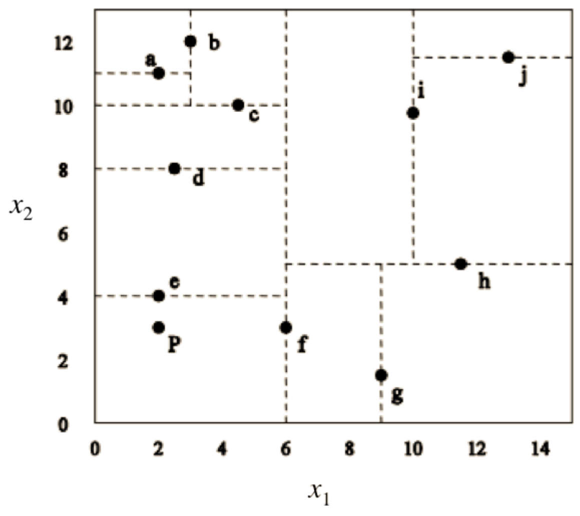
Homework Assignment #2
Due Wednesday, 10/28

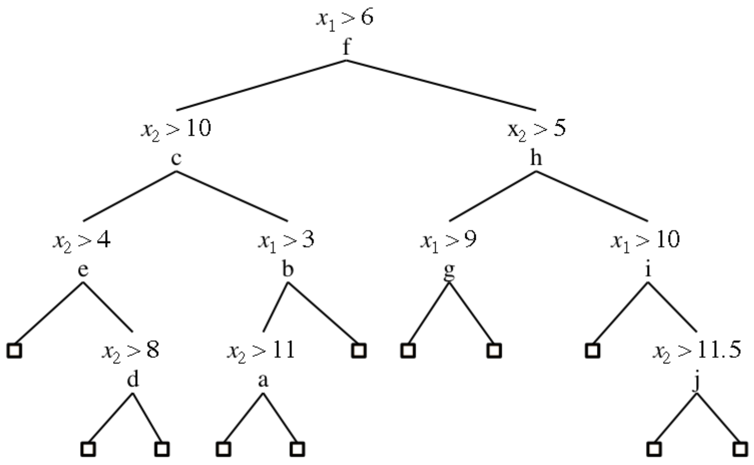
Part 1

1. Consider using single nearest-neighbor for a task with two continuous features that both have the range [0, 10]. Suppose you are given the following three training instances. Show the decision regions (their boundaries and associated class labels) induced by these three instances when Euclidean distance is used to identify the nearest neighbors for test instances.

x_1	x_2	y
4	4	pos
4	6	pos
6	2	neg

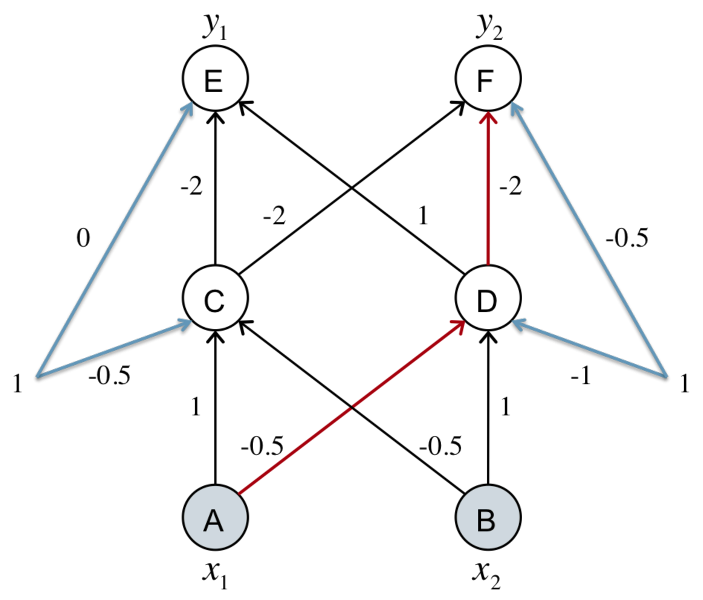
2. Using the k-d tree and the training set displayed in the figure below, show how the nearest neighbor for $\mathbf{x}^{(q)} = (7, 10)$ would be found. For each step in the search, show the distance to the current node, the best distance found so far, the best node found so far, and the contents of the priority queue. You should use Euclidean distance and assume that the coordinates of the instances in the figure below are at integer values.





