ith datapoint):

$$A(i, j) = A(j, i) = \begin{cases} 1 & \text{if } |x_i - x_j|_2 < \theta \\ 0 & \text{otherwise} \end{cases} \tag{1}$$



(a) (b)

Figure 1: Plots for problem 2

What θ value would you choose and why?

b) Suppose instead we use Gaussian kernel for our affinity matrix:

$$A(i,j) = exp\left(-\frac{|x_i - x_j|_2}{2 * \sigma^2}\right) \tag{2}$$

What σ value would you choose and why?

Answer: In general, you want to choose such a parameter that the similarity between points in different clusters is 0 (or close to it), while the similarity between *neighboring* points in the same cluster is close to 1. The answers to this question are based purely on eye-balling. So, for the affinity matrix in (a) and the dataset in Figure 1(a) θ in between about 2.5 and 3 will result in an ideal case. For the dataset in Figure 1(b) it is less clear, but a value of 0.5, for example, will give us what we want.

For the Gaussian kernel, we want to set σ to obtain the same effect (even though we can not really achieve 0 similarity). So, for example, for the dataset in Figure 1(a) $\sigma = 0.5$ and for the dataset in Figure 1(a) $\sigma = 0.3$ should separate points reasonably.

2.2 (4 pts) The second step is to compute first k dominant eigenvectors of the affinity matrix, where k is the number of clusters we want to have
a) For the dataset in Figure 1(a) and the affinity matrix defined by equation 1 is there a value of θ for which you can compute analytically eigenvalues corresponding to the first two dominant eigenvectors? If not, explain why not. If yes, compute and write these eigenvalues down.

Answer: Yes, consider $\theta = 2.6$. It will result in the affinity matrix that is a block matrix: $A = \begin{pmatrix} 1_{nxn} & 0 \\ 0 & 1_{mxm} \end{pmatrix}$
where 1_{nxn} is a block of ones of size n by n, and 1_{mxm} a block of ones of size m by m. n is the number of points in left cluster, and m is the number of points in the right cluster.

Such matrix has one eigenvalue $\lambda_1 = n$ (with a corresponding eigenvector $v_1 = [1 \dots 1_n 0 \dots 0]^T$), and a second eigenvalue $\lambda_2 = m$ (with a corresponding eigenvector $v_2 = [0 \dots 0_n 1 \dots 1]^T$). These are in fact the only eigenvalues of A, since $rank(A)$ is clearly 2.

2.3 (4 pts) The third step is to cluster the rows of the matrix Y into k clusters using K-means (or a similar algorithm), where Y is constructed by placing k dominant eigenvectors into columns and re-normalizing the rows (to make each row a unit vector).
a) For the dataset in Figure 1(a) and the affinity matrix defined by equation 1 write down your best guess for the coordinates of $k = 2$ cluster centers.

Answer: Given the eigenvectors $v_1 = [1 \dots 1_n 0 \dots 0]^T$ and $v_2 = [0 \dots 0_n 1 \dots 1]^T$, we have: $Y = \begin{pmatrix} 1 & 0 \\ \vdots & \vdots \\ 1_n & 0_n \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 1 \end{pmatrix}$

If we run K-means on the rows of Y, we get two clusters with centers at $< 1, 0 >$ and $< 0, 1 >$.

2.4 (4 pts) Finally, given the clusters on matrix Y, a point x_i is declared to be in cluster j iff the ith row of Y is in cluster j.
a) What are the final clusters you would expect to obtain for each of the datasets? Provide a rough sketch of the clusters to give an idea.

Answer: This should result in almost perfect clustering (that is, will cluster each connected component together).

b) What are the clusters you would expect to obtain if using EM algorithm for Gaussian Mixture Models with 2 clusters? Also provide a rough sketch of the clusters.

