Probabilistic Graphical Models

HWK#2 Part A
Assigned Tuesday, Feb. 13, 2024
Due for both Part A and B : Tuesday, March 5 (11:59PM EST), 2024

Problem 1 (Continuation of Problem 12 from HWK1) (20 points)

As the owner of an online bookstore, you would like to implement a recommendation system for your customers. After pouring over your records, you discover that you carry only four books. What's worse, you have only three customers. Even worse than that, you've only sold six books in the last year.

In contrast to Naïve Bayesian Model you used in Problem 12 in HWK1, after thinking about the problem, you come up with an elaborate model in Figure 4 in below:

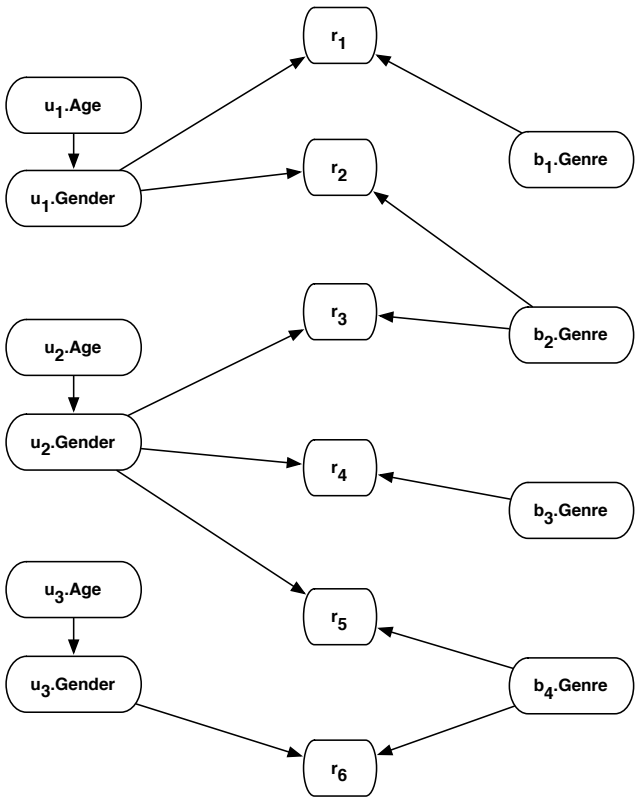


Figure 4: A elaborate model of recommendation. Customers are labelled u_i, books are labelled b_j, ratings are labelled r_k. There is a variable for each user's age and gender, each book's genre, and each rating. If the parents are r_k are u_i and b_j this means that user i bought book j and the rating he assigns to it is r_k.

Part (a) (2 points)

Consider the elaborate model in figure 4. If you already know r_2 and r_3, then what variables are influenced by revealing the value of r_1 ?

Answer:

The observed nodes are all the children in a v-structure. Hence, their respective parents are connected by an active trail and hence the following variables all influence each other: `u1.Age, u1.Gender, b1.Genre, u2.Age, u2.Gender, b2.Genre`.

Part (b) (6 points)

Hint part (b) and (c) of this problem: use your solution from Problem 12 in HWK1, where you implemented the following steps (they are in blue ink)

1. A data structure to represent a factor, a mapping from an assignment of variables to a real value. Conditional probability tables can be viewed as factors. The easiest way to encode a factor is as a multidimensional array where each dimension corresponds to a variable. See "table_factor.m" .
2. A data structure to represent a Bayesian network. The easiest way to do this is just to store a list of all the conditional probability tables as factors.
3. A data structure that represent an assignment to variables. The easiest way to do this is as a pair of vectors: vars and vals. Note that vals(i) is the value assigned to variable vars(i). See "assignment.m".
4. A function that takes a Bayesian network and an assignment to all the variables, and returns the probability of that assignment.
5. A function that iterates over all assignments to a set of variables, and accumulates the value of the joint distribution at these assignments. This is marginalization.
   Important: You will not get full marks for an implementation that stores the full joint distribution explicitly.

You are given a parameterization for the elaborate model in figure 6 as below

$$\alpha = 0.3, \beta = 0.6, \gamma = 0.46, \delta = 0.25, \epsilon = 0.35, \zeta = 0.21, \eta = 0.25, \theta = 0.15, \iota = 0.06, \kappa = 0.18, \lambda = 0.04, \mu = 0.11.$$

