CS 6780: Advanced Machine Learning
Homework 4

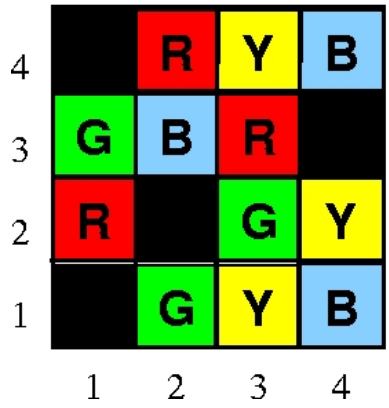
Due date: 12pm. **Dec 3, 2009** (Printed submission.)

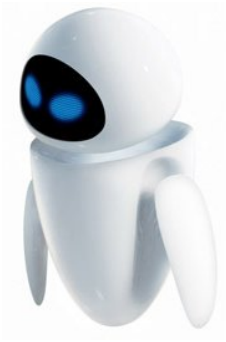
Note:
1. Write your name and net-id on the first page.
2. For derivation questions, please show all the necessary steps.
3. For data analysis / programming parts, you need to submit: (a) the specific values and plots requested, (b) explanation in the text if needed (don't give us the code printouts only!), and (c) the code in the appendix.

1 Hidden Markov Model

A lonely robot (Eve) is wandering through the following small world and is lost. Help her out by building a HMM algorithm that tells her where she is.

The robot can only occupy the colored squares. At each time step, the robot attempts to move `up, down, left or right`, where the choice of





direction is made at random. If the robot attempts to move onto a black square, or to leave the confines of its world, its action has no effect and it does not move at all. (I.e., multinomoial can model state transitions well.) The robot can only sense the color of the square it occupies. However, its sensors are only partly accurate, meaning that part of the time, it perceives a random color rather than the true color of the currently occupied square. More formally, p(sensed|real) is well modeled with a multinomial distribution.

In this problem, state refers to the location of the robot in the world in `x:y` coordinates, and output refers to a perceived color `(r, g, b or y)`. Thus, a typical random walk looks like this:

```
 3:3 r
3:3 r
3:4 y
2:4 b
3:4 y
3:3 r
2:3 b
1:3 g
2:3 b
2:4 r
3:4 y
4:4 y
```

Here, the robot begins in square 3:3 perceiving red, attempts to make an illegal move (to the right), so stays in 3:3, still perceiving red. On the next step, the robot moves up to 3:4 perceiving yellow, then left to 2:4 perceiving blue (*erroneously*), and so on.

By running your program on this problem, you will build an HMM model of this world. Then, given only sensor information (i.e., a sequence of colors), your program will re-construct an estimate of the actual path taken by the robot through its world.

The data for this problem is in `robot_no_momentum.data`, a file containing 200 training sequences (random walks) and 200 test sequences, each sequence consisting of 200 steps. (The first 200 are training and last 200 are test.)

Along with the code, and explanation in plain English, also submit:
(a) Clearly define the measurement and transition probabilities. Write/print the parameters estimated.
(b) Derive / describe how you estimated the parameters of the model.
(c) Describe (or cite the result discussed in class) the inference method.

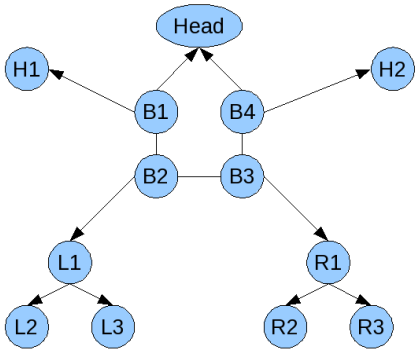
(d) Report the total number of times HMM prediction did not agree with the perceived color from the sensor in the *test* set. (In test set, only the sensor reading is available to the robot.)
(e) If the state is $s_1 : s_2$, define $\psi = 4s_1 + s_2$. Plot ψ for the last 3 test sequences.

You are not allowed to use a pre-existing HMM software. (I.e., you are supposed to write your own.)