Answer: The dataset in Figure 1(a) should be clustered perfectly, while the dataset in Figure 1(b) will probably be split in two clusters - half of the inner and outer circles in one cluster, and the other halves of the circles in the second cluster. Thus, the dataset in Figure 1(b) is clustered incorrectly (assuming we want each circle to form its own cluster)

Question 3. Spectral Clustering II (15 pts)

The version of spectral clustering we have studied in class made use of matrix $A = D^{-1/2} W D^{-1/2}$. W is an affinity matrix with $w_{ij} = w_{ji}$ being a non-negative distance between points x_i and x_j. D is a diagonal matrix whose i^{th} diagonal element, d_{ii}, is the sum of W's i^{th} row. In the following you will need to prove several properties about A that are important for a good understanding of spectral clustering.

For the proofs you might find useful the following property: for any symmetric matrix B with all non-negative entries if u is an eigenvector with all positive entries, then no other independent eigenvector of B has the same eigenvalue.

3.1 (3 pts) Show that a vector $v_1 = [d_{11}^{\frac{1}{2}} d_{22}^{\frac{1}{2}} \dots d_{nn}^{\frac{1}{2}}]^T$ is an eigenvector of A with an eigenvalue $\lambda_1 = 1$.

Answer: We need to show that $Av_1 = v_1$.

$$
\begin{aligned}
&D^{-1/2} W D^{-1/2} v_1 = \\
&D^{-1/2} W D^{-1/2} [d_{11}^{\frac{1}{2}} d_{22}^{\frac{1}{2}} \dots d_{nn}^{\frac{1}{2}}]^T = \\
&D^{-1/2} W D^{-1/2} [d_{11}^{\frac{1}{2}} d_{22}^{\frac{1}{2}} \dots d_{nn}^{\frac{1}{2}}]^T = \\
&D^{-1/2} W [11 \dots 1]^T = \\
&\left[\frac{\sum_j (w_{1j})}{d_{11}^{\frac{1}{2}}} \dots \frac{\sum_j (w_{nj})}{d_{nn}^{\frac{1}{2}}} \right]^T = \\
&\left[d_{11}^{1/2} \dots d_{nn}^{1/2} \right]^T = \\
&v_1
\end{aligned}
$$

3.2 (4 pts) Prove that $\lambda_1 = 1$ is the largest eigenvalue of A.

Answer: We have just shown that A has an eigenvalue of $\lambda_v = 1$ with a corresponding fully positive eigenvector v. We need to show no eigenvector can have an eigenvalue larger than 1. We prove by contradiction. Suppose not and there exists an eigenvector u, s.t. $\lambda_u > 1$.

Let us derive a scaled version of this vector, $u' = c * u$, where $c = \min_{i,s.t.u_i>0} \frac{v_i}{u_i}$ (Such c exists because v is fully positive, u is orthogonal to v since A is symmetric, and therefore in order to have $u \cdot v = 0$, u must contain both negative and positive elements). Clearly, u' is still an eigenvector of A with the same eigenvalue. We also now have the following: $\forall i, u'_i \leq v_i$ and $\exists j, u'_j = v_j$. As a result, $v - u' \geq 0$, where 0 is a column vector of zeros. Consequently, we must have $A(v - u') \geq 0$ since A is non-negative.

On the other hand, $A(v - u') = Av - Au' = \lambda_v v - \lambda_u u' = v - \lambda_u u'$. However, since $\lambda_u > 1$, we will have $v_j - \lambda_u u'_j < 0$ for $j, u'_j = v_j$. Thus, $A(v - u') \ngeq 0$.

3.3 (4 pts) Prove that all eigenvectors orthogonal to v_1 will have an eigenvalue strictly smaller than 1.

This property shows that if points are viewed as vertices in a Markov graph with transition probabilities proportional to distances between points (elements of W), then v_1 is the only eigenvector needed to compute the probability distribution over states matrix P^∞ (whose $(ij)^{th}$ element shows the probability of being at state j if starting at state i) after infinitely many steps.

Answer: This follows directly from the hint and the property we proved in 3.2.

3.4 (4 pts) Show that $P^\infty = D^{-\frac{1}{2}}(v_1 v_1^T)D^{\frac{1}{2}}$, where $P = D^{-1}W$ is the probability transition matrix.

Answer: $P^\infty =$
$D^{-\frac{1}{2}}(A^\infty)D^{\frac{1}{2}} =$
$D^{-\frac{1}{2}}(\lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T + \ldots)^\infty D^{\frac{1}{2}}$.

