Lab assignment 1: linear-threshold classifiers

- Task 1: Program a simple linear classifier
- Task 2: Simple linear classifier for 2D binary problems, manual design
- Task 3: Simple linear classifier for Iris data, manual design

"Manual design" means no training, just trial-and-error looking at graphs.

Describe everything in a report. One figure is required (a graph), others can be added if useful.

Task 1: Program a simple linear classifier

(**NOTE:** This is basically the `linclass` function that you should already have seen in Lab0 (Matlab intro). If you didn't save the `linclass.m`file, now it is time.)

Here you have to create a program (a Matlab function for instance) that takes two parameters:

1. a set of n data, as a $n \times d$ matrix (or possibly $d+1$), observations in rows
2. a row vector $\mathbf{w}$ of $\boldsymbol{d}$ (or $d+1$) coefficients

and outputs a vector $\mathbf{y}$ of n values in $\{-1,+1\}$, each representing the classification output for the corresponding input pattern (row of the input data matrix).

Task 2: Simple linear classifier for 2D binary problems, manual design

Create two data files, i.e., plain-text files in a format suitable for reading by a program, containing the following two-dimensional data sets (just the numbers):

DATASET 1

point		**class**
.12	.1	1
.1	.31	1
.6	.23	1
.4	.5	1
.61	.7	-1
.84	.4	-1
.2	.87	-1

DATASET 2

point		**class**
.27	.78	1
.1	.31	1
.6	.43	1
.4	.5	1
.61	.7	-1
.84	.4	-1
.2	.87	-1

Yes, copy-and-paste should do.

(1) For each data set, plot it to view it. Manually find a solution, a separating hyperplane. Manually means by trial and error, or by a graphical procedure. Then use your linear classifier program with the weights corresponding to that solution, and run it on the data. Report on whether it works, and, if not, why.

(2) Repeat the above procedure, but using weights affected by increasingly intense random perturbations, and check whether it still works:

A random perturbation is random noise added to your data or, as here, your parameters.

To apply a random perturbation to vector data, in Matlab it is handy to create a random vector **n** with a given norm *p*, and to add it to the weight vector. You obtain a random vector **n** of norm *p* as follows:

1. create a completely random vector as n = 2*rand(3)-1;
2. make it unit norm as n = n/norm(n);
3. multiply it by *p*.

Increase the norm *p* from a minimum to a maximum, with a step size that is not too large. Guideline: maximum not larger than 10% the norm of **w,** and 10 steps (what is the step size required to do this?)

For each value of *p*, do some statistics: create a number of perturbations, apply them, run the classifier on the data, and evaluate the number of errors. Guideline: use a fairly high number of random trials, e.g., 100.

For each data set, plot a graph (Excel, Matlab, hand... is fine) of the **average number of errors** vs **perturbation size** *p*. Do the results differ for the two different data sets? How? Is there a specific feature of these sets that can justify different behaviors?

Task 3: Simple linear classifier for Iris data, manual design

Download the simplified Iris data set (two classes, 2 and 3 merged). Design a linear classifier that solves the problem. Test it and describe.

UPLOAD BELOW YOUR CODE, DATA, AND REPORT