CS 534

Assignment #2: latents and probability

Due: March 20 @ 11:59 p.m.

# Part 1: Kalman filters

Your goal is to develop a Kalman filter to estimate the number of homeless people in Worcester. Counting the number of homeless is difficult as they are transient, and may not want to be found. As a result, it can be difficult to get an accurate count. For an example of the issues, see [As I See It: Worcester must address homeless issue once and for all](https://www.telegram.com/opinion/20180304/as-i-see-it-worcester-must-address-homeless-issue-once-and-for-all) (the professor does not necessarily subscribe to the views in the article, but is giving it as some clues for thinking about the Kalman filter aspects of the problem).

For this question, you will not program and should perform the calculations “by hand.” Code to multiply matrices is fine, but simply finding someone’s code for Kalman filters and running it is not. You are to show the steps of your work. The point of this question is to have you understand the inputs to a Kalman filter and how it updates its estimates. **Note: this question goes beyond the simple examples of Kalman filters we did in class.**

## Sensor and transition models

You will need to construct sensor and transition models for this problem. The transition model should take as input the homeless population at time **t-1**, and a predicted change in homelessness. You will have to decide how much variability to include across time steps. You can assume there are simply 3 actions: decrease the number of homeless, increase the number of homeless, and no change. These actions are trivial, but are designed to let you think about how actions influence the Kalman filter estimate process.

For the sensor model, you must come up with at least 2 sensors for estimating the number of homeless at time **t**. You should spend a few minutes thinking up plausible sensors (the linked article may give you some ideas). The key bit is whether the sensors are feasible, and the constructed model makes sense.

You can assume in March 2020 there is a homeless population in Worcester of 1200.

## What you will do

You are working as an intern at a nonprofit that wants estimates of the Worcester homeless population. You will work as an intern from March 2020 through August 2020. At the end of each month, your boss wants monthly progress reports with an estimate of the homeless population. You will need to pretend your sensors are working and create sensor estimates for each month, and use your Kalman filter to create an estimate of the homeless population each month. Create the monthly progress reports. To help your boss (and the grader) understand what you’re doing, write an explanation of the process for March 2020 (i.e., explain how all pieces of the model fit together).

At the end of your internship in August, your boss wants a final report containing two items:

1. A revised estimate of the homeless population for March 2020 - August 2020. Since you have 6 months of data, you should be able to better estimate the population for the intervening months. Explain the process for how you computed your revised estimate for April 2020.
2. Your boss would like a projection for the homeless population for the two months after you left (September and October). For your transition model, you should assume the new Mayor will adopt policies that cause the number of homeless to increase.

For all parts of your report, you will need to be clear about what are sensor readings and what is the estimate of the latent state.

# Part 2. Expectation maximization

## Basic functionality

Your program will take as input a set of (n-dimensional) data points that come from 1 or more clusters, and the number of clusters to find. For example:

em data.txt 3

would read in the points from data.txt, and return the best 3 cluster centers it can find.

You will use expectation maximization with random restarts to return the best-fitting cluster centers. Your program should output the best-fitting cluster centers (mean and variance), as well as the log-likelihood of the model.

You may assume the data points are generated from an n-dimensional Gaussian where each dimension has a differing mean and variance. You may assume the dimensions are independent of each other. A .csv file with the correct format has been posted under the assignment directory. In addition, a google doc with the cluster labels is available: [EM sample data](https://docs.google.com/spreadsheets/d/19aqOjpOQbRouk2xaQoYZkv6Atcjq1bx6iWDfnUnKaBg/edit?usp=sharing) Note that your program will **not** have access to cluster labels, and will only know the x,y, z, ... coordinates of the data. This file is just to help you understand what your program is *ideally* recovering from the data (with a bit of luck). Note that clustering algorithms will often be unable to recover the correct cluster membership of every data point, and for difficult cases may not recover the correct number of clusters. Estimating a latent means never being able to say you’re certain.

## Determining the number of clusters

Once you have your basic program working, you should extend it so that it takes input such as:

em data.txt 0

Since 0 clusters makes little sense, your program should automatically determine the best number of clusters in the data and output their center. Note that you can achieve a perfect fit to the data by having the number of cluster be equal to the number of datapoints (i.e., each point is its own cluster and is perfectly described by the cluster center). However, such a degenerate model is not what we are looking for. Instead, learn about BIC and use it as a means for guiding model fitting. Your program should determine which number of clusters best fits the data according to BIC, and output the number of clusters, log-likelihood, BIC, and cluster centers.

## Allocating time

To continue our theme of allocating time efficiently, you will be given 10 seconds to perform your EM search. Test files may be much bigger than the sample file. You should decide how to allocate time between:

1. Continuing an iteration of hill climbing to get better estimates of the cluster centers.
2. Performing a random restart to avoid falling into a local optimum.
3. Confirming that you have the correct number of clusters. This issue is only a concern if a “0” is passed in. If your program is given a number of clusters it should assume that number is correct.

You will have to determine how to best use your available 10 seconds. As usual, you don’t need to do complicated interrupts. Just stop the program at the top of the loop if more than 10 seconds have elapsed.

## Writeup

1. How did you initialize the cluster centers? Was it random? Did you methodically walk through the search space? Something else?
2. Explain how you used BIC (Bayesian Information Criterion, see https://en.wikipedia.org/wiki/Bayesian\_information\_criterion) as a modeling fitting criteria and how you used it to terminate your search.
3. How did you trade off: number of iterations, random restarts, and determining the number of clusters? Explain your approach for when to stop doing each of them.
4. Create a data file whose clusters are not easily separable. How does EM perform? Specifically, how accurate is it in determining the correct number of clusters? If given the correct number of clusters, does it find the correct means and variances of the clusters? Does it assign points to the correct cluster? Answer the same questions for the provided sample data file.
5. Sketch log-likelihood vs. # of iterations for a data set. Run it for longer than your typical termination criteria, and mark where the algorithm would normally stop on the graph. Provide the parameter estimates for where your program would normally stop, and what it would find if it kept going. Does convergence of log-likelihood correspond to convergence of model parameter estimates?

# Hints

1. Part 1 does not involve programming, but is conceptually more difficult than Part 2.
2. Part 2 requires some experimentation and analysis. Get basic EM clustering working *quickly*, then scale up to determining the number of clusters and trading off time.