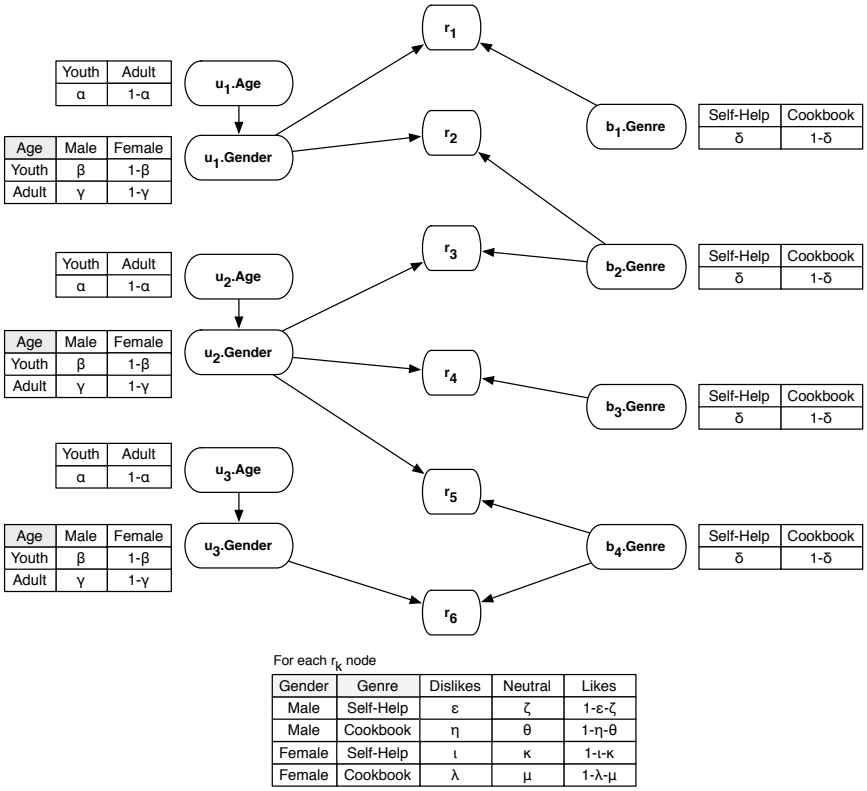


Figure 6: Parameters for the elaborate model. Note that nodes can share the same conditional probability table, e.g., all the Age nodes share the same parameters.

Let denote all the parameters of this model as

$$\Theta_{EL} = (\alpha, \beta \ldots \lambda, \mu)$$

Using your earlier implementation of inference (direct naïve marginalization without using variable elimination algorithm), what are the values of the following queries?

1. $P(u_1.\text{Age} = \text{Youth} | r_1 = \text{Likes}, r_2 = \text{Likes}, r_3 = \text{Dislikes}, r_4 = \text{Likes}, b_4.\text{Genre} = \text{Cookbook})$
2. $P(u_2.\text{Gender} = \text{Male}, u_2.\text{Age} = \text{Youth} | r_1 = \text{Likes}, r_6 = \text{Dislikes}, \ldots$
   $b_2.\text{Genre} = \text{Cookbook}, b_3.\text{Genre} = \text{Self-Help}, b_4.\text{Genre} = \text{Cookbook})$
3. $P(r_3 = \text{Likes} | r_1 = \text{Dislikes}, r_2 = \text{Likes}, r_4 = \text{Neutral}, r_5 = \text{Neutral}, r_6 = \text{Likes})$

Store the values of the above queries in your writeup, and submit your code as "infer.m"

Part (c) (3 points)

Here you will be doing parameter estimation:

You are provided sales data from a large book retailer. A list of records of the form

Age	Gender	Genre	Rating
Youth	Male	Cookbook	Likes
Adult	Female	Self-Help	Dislikes
...	...	...	...

for a large number of users data is located in "purchase.csv" file. Look at the provided function "loaddata.m" for how to load this data into MATLAB.

Using this data, implement a function that computes the maximum likelihood estimate for $\Theta_{EL,}$ all the parameters of this model. Record the estimates of the parameters in your writeup and submit your code as "ml_estimate_EL.m"

Hint: This is similar to Problem 12 in HWK1. Note that in the elaborate model, there is a variable for each attribute of every entity (user, book, rating). However, the data in purchase.csv only contains Age, Gender, Genre, and Rating attributes. The way we solved this problem is to tie parameters together – all the Age nodes share the same CPT; all the Gender nodes share the same CPT; all the Genre nodes share the same CPT; and all the Rating nodes share the same CPT. The problem has been reduced to estimating conditional probability tables over the four variables. For example,

- α is the fraction of all records that have Age = Youth.
- β is the fraction of records with Age = Youth that also have Gender = Male.
- ε is the fraction of records with Gender = Male and Genre = Self-Help that also have Rating = Dislikes.
- ζ is the fraction of records with Gender = Male and Genre = Self-Help that also have Rating = Neutral.

Part (d) (4 points)

Variable Elimination For query 2 in part (b), show the moralized graph. Choose a good elimination ordering and write down the variable elimination procedure. Show the intermediate factors produced after eliminating each variable. What is the time and memory complexity of the variable elimination algorithm?

> Note:- Do not substitute numbers for the marginal and conditional probabilities yet. You just have to describe the procedure for a good elimination ordering similar to lecture.

Part (e) (5 points)

Implement Variable Elimination or Message Passing Now: you can directly implement the variable elimination procedure or you could implement this using the message passing protocol. Your implementation should be general, i.e., your implementation should be able to handle any valid query. You can validate your implementation using the three queries described in part (b). Report the final answers for all the three queries and run times. Compare the result and runtime to the case when you simply used the naïve marginalization in part (b).

