Department of Mathematics and Computer Science
University of Southern Denmark, Odense

December 16, 2009
Marco Chiarandini

DM533 - Introduction to Artificial Intelligence

Assignment 3, Fall 2009

The submission deadline for this assignment is **11.00 of Tuesday, December 22**.

For submission instructions, refer to Assignment 0. Although I will not fail anyone on this assignment, all exercises are worth doing, as they contain elements that may arise at the exam. This assignment assumes knowledge from chapter 18, 20 of the course book.

1. Exercise 18.13 of the course book. No need to implement anything, just give the mathematical expression.

2. Consider the application of a perceptron network in classification. For example, consider the application in flower classification. The Fisher's iris data set gives the measurements in centimeters of the variables petal length and sepal width, respectively, for 50 flowers from each of 2 species of iris. The species are "versicolor", and "virginica".

   Your task is to develop a perceptron that is able to classify the last two of these categories. More specifically, you must apply the Perceptron Training Algorithm to learn the weights for a perceptron with x_1,x_2 and *bias* as inputs and $-1, +1$ as output. The output -1 corresponds to the classification as "versicolor" and 1 to the classification as "virginica".

   Use *k*-fold cross validation to assess the quality of the prediction returned by the perceptron. A value of *k*=10 is fine. Report the average rate of misclassified examples. Further, explore graphically the line separators produced and check how misclassification would vary by using the sign or the sigmoid function and by changing *k*.

   You may use python or R (the latter being a prefereable choice). For python you find something that you may need implemented in `learning.py` of the aima repository. For R you find a script next to the link where you downloaded this assignment.[1]

   If you used *R* you can compare your implementation against several already implemented methods:

   - function `nnet` from `library(nnet)` implementing a neural network approach (trying adding hidden layers),
   - function `rpart` from `library(rpart)` implementing a regression tree approach,
   - function `ctree` from `library(party)` implementing a regression tree approach,

[1] Note that in R the scalar product is represented by %*% and the modulo operator by %%.

- function knn from library(class) implementing k-nearest neighbor classifier.

3. Consider Neural Networks with inputs in the range $[0, 1]$ and with a step function g. A network is defined by the weights on the links and a threshold value of g at each node.

   (a) In Boolean logic, the majority function is a function from n inputs to one output. The value of the operation is false when $n/2$ or more arguments are false, and true otherwise. Draw a network that represent the majority function for 4 input nodes.

   (b) Draw a network that represent the "exactly two out of three" function for three inputs.

   (c) Draw a network to simulate the XOR operator in Boolean logic. XOR (exclusive-or) is a logical operator that results in the output being true if one of the inputs, but not both, is true. If both inputs are true the output is false.