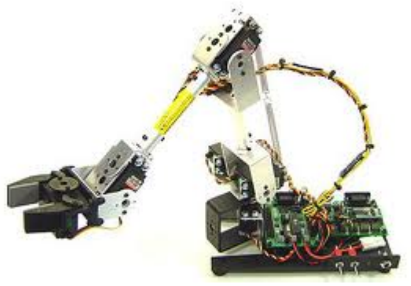
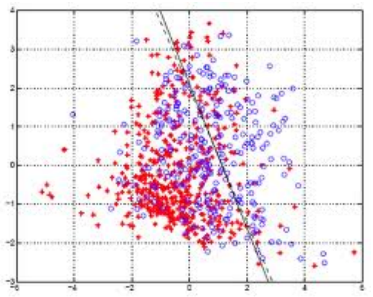


MLNC 2011/12 Assessed Coursework

Machine Learning & Neural Computation



- Issue: 28/10/2011
- Due: Mon 14/11/2011 12am
- Feedback: Fri 18 & 25/11/2011

Familiarise yourself in time on how to submit to CATE.

Question 1: Classification

Build two classifiers for the Bowel Cancer Data set using 1. the k-nearest neighbours and 2. the generative classification approach (a Gaussian per class). For each of the two classifiers do the following:

1. Describe your classifier + supply the Matlab code.

   Write a function *Classifier* that takes as arguments an *input* vector and a matrix/vector of *parameters* and returns the predicted *class*. Call the m-file Classifier.m, it's first line should look like this:

   ```
   function class = Classifier(input,parameters)
   ```

   Write a function TrainClassifier(inputs,output) that takes as arguments an input matrix of data points (each row one data point, the columns correspond to the dimensions of the data point) and the training class labels (desired class labels) as a column vector of numbers. The function should return the parameters required by your Classifier function. Call the m-file TrainClassifier.m, it's first line should look like this:

   ```
   function class = TrainClassifier(inputs, output)
   ```

2. Test it's performance by
   a. Dividing data into equally sized training and test data set. Use only the training data to train your classifier. Report your classifier's performance (% correct) on the test data set.
   b. Do "leave-one-data-point-out" cross validation. Train your classifier with the whole data set, leaving out one test data point. Measure the performance. Repeat this for each data point, report the averaged performance across all runs.

3. Explain what properties of the data are responsible for your classifiers performance.

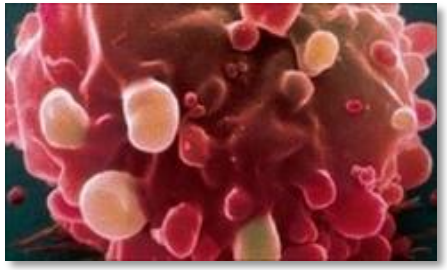
Bowel cancer classification

30 dimensional real valued features,

Two classes: {malign, benign}

Features such as

a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("how jagged is the contour")



Data on CATE as CourseworkData.zip (which contains a Matlab .mat file).

Question 2: Learn the inverse kinematics of a robot arm

A robotic arm's (Fig. A) forward kinematics maps the joint angles Θ_1 and Θ_2 (Fig. B.) to the spatial position of it's end point at location (x,y) (Fig. C). Inverse Kinematic inverts this problem: Given a position the joint configuration is to be found. Note, (x,y) has units of centimeter, (Θ_1, Θ_2) has units of degrees.

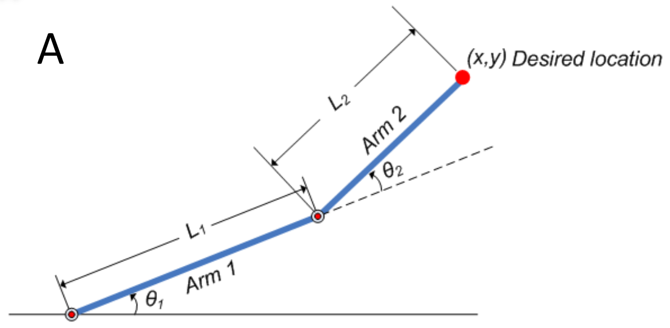
Develop the following MatLab code using linear basis function regression. Note: The code should be documented with sufficient comments so a stranger is able to follow why and what and you did.