(50 points)

Bonus (maximum 10 points): In some scenarios, the state are not even observed in the training set. I.e., in both training and test set, you are only given the sequence of colors. Training in this case would require some EM type method. Test your new training method on `robot_data2.data`.[1]

2 Junction Tree



Eve is now looking for WallE using her cameras, but could not find WallE. Help her out by helping in building a WallE classifier!

Eve has small circuits for performing sum-product, etc. We know that even any graph can be converted into a "tree" — junction tree. So, your goal

[1]Warning: This bonus question might take a lot of time, which might better be spent on your project. Its also okay to use an existing HMM software for the bonus question only.

is to design a **junction-tree** of the graph Eve has in her mind for WallE.[2] (15 points)

3 Kalman Filter

(a) One of the most computation intensive part in Kalman filter is inversion of matrices. For a state of size 10 and measurement of size 5, what is maximum size of the matrix one has to invert.

(b) Consider the version of the Kalman filter described in the class (where everything is Gaussian and there is no control input). Now, lets change the prior distribution $p(x_0)$, i.e., initial state from Gaussian to a mixture of K Gaussians. Does the inference in Kalman filter remains tractable[3] ? Explain concisely.

(c) Now, we model the state update error ϵ, $p(\epsilon)$ using a mixture of L Gaussians. Does it remain tractable? Explain concisely. (18 points)

4 Markov Random Field

In some applications such as stereo vision and image denoising, we deal with continuous variables. Gaussian densities are first ones that one could try.

Given a grid-MRF (e.g., each node is connected to the top, left, right and bottom node), each node $y_i \in \Re$ represents what we want to predict, and $x_i \in \Re^K$ are the input features for that pixel.

We model $p(y_i|x_i;\theta) \propto \exp(-(y_i - x_i^T\theta)^2/2\sigma_1^2)$ using linear regression, and we model $p(y_i, y_j) \propto \exp(-(y_i - y_j)^2/2\sigma_2^2)$ using a Gaussian.

During inference (testing) phase, we are given $\theta, \sigma_1^2, \sigma_2^2$ and x_i's, and we have to calculate $y^* = \arg\max_y p(y|x;\theta,\sigma_1,\sigma_2)$. Write the above maximization as a convex program (specifically a quadratic program).

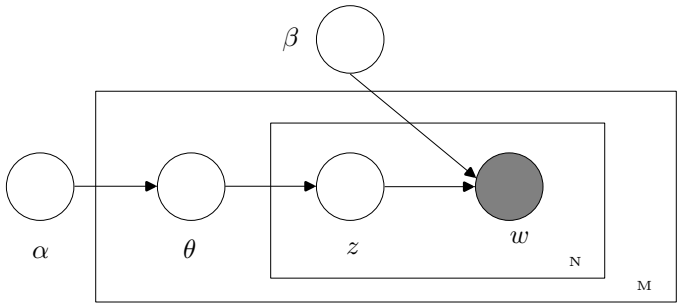
(17 points)

[2]Loopy methods are not being used here, because if they don't converge Eve might not be able to find WallE.

[3]Here, tractable means that the exact inference does not grows (exponentially) with time.

5 Bonus/Optional: Latent Dirichlet allocation

Latent Dirichlet Allocation, or LDA, (Blei et. al. 2003) is a model for discovering topics in a collection of documents. The graphical model we will work with is as follows:



Here's how the data are generated according to the model:

- For each document, m = 1,...,M
  - Draw topic probabilities $\theta_m \sim p(\theta|\alpha)$
  - For each of the N words:
    - Draw a topic $z_{mn} \sim p(z|\theta_m)$
    - Draw a word $w_{mn} \sim p(w|z_{mn}, \beta)$

where $p(\theta|\alpha)$ is a Dirichlet distribution, and where $p(z|\theta_m)$ and $p(w|z_{mn}, \beta)$ are multinomial distributions. Treat α and β as fixed hyperparameters. Note that β is a matrix, with one column per topic, and the multinomial variable z_{mn} selects one of the columns of β to yield multinomial probabilities for w_{mn}. (See the paper "Latent Dirichlet Allocation" by Blei et. al. on the course website for more details if needed). Write down a Gibbs sampler for the LDA model. That is, write down the conditional probabilities of z and θ given their Markov blankets so that we can sample from these distributions. (15 points)