Report: You should provide a short (3 pages) summary describing your explorations and results. Submit a hard copy of the report with the assignment. Please also upload the complete source code of your implementation to canvas. Remember to include a small Readme file and a script that would help us to execute your code. You are not allowed to use any package for computing messages or performing variable elimination. Only basic computations are allowed. Upload all your codes, data, and report in a single folder for Problem 1.

Solution:

Part (b)

The code for representing the Bayesian network can be found in the class `BNet` inside the `bayes_net.py` file. The class contains the following member functions:

- `factor(variables, values)` - returns the value of the factor which takes the `variables` as arguments, for the variable assignments specified in `values`. This is similar to `table_factor.m`.
- `joint_prob(values)` - returns the joint probabilities for a specific value assignment of the variables in the Bayesian network. The values of the nodes are specified by the variable `values`. (Step 4 of the hints)
- `marginalize(variables, non_marginal_assignments)` - returns the marginal probability for the assignment of variables defined in `non_marginal_assignments`, where the marginalization takes place with respect to the variables in the list `variables`. (Step 5 of the hints)
- `variable_elimination` - returns the results of performing variable elimination over the nodes in the BN for a given elimination ordering.
- `learn_params(data)` - learns the factor table entries using MLE based on the dataset provided in `data`.

From the definition of conditional probability, we have $P(A \mid B) = \frac{P(A,B)}{P(B)}$, using which part 1 can be written as follows,

$$P(\mathtt{u_1.Age = Youth} \mid \mathtt{r_1 = Likes}, \mathtt{r_2 = Likes}, \mathtt{r_3 = Dislikes}, \mathtt{r_4 = Likes}, \mathtt{b_4.Genre = Cookbook}) = \frac{P(\mathtt{u_1.Age = Youth}, \mathtt{r_1 = Likes}, \mathtt{r_2 = Likes}, \mathtt{r_3 = Dislikes}, \mathtt{r_4 = Likes}, \mathtt{b_4.Genre = Cookbook})}{P(\mathtt{r_1 = Likes}, \mathtt{r_2 = Likes}, \mathtt{r_3 = Dislikes}, \mathtt{r_4 = Likes}, \mathtt{b_4.Genre = Cookbook})}.$$

The RHS in the above equa tion can be computed using two calls to `marginalize()`. In the first call, we marginalize the variable `Gender` and in the second call, we marginalize the variables `Rating` and `Gender`. Parts 2 and 3 can solved similarly. The results are summarized below:

1. $P(\mathtt{u_1.Age = Youth} \mid \mathtt{r_1 = Likes}, \mathtt{r_2 = Likes}, \mathtt{r_3 = Dislikes}, \mathtt{r_4 = Likes}, \mathtt{b_4.Genre = Cookbook}) \approx 0.277$.
2. $P(\mathtt{u_2.Gender = Male}, \mathtt{u_2.Age = Youth} \mid \mathtt{r_1 = Likes}, \mathtt{r_6 = Dislikes}, \mathtt{b_2.Genre = Cookbook}, \mathtt{b_3.Genre = Self\text{-}Help}, \mathtt{b_4.Genre = Cookbook}) \approx 0.179$.
3. $P(\mathtt{r_3 = Likes} \mid \mathtt{r_1 = Dislikes}, \mathtt{r_2 = Likes}, \mathtt{r_4 = Neutral}, \mathtt{r_5 = Neutral}, \mathtt{r_6 = Likes}) \approx 0.665$.

Part (c)

The method `learn_params(data)` inside `bayes_net.py`, performs MLE of the network parameters using the provided data. This part is similar to part (c) of problem 12 in HWK 1 with the main difference being that we tie all the user age and genders together. Similarly, we also tie all the book genre's and the ratings together. We start by initializing all the factor tables to zero. For each sample (row) in the data file, one is added to the corresponding entries in each factor tables associated with the Bayesian network. The learnt parameters are reported below:

Parameter	Value
α	0.677
β	0.811
γ	0.628
δ	0.603
ϵ	0.073
ζ	0.869
η	0.218
θ	0.651
ι	0.301
κ	0.604
λ	0.610
μ	0.261

Table: MLE parameters

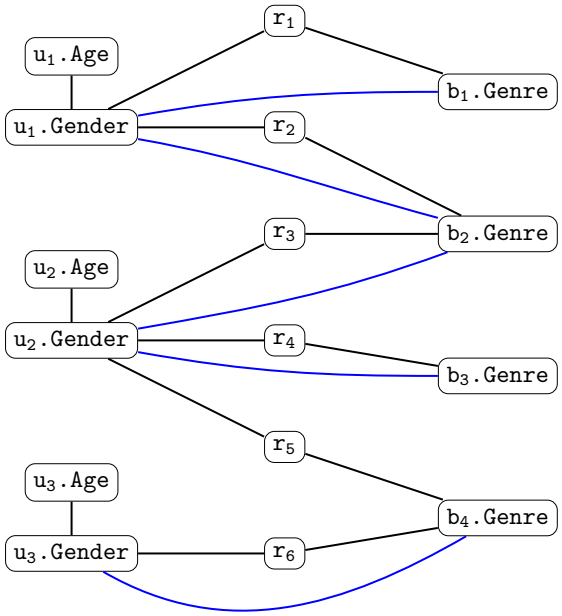


Figure: Moralized graph

Part (d)

