1. Create two datasets of about 200 examples where each instance \( \mathbf{x} \) has 10 numeric (0-1 valued) features and a 0-1 valued label. For the first dataset, let the label \( t \) be the XOR of the first two features. For the second dataset, let the label be the majority vote of the first three features. The MakeARFF.java program is one way to do this. Note that the second dataset is linearly separable while the first is not. For the following experiments use a 50% train/test split.

(a) Use Weka to learn two decision trees, one for each dataset. Compare the complexity of the two trees (depth, number of nodes). Can you interpret the decision trees? What happens if you adjust your data file for the xor problem so that each combination of the values of the two relevant features (ie. 00, 01, 10, and 11) appear equally often?

(b) Use Weka to train two neural networks (say with 2 hidden nodes), one for each dataset. How many epochs are required for convergence for each dataset? (You can use the training time parameter to set the number of epochs, and the hidden-layers parameter to "2" to get one hidden layer of 2 units, try as few as 5 or 10 epochs.)

(c) Use Weka to train a support vector machine (use the SMO algorithm) on these two data sets. Use polynomial kernels of degree 1 and compare with polynomial kernels of degree 2, and polynomial kernels of degree 2 with lower order terms turned on. (You can select the kernel as a parameter to the SMO algorithm). Which of the three settings is best, and why?

2. Read the handout on the back-propagation algorithm. Assume the function \( \sigma(\cdot) \) is used at the hidden nodes, and output node 5 uses the identity function. Write a single mathematical formula for the value produced by the network drawn in Figure 1 of the handout. Your formula should use the sigma function \( \sigma() \), the weights (like \( w_{53} \) for the weight at node 5 given to the output of node 3), and the values of the input nodes (\( z_0 \), \( z_1 \), and \( z_2 \)).