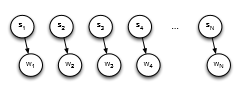
**Part 2 - Mountain Finding**

1. **Using Bayes Net in Figure 1b) –**

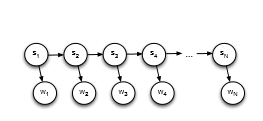


**Problem Formulation –** A bayes net as shown above comprises unobserved row scalars – s1, s2, s3 and so on, such that no two unobserved scalars depend on each other. Instead every scalar depends on the gradient distribution on the corresponding n’th column of the gradient image.

**Description -** Inference from the bayes net as shown in figure 1b), can be drawn for detecting a mountain in the image by making a probability query on the bayes net. For this the algorithm works by taking a max of the edge\_intensity for every column and making a ridge from the corresponding rows.

**Assumption –** Max edge intensity in a column will correspond to a point on the mountain.

1. **MCMC on a Bayes Net as in figure 1a) –**



**Problem Formulation –** A bayes net as shown above adds another dependency as compared to the one in sub-part1 – now row scalars depend on the scalar preceding them as well as the corresponding column gradient.

**Description -** Gibbs Sampling or MCMC was used on a bayes net of observed - edge-intensities and unobserved - row values for every column. This was implemented as the function construct\_ridge2() and runs by taking a sampled particle and then updates the unobserved sampled particle attributes by calculating a probability distribution based on bayes’ law. The sampled particle was chosen at random.

The function probability\_distribution() then calculates the probability distribution of the unobserved attributes of the sample by first narrowing down the search to one fourth image height above and below the intended row and then calculates emission and transition probabilities from those rows into the intended unobserved attribute row of the sampled particle. This is done for every unobserved attribute of the sampled particle previously generated and updated in a dictionary data-structure which keeps track of the most recent sample at time t.

Based on an interval of time during which this function is called, the sampled particle in the dictionary data-structure smoothens given the observed attributes (which is basically the column values and edge-intensities).

The algorithm performs pretty well for a smaller interval of time. It can be perfected by doing the following –

1. Narrowing down the row range in which to look for the transition probabilities for a transition into the unobserved row.
2. Running the sample for a longer amount of time.
3. Or taking human feedback as done in sub-part3.

**Assumptions/Simplifications –** Restriction of probability distribution for unknown row scalars to transition to it from one fourth image height above and below it. This was done so as to restrict the calculation to a best approximate range of rows rather than the entire range of rows in the edge\_strength map. This helps in better performance.

**3) Taking Human Feedback**

**Problem Formulation –** Same as sub-part 2 above, except now the user input would tell us about an exact point on the mountain ridge.

**Description –** For the human feedback, we add two more parameters to the be considered for the probability distribution.

1. How close the column is to the given colum
2. How close the row is to the given

We take a product of these two factors to give more precise input. We take impact of this on only the 50 columns before and after the given column. Rest all columns are indirectly affected by the given human feedback.

This gives us a better position to start looking for the mountain edge as we know for sure that the input point lies on the edge.