The moralized graph of the Bayesian network is shown in the figure above, where the blue edges correspond to the moralizing edges. We now choose the following ordering for eliminating the variables (left to right, top to bottom):

$$\mathtt{r_6},\ \mathtt{r_5},\ \mathtt{r_4},\ \mathtt{r_3},\ \mathtt{r_2},\ \mathtt{r_1},\ \mathtt{u_3.Gender},\ \mathtt{u_2.Gender},\ \mathtt{u_1.Gender},\ \mathtt{b_4.Genre},\ \mathtt{b_3.Genre},\ \mathtt{b_2.Genre},\ \mathtt{b_1.Genre},\ \mathtt{u_3.Age},\ \mathtt{u_2.Age},\ \mathtt{u_1.Age}$$

From the definition of conditional probability, query 2 in part (b) becomes

$$P(\mathtt{u_2.Gender = Male}, \mathtt{u_2.Age = Youth} \mid \mathtt{r_1 = Likes}, \mathtt{r_6 = Dislikes}, \mathtt{b_2.Genre = Cookbook}, \mathtt{b_3.Genre = Self\text{-}Help}, \mathtt{b_4.Genre = Cookbook}) = P(\mathtt{u_2.Gender = Male}, \mathtt{u_2.Age = Youthr_1 = Likes}, \mathtt{r_6 = Dislikes}, \mathtt{b_2.Genre = Cookbook}, \mathtt{b_3.Genre = Self\text{-}Help}, \mathtt{b_4.Genre = Cookbook})/P(\mathtt{r_1 = Likes}, \mathtt{r_6 = Dislikes}, \mathtt{b_2.Genre = Cookbook}, \mathtt{b_3.Genre = Self\text{-}Help}, \mathtt{b_4.Genre = Cookbook}).$$

Focusing on the numerator on the right-hand-side of the equation above, the variable elimination procedure is summarized here:

1. Eliminating $\mathtt{r_6}$ when it is observed results in the factor $m_{\mathtt{r_6}}(\mathtt{u_3.Gender}, \mathtt{b_4.Genre}) = P(\mathtt{r_6 = Disikes} \mid \mathtt{u_3.Gender}, \mathtt{b_4.Genre})$.
2. Eliminating variables $\mathtt{r_5}$, $\mathtt{r_4}$, $\mathtt{r_3}$, $\mathtt{r_2}$ results in a factor with constant value 1.
3. Eliminating $\mathtt{r_1}$ when it is observed results in the factor $m_{\mathtt{r_1}}(\mathtt{u_1.Gender}, \mathtt{b_1.Genre}) = P(\mathtt{r_1 = Likes} \mid \mathtt{u_1.Gender}, \mathtt{b_1.Genre})$.
4. Eliminating $\mathtt{u_3.Gender}$ results in the following factor

$$m_{\mathtt{u_3.Gender}}(\mathtt{u_3.Age}, \mathtt{b_4.Genre}) = \sum_{\mathtt{u_3.Gender}} P(\mathtt{u_3.Gender} \mid \mathtt{u_3.Age}) m_{\mathtt{r_6}}(\mathtt{u_3.Gender}, \mathtt{b_4.Genre}).$$

5. Eliminating $\texttt{u}_2\texttt{.Gender}$ results in the factor $m_{\texttt{u}_2\texttt{.Gender}}(\texttt{u}_2\texttt{.Age}) = P(\texttt{u}_2\texttt{.Gender = Male} \mid \texttt{u}_2\texttt{.Age})$.

6. Eliminating $\texttt{u}_1\texttt{.Gender}$ results in the following factor

$$m_{\texttt{u}_1\texttt{.Gender}}(\texttt{u}_1\texttt{.Age}, \texttt{b}_1\texttt{.Genre}) = \sum_{\texttt{u}_1\texttt{.Gender}} P(\texttt{u}_1\texttt{.Gender} \mid \texttt{u}_1\texttt{.Age}) m_{\texttt{r}_1}(\texttt{u}_1\texttt{.Gender}, \texttt{b}_1\texttt{.Genre}).$$

7. Eliminating $\texttt{b}_4\texttt{.Genre}$ results in the factor

$$m_{\texttt{b}_4\texttt{.Genre}}(\texttt{u}_3\texttt{.Age}) = P(\texttt{b}_4\texttt{.Genre = Cookbook}) m_{\texttt{u}_3\texttt{.Gender}}(\texttt{u}_3\texttt{.Age}, \texttt{b}_4\texttt{.Genre = Cookbook}).$$

8. Eliminating $\texttt{b}_3\texttt{.Genre}$ results in the factor $m_{\texttt{b}_3\texttt{.Genre}} = P(\texttt{b}_3\texttt{.Genre = Self-Help})$.