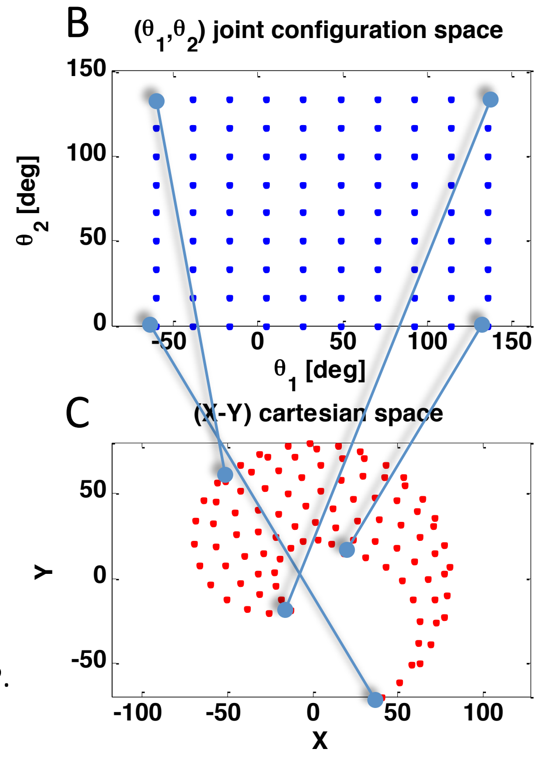
1. A function file *TrainRegressor.m* that takes a 2-column matrix of training data **inputs** and a 2-column matrix of training **outputs** and returns a matrix or data structure of **parameters**.
Note, the first line of your .m-file should look like this:
function **parameters** = *TrainRegressor*(**inputs**,**outputs**)

2. A function file *TestRegressor.m* that takes a 2-d vector **input**, the test data point, and the variable **paremeters** generated by TrainRegressor, it should return a predicted 2-d **output** vector.
Note, the first line of your .m-file should look like this:
Function **output =** *TestRegressor*(**parameters**,**input**)

To generate your data you are provided with the file *GenerateData.m* *(code on next page).*

Use it to obtain your data: [**outputs**, **inputs**] = GenerateData(*numSamples*)
numSamples is the number of data points you have at your disposal for training and testing your regressor. Each training data points costs you 1 GBP.





Submit TrainRegressor.m & TestRegressor.m and how much you want to spend on training data. For the chosen budget, report your test error (mean squared error of difference between predicted and actual angles), how you used test and training data the data cost you are prepared to pay with your learning machine.

```
function [outputs,inputs] = GenerateData(numSamples)

l1 = 43; % length of first arm
l2 = 70; % length of second arm
theta1U = randn(numSamples,1)+rand(numSamples,1)*195-65;
theta2U = randn(numSamples,1)+rand(numSamples,1)*155;
theta1G = randn(numSamples,1)*31 + 27.5;
theta2G = randn(numSamples,1)*20 + 95;
p=0.5;
select = rand(numSamples,1)>p;
theta1 = theta1U;
theta2 = theta2U;
theta1(select) = theta1G(select);
theta2(select) = theta2G(select);

%generate a grid of theta1 and theta2 values
X = randn(numSamples,1) + l1 * cosd(theta1) + l2 * cosd(theta1 + theta2); % compute x coordinates
Y = randn(numSamples,1) + l1 * sind(theta1) + l2 * sind(theta1 + theta2); % compute y coordinates
data1 = [X(:) Y(:) theta1(:)]; % create x-y-theta1 dataset
data2 = [X(:) Y(:) theta2(:)]; % create x-y-theta2 dataset

subplot(2,1,1)
plot(theta1(:), theta2(:), 'b.');
xlabel('\theta_1 [deg]')
ylabel('\theta_2 [deg]')
title('(\theta_1,\theta_2) joint configuration space')
axis equal
subplot(2,1,2)
plot(X(:), Y(:), 'r.');
xlabel('X')
ylabel('Y')
title('(X-Y) cartesian space')
axis equal;

inputs  = [X Y];
outputs = [theta1 theta2];
```