Since the eigenvectors of a symmetric matrix A can always be made orthonormal, the dot-product of any two distinct eigenvectors will be 0 and the dot-product of same eigenvectors will be 1. Hence, $(\lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T + \ldots)^\infty = (\lambda_1^\infty v_1 v_1^T + \lambda_2^\infty v_2 v_2^T + \ldots)$. Finally, since $\forall i > 1, |\lambda_i < 1|$, $P^\infty = D^{-\frac{1}{2}}(v_1 v_1^T)D^{\frac{1}{2}}$. (Here, we took a bit of a shortcut: in 3.4 we have proved that $\lambda_i < 1$, but the fact that $\lambda_i > -1$ follows exactly from the same arguments as in 3.2 except using $\max_{i,s.t.u_i<0}$ instead of $\min_{i,s.t.u_i>0}$ in the computations of c)

Question 4. Classifiers (40 pts)

In this problem you will implement two common classifiers and use them to predict attribute values for a series of data. The data you will be working with is a series of records containing 4 continuous-valued attributes and one Boolean class attribute. You will be attempting to predict the Boolean attribute value based on the values of the first four.

All implementation should be done in Matlab. For each problem, we specify the prototype of the required function(s). Please follow these prototypes! It should make your life easier and keep your TA's from getting crotchety.

4.1 (12 pts) Create a naïve Bayes classifier with the assumption that each attribute value for a particular record is generated from a Gaussian with μ and σ determined by the class of the record. In other words, if a record is from class 1, then its first four attributes (a_1, a_2, a_3, a_4) will have their values generated from a Gaussian centered at μ_1 with diagonal covariance matrix Σ_1. You're going to use this assumption to classify a series of new records.

Here is the prototype of the matlab function you need to implement:

$$\textbf{function } \textit{percent\_correct} = \textbf{gaussian\_naive\_classify}(\textit{training\_data}, \textit{testing\_data}) \tag{3}$$

training_data is Gaussian-generated input data (with an arbitrary number of rows), where each row contains 4 continuous values and one Boolean value. *testing_data* is of the same format. *percent_correct* is the percent of records from *testing_data* whose classes were accurately predicted.

Report the accuracy of your classifier when using the training data given in 'ind_training_data.txt' and testing data in 'ind_testing_data.txt'.

Answer: The classifier should have an accuracy of 96%

4.2 (13 pts) Create a joint classifier with the assumption that each record has its attribute vector generated from a single multi-dimensional Gaussian with mean vector and covariance matrix determined by the class of the record. So, if a record is from class 1, then its attribute vector (a_1, a_2, a_3, a_4) is generated from a Gaussian centered at μ_1 with a full (non-diagonal) covariance matrix Σ_1.

Here is the prototype of the matlab function you need to implement:

$$\textbf{function } \textit{percent\_correct} = \textbf{gaussian\_joint\_classify}(\textit{training\_data}, \textit{testing\_data}) \tag{4}$$

again, *training_data* is input data (with an arbitrary number of rows), where each row contains 4 continuous values and one Boolean value. *testing_data* is of the same format. *percent_correct* is the percent of records from *testing_data* whose classes were accurately predicted.

Report the accuracy of your classifier when using the training data given in 'dep_training_data.txt' and testing data in 'dep_testing_data.txt'.

Answer: The classifier should have an accuracy of 98%

4.3 (15 pts) Now find a general algorithm for performing classification that assumes each attribute is generated independently, but each from a different arbitrary distribution based on the class of the record. Your approach should use a set of training data to optimally split each attribute's continuous space into a collection of non-overlapping intervals, with each interval having an associated class prediction.

For each attribute, you should find the number of intervals (at most 3) that provides the best class prediction over the training data. These intervals can then be used for classifying new records for testing.

[Hint: from training data we can compute $P(class = 1 \mid a_i \in X)$ for each interval X associated with attribute a_i. When given a new record, we can use the independence assumption to multiply these probabilities together and see which class is more likely overall.]