9. Eliminating $\texttt{b}_2\texttt{.Genre}$ results in the factor $m_{\texttt{b}_2\texttt{.Genre}} = P(\texttt{b}_2\texttt{.Genre = Cookbook})$.

10. Finally, eliminating $\texttt{b}_1\texttt{.Genre}$ results in the factor

$$m_{\texttt{b}_1\texttt{.Genre}}(\texttt{u}_1\texttt{.Age}) = \sum_{\texttt{b}_1\texttt{.Genre}} P(\texttt{b}_1\texttt{.Genre}) m_{\texttt{u}_1\texttt{.Gender}}(\texttt{u}_1\texttt{.Age}, \texttt{b}_1\texttt{.Genre}).$$

Each step of the elimination process takes at most $O(n^3)$ computations, where n is the largest sample space of the variables in the graph. Therefore in total, the time complexity of variable elimination becomes $O(n^3 k)$, where k is the total number of variables in the graph. Memory complexity on the other hand is $O(n^2)$, where the factors resulting from eliminating the variables are stored in place.

Part (e)

The codes for variable elimination can be found within `bayes_net.py` code file. The implementation utilizes factor product and marginalization in order to perform variable elimination. The code for testing variable elimination can be found in `ve.py`. Here we summarize the results for the queries in part (b).

Query	VE (ans)	VE (time - sec)	Naive (ans)	Naive (time - sec)
1	0.277	9.9×10^{-4}	0.277	0.17
2	0.18	9.01×10^{-4}	0.179	0.28
3	0.665	8.3×10^{-4}	0.665	0.09

Problem 2 An HMM for the Stock Market (25 points)

The Hidden Markov Model (HMM) has been a workhorse in the financial industry for decades, and is still a popular model today. In this question, we will attempt to use an HMM to infer hidden market states based on observations of day-to-day changes in the S&P 500, which is a major stock index computed as a weighted average of the stock prices of 500 leading companies in the US.

HMMs are state space models composed of a sequence of latent variables (states) that each generate an observed variable (emissions). The latent variables form a Markov chain, which means that the state at each time point, Z_t, is conditionally independent of all prior states given the state at the previous time point, Z_{t-1}. The HMM graphical model is shown in Figure 2.

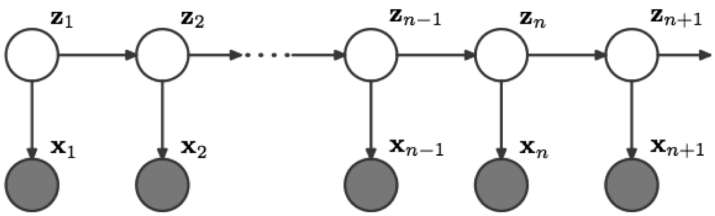


Figure 2: The graphical model for an HMM. Each emission X_t is conditioned on the corresponding latent state Z_t. The latent variables form a Markov chain.

The joint probability distribution is:

$$P(\boldsymbol{X}, \boldsymbol{Z}; \theta) = P(Z_1; \pi) \left[\prod_{t=2}^{T} p(Z_t | Z_{t-1}; A) \right] \left[\prod_{t=1}^{N} P(X_t | Z_t; E) \right]$$

where $\boldsymbol{X} = \{X_1, ..., X_T\}$ is the sequence of emissions, $\boldsymbol{Z} = \{Z_1, ..., Z_T\}$ is the sequence of states, and $\theta = \{\pi, A, E\}$ is the set of model parameters. Here π is the categorical parameter for $P(Z_1)$, A is the state transition matrix, and E is the matrix of emission probabilities. Whenever appropriate, we will omit π, A, E for brevity since we assume they are fixed.

Part 1: Derive the Forward-Backward Algorithm (8 points)

In this part, we are going to derive the forward-backward algorithm for an HMM using a factor graph. One way to do this is to put a factor on each edge of the graphical model, in which case we'd have factors of the form $\phi_t(Z_t, Z_{t-1})$ and $\psi_t(Z_t, X_t)$. But since $\boldsymbol{X}$ is observed, we can absorb the ψ factors into ϕ and write

$$\phi_t(Z_t, Z_{t-1}) := \phi_t(Z_t, Z_{t-1}, X_t) = P(Z_t | Z_{t-1}) P(X_t | Z_t)$$

without increasing the complexity (i.e. the tree-width) of the factor graph. We also define

$$\phi_1(Z_1) := \phi_1(Z_1, X_1) = P(Z_1) P(X_1 | Z_1)$$

1. Draw the factor graph corresponding to these definitions of the factors.

2. Based on the standard message passing algorithm on a factor graph, we define the following two messages:

$$m_{Z_{t-1} \to \phi_t}(Z_{t-1}) = m_{\phi_{t-1} \to Z_{t-1}}(Z_{t-1})$$
$$m_{\phi_t \to Z_t}(Z_t) = \sum_{Z_{t-1}} \phi_t(Z_{t-1}, Z_t)\, m_{Z_{t-1} \to \phi_t}(Z_{t-1})$$

