Homework #3, EE556, Fall 2018

Due: 9/27/18; hand in Problems 1, 2, and 6

Problem #1

A classifier uses four linear discriminant functions in the plane: $g_1 = x_1$, $g_2 = x_2$, $g_3 = x_2 + x_1 - 3$, and $g_4 = x_2 - x_1 + 1$.

i) Sketch the line boundaries in pattern space and label each region with a 4-bit binary codeword.

ii) Make a logical table, identifying for each codeword whether or not there is an associated region.

iii) Sketch the **four**-dimensional hypercube in state space. (**Don't panic !** Use two 3-dimensional projections, one yielding a 3D cube for the case $T_1 = 0$ and the other giving a 3D cube for $T_1 = 1$.). Label each vertex with its corresponding binary codeword. Draw a curved line joining each vertex in the first cube with the corresponding vertex in the second cube.

ii) Consider the discriminant function $y = \text{sgn}(\sum_{i=1}^{4} T_i - 2.5)$. Sketch the decision region induced by this rule in the (2D) feature space.

Problem #2

Construct a multilayer perceptron that solves the N-bit parity problem. (Recall this problem from homework 2).

Problem #3

i) Consider an MLP with I inputs, J hidden units, and K output units (a single hidden layer). What is the space complexity of the network ? (Include the the storage required for MLP parameters, training data, and and any additional storage needed during training).

ii) What is the computational complexity of backpropagation training in batch gradient descent mode ?

Problem #4

Consider a standard multilayer perceptron. Show that if the sign on every weight is flipped the operation of the network remains unchanged (a type of "polar symmetry") – does this property extend to changing the signs of the inputs to the network ?

Problem #5

Derive the learning rule for updating an input-to-hidden unit weight for a single-hidden layer MLP that uses a "softmax function" in the output layer and the cross entropy criterion for training (both "softmax" and "cross entropy" are discussed in lecture).

Problem #6

Consider the problem of neural network *inversion*, wherein, given a fixed network and target output values, the objective is to learn the associated *input* patterns which, when forward-propagated through the network, produce outputs that well-approximate the targets. Sketch a method (an optimization technique ?) that approximately achieves NN inversion. Also suggest some possible applications for NN inversion.