Problem 1: Chow-Liu Trees

In this problem, we will build a tree-structured graphical model for rainfall events measured at 30 weather stations in Australia (data from TODO CITE). We will use the graph to capture the spatial correlation in the data. (Note that the data are also temporally correlated, but for this model we will ignore this aspect and treat the measurements as i.i.d.

(a) Load the rainfall data from `data/data.txt`:

```
import numpy as np
# Load the data points D, and the station locations (lat/lon)
D = np.genfromtxt('data/data.txt',delimiter=None)
loc = np.genfromtxt('data/locatons.txt',delimiter=None)
m,n = D.shape       # m = 2760 data points, n=30 dimensional
# D[i,j] = 1 if station j observed rainfall on day i
```

(b) Compute the empirical mutual information score for each pair of locations,

$$\hat{I}(x_j, x_k) = \sum_{x_j, x_k} \hat{p}(x_j, x_k) \log \left[\frac{\hat{p}(x_j, x_k)}{\hat{p}(x_j)\,\hat{p}(x_k)} \right]$$

where $\hat{p}$ is the empirical probability distribution.

Note: you may find it helpful to use my provided Python `factor` class, for manipulating tabular representations of functions. For example,

```
import pyGM as gm
X = [gm.Var(i,2) for i in range(n)]    # define binary random variables
p01 = gm.Factor([X[0],X[1]],0.0)       # create empty factor over binary x0,x1
```

Several pages of example usage can be found on either our course page, or CS179 (Fall 2015). There is also Matlab/Octave versions of the code, similar but not identical in interface; some basic documentation for the Matlab version is at `http://sli.ics.uci.edu/Code/Matlab-Factor`. However, if you prefer, you do not need to use my code to answer this question.

(c) Use your favorite maximum weight spanning tree algorithm to find the maximum likelihood tree structure. Report the edges that are included in your model, and create a visualization of the spanning tree by plotting an edge between the lat/lon locations of stations j and k if the edge is included in your model.

(d) Report the average log-likelihood of your model, $\mathcal{L} = \frac{1}{m}\sum_i \log p(x^{(i)})$.

Problem 2: Iterative Proportional Fitting

Next, let's train a loopy model on the same data. The file `edges.txt` contains a list of 51 edges forming a simple loopy model of the data.[1]

(a) Plot the edges of the loopy model along with the Chow-Liu tree you found in Problem 1 and compare.

(b) Now, find the empirical marginal probabilities of each edge in the loopy model, similarly to in Problem 1.

(c) Now, perform iterative proportional fitting on the model. In each iteration, for each edge, compute the marginal probability of the current model over that pair of variables, and update your factor to cause your model to match the empirical statistics, e.g.,

$$p(X_j, X_k) \propto \sum_{x \setminus [X_j, X_k]} \prod_{e \in E} f_e(x_e)$$

and

$$f_{jk}(X_j, X_k) \leftarrow f_{jk}(X_j, X_k)\hat{p}(X_j, X_k)/p(X_j, X_k)$$

At each step, also compute the log-likelihood of data under the current model.

Again, you may find my Matlab or Python code helpful for this exercise; in particular, the GraphModel class can be used to find an efficient elimination ordering for the model, and compute the marginal probabilites and partition function. For example, see `http://sli.ics.uci.edu/misc/cs179/pyGM_Inference.html` for examples of several methods of computing marginal probabilities. For the log-likelihood, you may find it helpful to use `model.logValue( X[i] )` to evaluate $\log F(x^{(i)})$; you will also need to evaluate the log partition function $\log Z$.