   It can be shown that $\alpha(Z_t) := P(X_1, ..., X_t, Z_t) = m_{\phi_t \to Z_t}(Z_t)$ (to verify this, you can try it yourself). Now consider the reverse direction. Write down messages $m_{Z_{t-1} \to \phi_{t-1}}(Z_{t-1})$ and $m_{\phi_t \to Z_{t-1}}(Z_{t-1})$, and then show that

$$m_{\phi_t \to \phi_{t-1}}(Z_{t-1}) = \sum_{Z_t} \phi_t(Z_t, Z_{t-1})\, m_{\phi_{t+1} \to \phi_t}(Z_t) \tag{1}$$

3. Given $\beta(Z_t) := m_{\phi_{t+1} \to \phi_t}(Z_t)$, we have the following recursive property

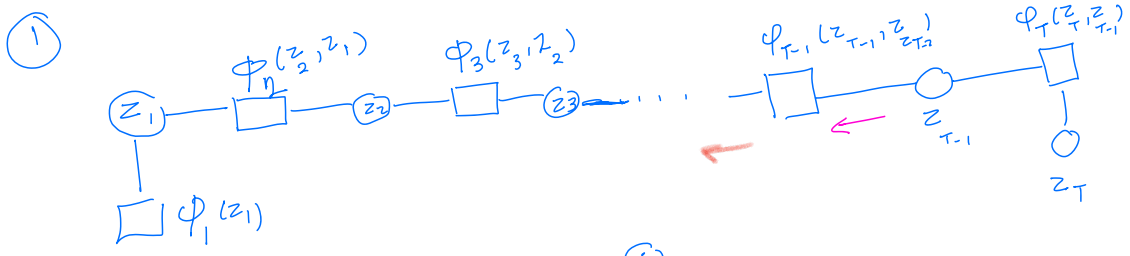
$$\beta(Z_t) = \sum_{Z_{t+1}} \beta(Z_{t+1}) P(X_{t+1} | Z_{t+1}) P(Z_{t+1} | Z_t)$$

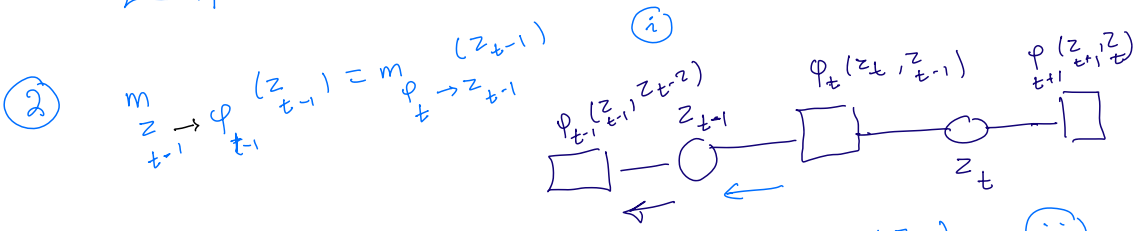
   Show that this property holds when we define $\beta(Z_t) = P(X_{t+1}, ..., X_T | Z_t)$.

output symbol	description	% daily change
1	large drop	$< -2\%$
2	small drop	-2% to -1%
3	no change	-1% to 1%
4	small rise	1% to 2%
5	large rise	$> 2\%$

Table 1: The output symbols for the HMM model. The change in the S&P 500 index is measured as (closing index today − closing index yesterday) / (closing index yesterday)

Solution:





where: $m_{\psi_t\to z_{t-1}}(z_{t-1}) = \sum_{z_t} \psi_t(z_t,z_{t-1}) \cdot \underset{z_t\to\psi_t}{\varphi}(z_t)$ (iii)

But using eqn (ii), we can also write $\underset{z_t\to\psi_t}{\varphi}(z_t) = m_{\psi_{t+1}\to z_t}(z_t)$

Thus (ii) becomes: $m_{\psi_t\to z_{t-1}}(z_{t-1}) = \sum_{z_t} \psi_t(z_t,z_{t-1})\, m_{\psi_{t+1}\to z_t}(z_t)$ (iii)

3 Let $\beta(z_t) \triangleq m_{\psi_{t+1}\to z_t}(z_t)$ (iv)

Then, using eqn (iii) in part (2), replace t with $t+1$: $\beta(z_t) = \sum_{z_{t+1}} \psi_{t+1}(z_{t+1},z_t)\,\beta(z_{t+1})$

since $\psi_{t+1}(z_{t+1},z_t) = p(z_{t+1}|z_t)\, p(x_{t+1}|z_{t+1})$

Then, $\beta(z_t) = \sum_{z_{t+1}} \beta(z_{t+1}) \cdot p(x_{t+1} | z_{t+1}) \cdot p(z_{t+1} | z_t)$

we verify that

we have $\beta(z_t) = m_{\varphi_{t+1} \to z_t}(z_t)$

we also have: $p(x_{t+1}, \dots, x_T | z_t) = \frac{\sum_{z_{t+1}, \dots z_T} p(x_{t+1}, \dots, x_T, z_t, \dots, z_T)}{p(z_t)}$