3. Show a neural network that represents the logical function $y = (x_1 \wedge x_2) \vee (x_3 \wedge x_4)$. Specifically, show the network topology, weights and biases. You should assume that hidden and output units use sigmoid output functions, and an output-unit activation of 0.5 or greater represents a *true* prediction for y.

4. Given the neural network below, calculate and show the weight changes that would be made to the two weights shown in red ($w_{D,A}$ and $w_{E,D}$) by one step of the online version of backpropagation for the training instance shown. Assume that the hidden and output units use sigmoid functions, the network is being trained to minimize squared error, and the learning rate is 0.1 (with no momentum term). Blue weights in the figure indicate bias parameters.



training instance: $\boldsymbol{x}$ = [0, 1] $\boldsymbol{y}$ = [1, 0]

Part 2

For this part of the homework, you are to implement a program that learns a single-layer neural network using *stochastic gradient descent* (on-line training). Specifically, you should assume:

- Your code is intended for binary classification problems, and therefore the network has one output unit with a sigmoid function.
- All of the features are numeric, and your network uses one input unit to represent each feature.
- Your network does not have any hidden units.
- A momentum term is *not* used during training.
- All weights and the bias parameter are initialized to the value 0.1.
- Stochasic gradient descent is used to minimize squared error.

As in Homework #1, your program should read files that are in the ARFF format. Your program needs to handle only numeric attributes, and simple ARFF files (i.e. don't worry about sparse ARFF files and instance weights). Example ARFF files are provided below. Your program can assume that (i) the class attribute is binary, (ii) it is named 'Class', and (iii) it is the last attribute listed in the header section. You should treat the second value listed for the 'Class' attribute as the positive class.

For the next two questions, you should use sonar.arff for your training and test data. This data set is from the task of classifying sonar signals according to whether they are bouncing off a rock or a mine. The features in this data set represent energy within particular frequency bands when the signal bounces off a given object, and the class represents whether the object is a rock or a mine.

5. Using **stratified** 10-fold cross validation, make a set of graphs showing the average training- and test-set accuracy after 1, 10, 100, and 1000 training epochs (i.e. passes through the entire training set). **When the *activation* on the output unit (i.e. the value computed by the sigmoid) > 0.5, you should treat this as a prediction for the 'Mine" class.**
6. Plot an ROC curve for a run of **stratified** 10-fold cross validation with a learning rate of 0.1 and 100 training epochs. You should pool the classifications from the 10 test sets to make one curve. Use the activation of the output unit as the measure of confidence that a given test instance is positive. You should consider 'Mine' to be the positive class.
7. Make a version of your program that can be called from the command line. Your program should be called `neuralnet` and should accept four command-line arguments as follows:
   ```
   neuralnet <data-set-file> n l e
   ```
   where `n` is the number of folds for cross validation, `l` is the learning rate, and `e` is the number of training epochs.

   Your program should do stratified cross validation with the specified number of folds. As output, it should print one line for each instance (in the same order as the data file) indicating (i) the fold to which the instance was assigned (1 to `n`), (ii) the predicted class, (iii) the actual class, (iv) the confidence of the instance being positive (i.e. the output of the sigmoid).

   If you are using a language that is not compiled to machine code (e.g. Java), then you should make a small script called `neuralnet` that accepts the command-line arguments and invokes the appropriate source-code program and interpreter, as you did with HW #1.

Submitting Your Work

You should turn in your work electronically using the course Moodle.

Turn in all source files and your runnable program as well as a file called hw2.pdf that shows your answers to questions 1-6. All files should be compressed as one zip file named `<Wisc username>_hw2.zip`. Upload this zip file as Homework #2 at the course Moodle.