Here is the prototype of the matlab function you need to implement:

$$\textbf{function } \textit{percent\_correct} = \textbf{general\_naive\_classify}(\textit{training\_data}, \textit{testing\_data}) \tag{5}$$

again, *training_data* is input data (with an arbitrary number of rows), where each row contains 4 continuous values and one Boolean value (as in training_data.txt). *testing_data* is of the same format. *percent_correct* is the percent of records from *testing_data* whose classes were accurately predicted.

Report the accuracy of your classifier when using the training data given in 'ind_training_data.txt' and testing data in 'ind_testing_data.txt'. Also report the number of intervals used for each attribute. Why is this number not the same for each attribute, and what does this mean about the underlying Gaussian distributions used to generate each attribute's value? How does the performance of this approach compare with the naïve Gaussian classifier from 4.1? Why is it different/the same?

Answer: The method hinted at is as follows.

For each attribute a_i (from $a_1 to a_4$), do the following:

(a) Sort the records by attribute a_i.

(b) Find the best value of a_i to use to split the records into two intervals. To do this, manually try each possible value given in the records as the split value s_i, and look at the classification accuracy when all records with $a_i < s_i$ are assigned the class 1 and all records with $a_i > s_i$ are assigned the class 0 (and vice versa).

(c) Do the same for two split points.

(d) Select the split point(s) and classification which gave the maximum accuracy. Record the classification accuracies: $P(class = 1|a_i \leq s_i)$, $P(class = 1|a_i > s_i)$.

Once we have these probability measures, we can predict the class of any new record pretty easily. The math (for two attributes) is as follows:

$$
\begin{aligned}
P(class = 1 | a_1 \leq s_1 \wedge a_2 > s_2) &= P(a_1 \leq s_1 \wedge a_2 > s_2 \wedge class = 1) \\
&= P(a_1 \leq s_1 | a_2 > s_2 \wedge class = 1) \cdot P(a_2 > s_2 | class = 1) \\
&\quad \cdot P(class = 1) / P(a_1 \leq s_1 \wedge a_2 > s_2) \\
&= [P(a_1 \leq s_1 | class = 1)] \cdot P(a_2 > s_2 | class = 1) \\
&\quad \cdot P(class = 1) / P(a_1 \leq s_1 \wedge a_2 > s_2) \\
&= [P(class = 1 | a_1 \leq s_1) \cdot P(a_1 \leq s_1) / P(class = 1)] \cdot \ldots \\
&= P(class = 1 | a_1 \leq s_1) \cdot P(class = 1 | a_2 > s_2) / P(class = 1)^2
\end{aligned}
$$

