Name	______________________
Date	______________________
Lab teacher	______________________

Demo Lab: Neural Network and Learning Systems

Lab

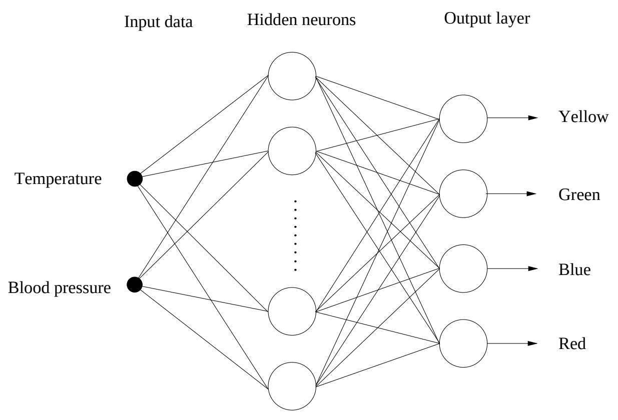
January 15, 2012

The purpose of this lab is to give an overview of the methods used in the course.
In order to run the lab, you have to download a package of Matlab files from the course home site (http://www.imt.liu.se/edu/courses/TBMI26/), unpack the files and initiate the course in Matlab. To initiate the course you run the command:

```
>> init_nnkurs
```

Backpropagation

Error backpropagation is an example of supervised learning. In this part you will learn a "backprop"-network to diagnose different diseases based on measurements from blood pressure and temperature. The image below shows the topology of the network:



You will by yourselves simulate (by clicking in the GUI) the measurements of blood pressure and temperature on patients suffering from one of these common diseases *Yellow*, *Green*, *Blue* or *Red*. The measurements are presented for the network which will learn to indicate the correct disease. You can control the following parameters: the number of hidden neurons, how fast the network is allowed to change (the learning rate/velocity), the strength of the connections between the neurons (the weights) and the number of times the network are allowed to train on the simulated data (the number of epochs). The strength of the connections (the weights) are random in the beginning. You start the program with the command

```
>> bp_diagnost
```

Question: How does the number of hidden neurons affect the result?

Question: What happens if the learning rate is too large? Too small?

Question: Is there a disadvantage allowing many epochs?

Question: Do the parameters affect each other? Do you, for example, have to decrease the learning rate when increasing the number of hidden neurons?

Question: The learning rate in this example is constant over the epochs. Is there a better strategy with changing learning rate?

Question: Are there other applications for a backprop network?

Character recognition

In a later laboratory exercise you will use CCA (Canonical Correlation Analysis) in order to classify characters written with a computer mouse, a pencil or similar. A preview of how a similar kind of character recognition may work is presented here.

Run the function 'numbers' and draw the numbers 0-9 in the presented image window. The program learns how the numbers look like and then, after the training, you can try to draw the different numbers again and observe the classifications estimated by the program. You have one second to continue to draw after you release the mouse button. After one second, the program assumes you are finished with the number. To retrain, abort the program, enter the command 'clear' and restart 'numbers'.

Question: How sensitive is the recognition for variations in character size, position, rotation, drawing speed etc?

Question: How does the number classification perform when the test-numbers and the training numbers are drawn by different persons?

In contrast to OCR (Optical Character Recognition) this method depends on how you draw the characters (downwards or upwards for instance).

Question: In what kind of applications do you think this type of character recognition are applicable?

Tic-Tac-Toe

The purpose of tic-tac-toe is to get three of your own symbols in a row (horizontal, vertical or diagonal) in a 3×3 grid. In this demo program, you can choose to play against an untrained or trained system. In addition, you can choose to train the system yourself or use a finished "reference" system.

When you train the system, it will learn the game states, the moves that connect them and who won (reward). The system then tries to maximize the long term reward at each move. This kind of learning is usually called reinforcement learning and is characterized by problems where you don't know how to move to reach the goal (e.g. to win). This is in contrast to supervised learning where you for each input signal know the corresponding output signal.

Start the program by entering 'tictactoe' on the command line.

Question: How many different game states are there? Are they all legal/obtainable? If you tried to train a system to play 5×5 tic-tac-toe; how would it turn out? 17×17 ?

Question: A challenge when designing a system using reinforcement learning is the *Exploration-exploitation dilemma*. Consider what this mean!

k-Nearest Neighbour

Now it is time for you to implement your very first classifier, k-Nearest Neighbor (kNN). kNN classifies an unknown sample **x** by measuring the distance between **x** and a training dataset containing samples of known classes. The most common class of the k closest training samples is the one given to **x**.

Your task is to fill in the function kNN. Type 'edit kNN.m' to open the function in the matlab editor. The inputs and output of this function are given below.

You can test your classifier by typing 'kNN_Eval' or 'kNN_Eval_Brain' in the command window. These test run rather slowly since kNN() is called thousands of times in order to map the whole space.

A short introduction to matlab can be found on the assignments page of the TBMI26 website.

List of variables

Variable Name	Input/output	Size and type	Description
$\mathbf{x}$	Input	Vector of length M	This is the sample the function *kNN()* shall classify. It is a vector with N features.
$\mathbf{Y}$	Input	Matrix of size $M \times N$	This is your training data. It contains M samples, each of length N.
$\mathbf{c}$	Input	Vector of length N	This vector stores class of each sample in $\mathbf{Y}$. Each class is identified as in integer (1, 2, 3 etc.)
k	Input	Integer	k is a scalar integer describing how many of the closest neighbours of $\mathbf{x}$ that will participate in the voting.
$cOut$	Output	Integer	The output after classifying using kNN.

Question: Explain how the kNN algorithm works, step by step

Question: How does the result change when k grows larger?