$= \sum_{z_{t+1}, \dots z_T} \frac{p(x_T | z_T)}{p(x_T | x_{t+1}, \dots, x_{T-1}, z_t, \dots, z_T)} \cdot \frac{p(x_{t+1}, \dots, x_{T-1}, z_t, \dots, z_T)}{p(z_t)}$ (eq 1)

↗ Chain Rule

similarly: $p(x_{t+1}, \dots, x_{T-1}, z_t, \dots, z_T) = p(x_{T-1} | z_{T-1}) \cdot p(x_{t+1}, \dots, x_{T-2}, z_t, \dots, z_T)$

$= p(x_{T-1} | z_{T-1}) \cdot p(x_{T-2} | z_{T-2}) \cdots p(x_{t+1} | z_{t+1}) \cdot p(z_t, \dots, z_T)$

$= \cdot\, p(z_T | z_{T-1}) \cdot p(z_{T-1} | z_{T-2}) \cdots p(z_{t+1} | z_t) \cdot p(z_t)$ (eq 2)

Replacing (eq 2) in (eq 1), we get ($p(z_t)$ get cancelled from numerator and denominator)

$p(x_{t+1}, \dots, x_t | z_t) = \sum_{z_{t+1}} \cdots \sum_{z_T} p(z_{t+1} | z_t) \cdots p(z_T | z_{T-1}) \cdot p(x_{t+1} | z_{t+1}) \cdots p(x_T | z_T)$

$= \sum_{z_{t+1}} p(x_{t+1} | z_{t+1}) p(z_{t+1} | z_t) \cdots \sum_{z_{T-1}} p(z_{T-1} | z_{T-2}) p(x_{T-1} | z_{T-1}) \cdot \sum_{z_T} p(z_T | z_{T-1}) p(x_T | z_T)$

($\sum_{z_T} p(z_T | z_{T-1}) p(x_T | z_T) = \beta(z_T)$ = one; $\sum_{z_{T-1}} p(z_{T-1} | z_{T-2}) p(x_{T-1} | z_{T-1}) \beta(z_T) = \beta(z_{T-1})$; $\sum_{z_{T-2}} p(z_{T-2} | z_{T-3}) p(x_{T-2} | z_{T-2}) \beta(z_{T-1}) = \beta(z_{T-2})$; $\sum_{z_{t+1}} p(x_{t+1} | z_{t+1}) p(z_{t+1} | z_t) \beta(z_{t+1}) = \beta(z_t)$)

$= \sum_{z_{t+1}} p(x_{t+1} | z_{t+1}) p(z_{t+1} | z_t) \beta(z_{t+1})$

$= \beta(z_t)$

Part 2: Perform HMM Inference

For this problem, we will use a simple HMM with $Q = 3$ hidden states and a sequence length of $T = 100$ trading days. The choice of a small Q reduces the model complexity, which helps in performing inference over the hidden states. You can roughly think of these hidden states as "bull market", "bear market", and "stable market". The emission model is a multinomial distribution with $O = 5$ observed symbols. See Table 1 for details on the output states. The parameters of this HMM have already been estimated for you using the EM algorithm on a much longer portion of data (from 01/01/2010 to 07/31/2013). You will use this fully parameterized model to carry out inference over a sequence of 100 trading days (starting on 08/01/2013), and then you will perform forward prediction of the output values over the next 28 days. All of the data used in this problem was downloaded from Yahoo! finanace.

In "hmm_params.mat" you will find the following parameters:

- *transition*: the transition probability matrix, where $transition(i, j) = P(Z_{t+1} = j | Z_t = i)$
- *prior*: the prior distribution over Z_1, where $prior(i) = P(Z_1 = i)$
- *emission*: the emission probability matrix, where $emission(i, j) = P(X_t = j | Z_t = i)$
- *price_change*: the observations, where $price\_change(i) = \frac{price(i) - price(i-1)}{price(i-1)}$

Here are the details of your tasks:

1. [5 pts] Implement the Forward-Backward algorithm and run it using the observations for time points $t = 1, ..., 100$. Report the inferred distributions over the hidden states at $t = 1, ..., 100$ by plotting the probabilities $P(Z_t = i | X_1, ..., X_{100})$ for $i = 1, 2, 3$ over $t = 1, ..., 100$. Make sure you label the 3 time series (one for each hidden state) in your plot.
2. [5 pts] Implement the Viterbi algorithm and find the most likely sequence of hidden states $Z_1, ..., Z_{100}$ for the same time period. Report the most likely hidden states over $t = 1, ..., 100$ by plotting these values as a time series.
3. [5 pts] Now, let's see how well this model performs. Predict the output symbols for time points $t = 101, ..., 128$ by carrying out the following steps for each time point t:
   (a) Run the forward algorithm to estimate $P(Z_{t-1} | X_{t-101}, ..., X_{t-1})$. Note that $X_{t-101}, ..., X_{t-1}$ are the ground truth observations. (Alternatively, you can compute $P(Z_t | X_1, ..., X_{t-1})$, but clarify if you take this approach.)
   (b) Compute $P(Z_t)$ from $P(Z_{t-1})$ using the transition matrix. Generate the output value X_t by sampling a state z_t from $P(Z_t)$ and then from $P(X_t | Z_t = z_t)$.