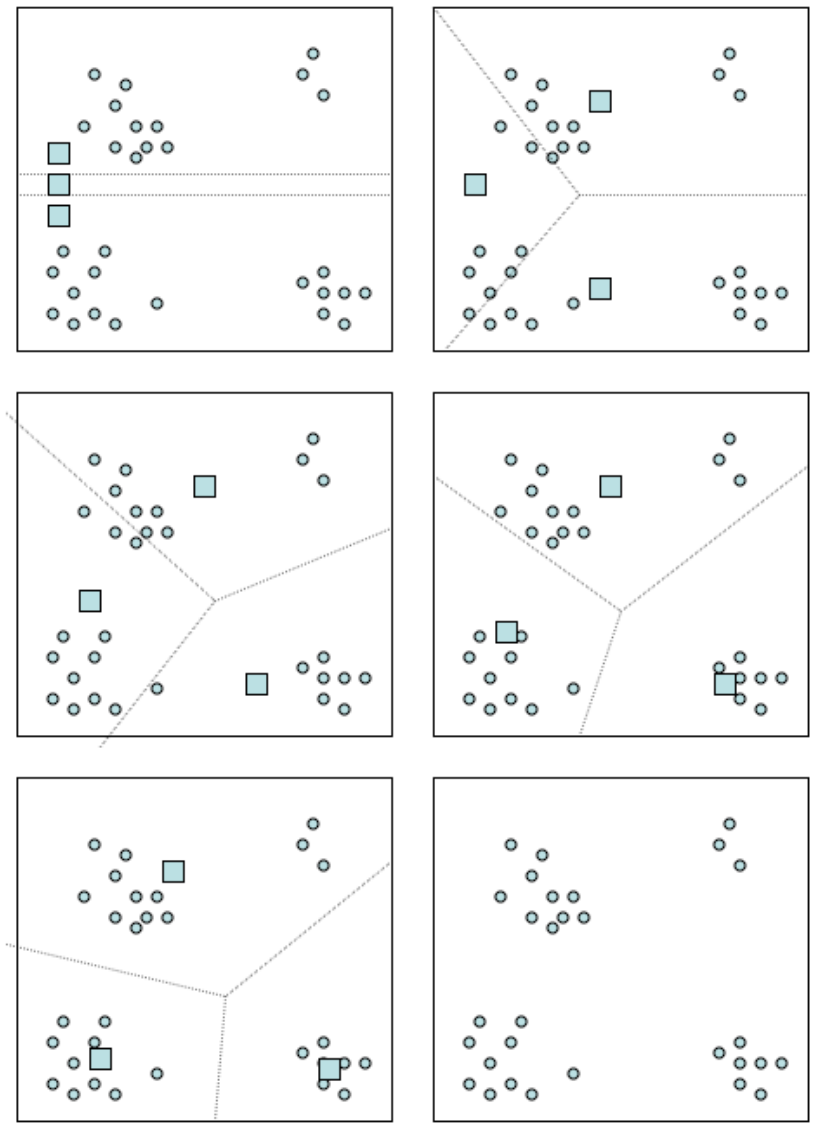
And since we have $P(class = 1) = P(class = 0)$ in our training set, we can remove the denominator, so that we can just multiply the individual $P(class = 1 | a_1 \leq s_1)$ terms together and see which class gives the higher overall result.

If the method hinted at was implemented, the classifier should have an accuracy of 82%. 2 intervals are used for attributes 1, 3, and 4, and 3 intervals are used for attribute 2. This number differs depending on how the two Gaussians used to generate attribute a_i interact. If the variance of each Gaussian (class 1 and class 0) differs, it is possible that one class could "straddle" the other, so that one class is more likely near its mean but the other class is more likely on either side of this. This results in 3 intervals. If the variance of each Gaussian is the same, then only two intervals will be needed.

The interval classifier does worse than the independent Gaussian classifier because we are assuming uniform probability across each interval. The Gaussian classifier is able to distinguish between records within the same interval and assign them different probabilities of classification. In the same way, it is also able to blur the boundaries of the intervals.

Question 5. K-means and Hierarchical Clustering (15 pts)

5.1 (5 pts) Run K-means manually on the following dataset. Circles are data points and squares are the initial cluster centers. Draw the cluster centers and the decision boundaries that define each cluster. If no points belong to a particular cluster, assume its center does not change. Use as many of the pictures as you need for convergence.



5.2 (5 pts) Using hierarchical clustering (with the minimum distance criteria), what is the maximum height of the hierarchy tree required to cluster a set of N points? What is the minimum height? Give an example of a collection of 16 points that requires (i) a hierarchy tree of maximum height, and (ii) a hierarchy tree of minimum height.

Answer: Worst case: $N-1$. Best case: $ceil(log_2 N)$. Example of set of points requiring hierarchy tree of maximum height: points on a line, where separation between adjacent points increases (e.g. on x axis with x values $0, 1, 4, 9, 16, 25, \ldots$). Example of set of points requiring hierarchy tree of minimum height: $1, 2, 4, 5, 8, 9, 11, 12, 101, 102, 104, 105, 108, 109, 111, 112$.

5.3 (a) (3 pts) Is it possible to have Gaussian Mixture Models exhibit equivalent behavior to K-means by restricting the Gaussians used? How?

Yes. If we restrict the covariance matrices to be diagonal, with each diagonal element being the same σ^2 value (with $\sigma \to 0$), then Gaussian Mixture Models will behave like K-means.

(b) (2 pts) For what sort of data do general Gaussian Mixture Models produce much better results than K-means? Provide an example of such a dataset.

Answer: Any data that contains non-spherical clusters. Two long skinny rectangular classes is a classic example.