CS 4758/6758 Homework 1

Due: Feb. 15 2013

Please attach any code written to your submission.
Be sure to include your netid on your submission.

1 Kinematics (60 pts.)

1.1 Planar Arm Forward Kinematics (15 pts.)

Figure 1 shows the schematic of a robot arm lying in the X - Y plane. The robot has three links each of length $l_1 = 60$ cm, $l_2 = 40$ cm, $l_3 = 30$cm. The first link of the robot is located at point (0,0) on the X - Y plane. The angles at each of these joints are θ_1,θ_2, and θ_3

Given θ_1,θ_2, and θ_3, compute the position of the robot's hand (X_{hand}, Y_{hand}). (a) Make rotation matrix for each of these links. (b) Write the homogenous transformation matrices T_1^0, T_2^1, and T_3^2. (3x3 matrix: rotation matrix appended with translation) (c) Write the expression for obtaining(X_{hand}, Y_{hand}) as a function of homogenous tranformation matrices. (d) Evaluate the expression for $\theta_1 = 30^o, \theta_2 = 45^o, \theta_3 = 45^o$ and report the (X_{hand}, Y_{hand}).

h

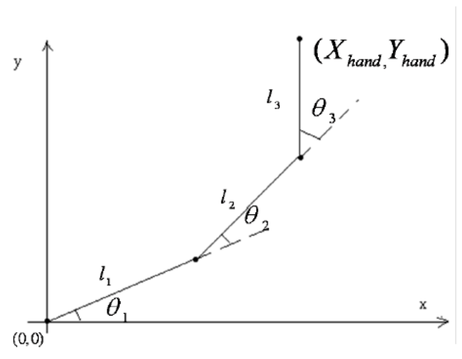


Figure 1: A planar robot arm with three links

1.2 PR2 Forward Kinematics (Programming, 15 pts.)

We will now use ROS to run forward kinematics PR2's arm. This arm is much more complex - it consists of 7 degrees of freedom (specified as r_shoulder_pan, r_shoulder_lift, r_upper_arm_roll, r_elbow_flex, r_forearm_roll, r_wrist_flex, r_wrist_roll). So, we'll use ROS's pr2_kinematics stack to compute the forward kinematics for us.

The tutorial at: http://www.ros.org/wiki/pr2_kinematics/Tutorials/Tutorial%203 will probably be useful for this part. Give end-effector coordinates in the form x, y, z, and orientations in quaternion form (the form output by ROS's forward kinematics) of the r_wrist_roll link.

Compute the gripper cartesian position and orientation in torso_lift_link frame for following value:

```
r_shoulder_pan: -0.3
r_shoulder_lift: 0
r_upper_arm_roll: -1.5
r_elbow_flex: -0.9
r_forearm_roll: 3.1
r_wrist_flex: -0.5
r_wrist_roll: -1.6
```

1.3 PR2 Inverse Kinematics (Programming, 30 pts.)

Now, write a program which performs inverse kinematics for PR2. For this question, we will ignore gripper orientation, so that the goal is merely to find a set of joint parameters, as specified above, which position the gripper r_wrist_roll link at a set of xyz coordinates.

You are not allowed to use any existing inverse kinematics code, but you also don't need to solve the inverse kinematics for PR2 directly. Instead, you can use data taken from your forward kinematics program in the previous part and the nearest-neighbor algorithm to find joint angles for a particular point.

In detail, your program could use your forward-kinematics code from the previous part to compute and record the gripper position for a series of arm configurations. Then, when you want to find joint parameters for a new set of end-effector ones, you could find the nearest neighbor to the given position in the recorded data, and use its corresponding joint parameters as a starting point for gradient descent to fine-tune the solution (using numerical methods since we don't have an analytical form for the gradient).

Give the cartesian position of r_wrist_roll in torso_lift_link frame given below. Verify that these parameters give the desired position using your forward kinematics code from the previous part.

```
a) 0.821,-0.188,0
b) 0.742415,-0.155112,0.0177867
c) -0.0174545,0.429794,0.436441
d) 0.182538,-0.188,-0.628473
e) 0.324203,-0.453238,0.61856
```

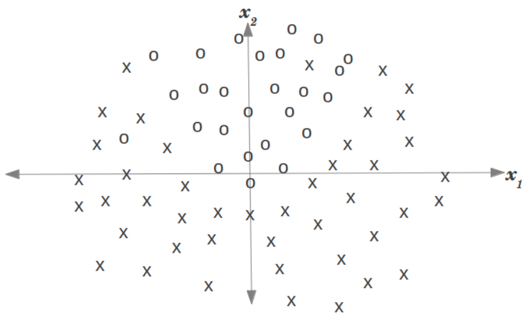
2 Classification, Regression, and Generalization (40 pts.)

2.1 Classification (15 pts.)

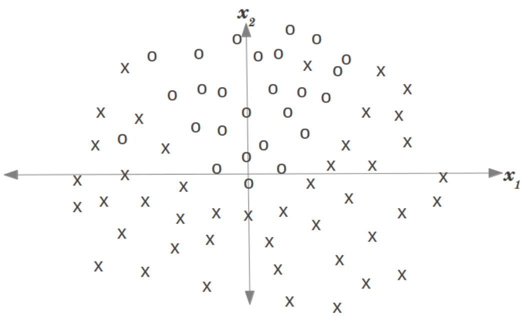
For the following set of points, where X's indicate positive cases and 0's negative, draw decision boundaries for logistic regression and nearest neighbors with k=1 and k=3 (you can do this on the plots below, or use the attached .png image in your report).

What do you notice about the performance of each algorithm? Which one do you think would generalize better to new cases for this data?

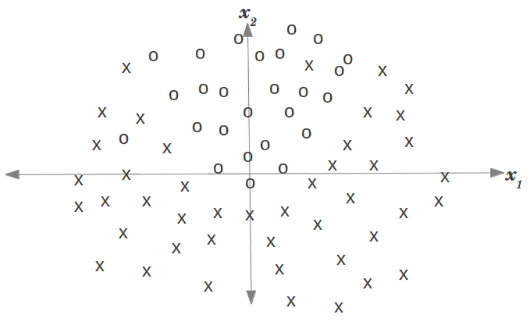
Logistic



1-NN



3-NN



2.2 Regression (Programming, 25 pts.)

Suppose we have some data x_1 and x_2 taken from two sensors and y is an output value, unknown at runtime. For this problem, the training data is given in trainData.txt as 100 points of the form x_1, x_2, y. The hold-out testing data in testData.txt is in the same format and taken from the same distribution.

Your goal is to estimate the value of y given x_1 and x_2. Write a program which fits polynomials of degrees 1, 2, and 3 to the data (x_1, x_2) to compute y. You can do this by computing each power of each x up to the given degree as an input feature and then running linear regression from these features to y. Be sure to include a constant bias term as well.

Run your program for the first k data points from the training set, for k from 5 to 100 in increments of 5. Plot the mean squared error (total squared error divided by number of points) for estimating y on the training and testing sets vs. k for polynomials of order 1,2, and 3.

What do you notice about the error and fits for each order of polynomial and how they change as we increase the number of data points?