   Compare your predictions with the ground truth observations at time points $t = 101, ..., 128$. What's the percentage of these values that your model predicts correctly? Report the average and variance over 100 runs.

4. [2 pts] One of your TAs is considering investing in the S&P 500. Would you recommend that he/she use this model to guide their investment decisions? Why or why not? Answer this question carefully as your TA's money is on the line!

Note that the Viterbi algorithm in question 2 in above is nothing but solving for MPE problem in HMM. You must not use open source Viterbi to solve this. You need to use max-sum or max-product algorithm as we discuss in the lecture.

Report: You should provide a short summary describing your explorations and results. Submit a hard copy of the report with the assignment. Please also upload the complete source code of your implementation to canvas. Remember to include a small Readme file and a script that would help us to execute your code.

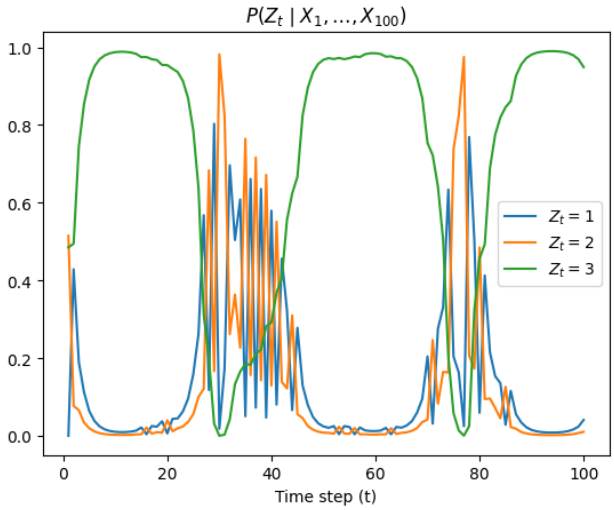
Part 2: Perform HMM Inference

The code for all the subparts of part 2 can be found in `problem2.py`.

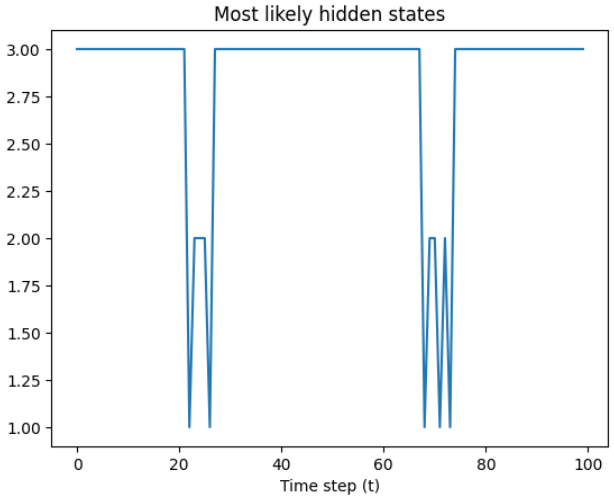
1. Using the forward-backward algorithm we can compute $\alpha(Z_t)$ and $\beta(Z_t)$. From their definitions

$$P(Z_t \mid X_1, \ldots, X_T) \propto \alpha(Z_t)\beta(Z_t),$$

which can then be normalized to obtain the true distribution. For $t - 1, \ldots, 100$ the following plot shows the probability distribution for $Z_t = 1, 2, 3$.



2. Using the Viterbi algorithm, implemented using the max-product message passing algorithm, the most likely hidden state sequence is computed. The max-product message passing algorithm starts from the final node and iteratively sends a message to its neighbhor, until we reach the first node in the graph. Using the final message the most likely hidden node state is computed by taking the state leading to the highest probability. This then allows us to backtrack and find the other hidden states. The hidden states are shown in the plot below.



3. Using the forward-backward algorithm, we can compute $P(Z_{t-1} \mid X_{t-1}, \ldots, X_{t-100})$ for $t = 101, \ldots, 128$. Using the conditional distribution for Z_t we can then solve for $P(Z_{t+1} \mid X_{t-1}, \ldots, X_{t-100})$ using the

transition density function $P(Z_{t+1} \mid Z_t)$. Now, sample $Z_{t+1} \sim P(Z_{t+1} \mid X_{t-1}, \ldots, X_{t-100})$, and $X_t \sim P(X_t \mid Z_t)$. The average accuracy and the variance over 100 separate datasets of 28 samples are shown in the table below.

Av. Mean	0.642
Av. variance	≈ 0

4. The HMM predictor is quite poor with accuracy around 64%, whereas a basic predict which predicts the most likely state from the first 100 samples has an accuracy of 80%. Hence, HMM fails to capture the underlying mechanics of the data.