Bayesian networks, Dynamic Bayesian networks, and Hidden Markov models

Valliammai Meenakshi Sundaram, Jennifer Stewart, Connor Baugh

April 18, 2022

1 Introduction

1.1 Bayesian networks

Bayesian Networks have been applied to data from multiple fields, including genetics and artificial intelligence [Pea11], since they were first introduced as *inference nets* by Pearl in 1963 [Pea22]. The benefits of Bayesian networks include the ability to learn about the structure and interactions within system from data that is noisy, multi-level, hierarchical, part of an unknown system, or part of a system for which their is some prior knowledge, such as previous data or rules. Bayesian networks developed from the application of the probabilistic Bayes' Theorem (Theorem 1) to graph theory. This development was spurred by the need to model the top-down and bottom-up systems of the 1970s [Pea11].

Theorem 1 (Bayes' Theorem)**.** *Bayes' Theorem states*

$$\Pr(A|B) = \frac{\Pr(B|A)\Pr(A)}{\Pr(B)} \tag{1}$$

Where:
Pr(A—B) – the probability of event A occurring, given event B has occurred
Pr(B—A) – the probability of event B occurring, given event A has occurred
Pr(A) – the probability of event A
Pr(B) – the probability of event B

A Bayesian network consists of a directed acyclic graph (DAG) (Theorem 2) whose vertices (nodes) represent variables and arcs (connections) represent interactions among the variables. Within the DAG, the interactions among the variables are represented by arrows, showing the direction of interaction. The Bayesian network may be built using a top-down (prognostic) or bottom-up (diagnostic) view. The DAG is built using Bayes likelihood-ratio to update the conditional probabilities based upon the previous node at the local-level or for the root node, based on a prior probability estimation [Pea22]. A Bayesian network represents the joint probability distribution of every possible event by factoring it into a conditional probability distribution for each variable dependent on its parent node, for which the probability is known [Pea11]. In other words, the overall structure of the network is learned starting at the local-level, from the root node, and then the conditional probabilities are calculated based on Bayes Theorem from there and outward to any potential conditional dependencies between nodes.

Theorem 2 (Directed acyclic graph)**.** *The directed acyclic graph follows the graphical structure*

$$\mathcal{G} = (V, A), \tag{2}$$

Where:
V is the node (or vertex) set
A is the arc (or edge) set

The DAG defines a factorization of the joint (global) probability distribution of $V = X_1, X_2, ..., X_v$ into a set of local probability distributions, one for each variable [Scu09]. A Bayesian network is a directed acyclic graph that contains only directed arcs, does not contain loops within an individual node, and does not contain any cyclical structure within the graph. However, dynamic Bayesian networks allow for loops within individual nodes.

1.2 Dynamic Bayesian networks

Dynamic Bayesian networks (DBN) are an extension of Bayesian networks and are used in time series data modelling and prediction. Bayesian networks represent the conditional dependencies between variables at a particular moment in time. Whereas, dynamic Bayesian networks are Bayesian networks that represents the statistical relationship and conditional dependencies between multiple discrete times. Dynamic Bayesian networks involve slicing the data and learning based on the dependencies between the time slices and within the time slices itself [PSD$^+$20].

Dynamic Bayesian network is used for Bayesian networks are graphical probabilistic model in which nodes represent the variables and directed edges or arrows represent the conditional dependencies between the variables.

In a dynamic Bayesian network, each time slice has the same set of nodes and conditional dependencies between the nodes at that slice (will be intra-dependencies within the time slice). All time slices will have the same variables and will have the same intra-dependencies. Each time slice will represent the state of the system at the particular time. The same set of arcs are placed between nodes from different time slices and that represents the inter-dependencies, with their directions following the direction of time [Lè13].

With temporal data, we can use Bayesian Networks for the following:

1. Smoothing: Predicting the values of missing values in the past.
2. Filtering: Predicting the unobserved values at the current time.
3. Prediction: Predicting the values at the future steps.

1.3 Hidden Markov models

Hidden Markov models (HMMs) are considered a special case of Dynamic Bayesian Networks (DBNs) because of the assumption that there is only one discrete hidden state as opposed to the arbitrarily many state variables that can be modelled using DBNs [299]. Because they only predict a single hidden state for each time step of a Markov process, HMMs are technically the simplest possible form of a DBN. HMMs are a type of graphical model used to predict unobserved discrete "hidden" states given observations of some Markov process. They are a natural extension of Markov chains, a model which can predict states given only the previous state observed, but which are incapable of modelling "hidden" states with unknown probabilities given the same observations. HMMs are able to achieve this by deriving the probabilistic features of a process through the use of probability distributions that explain the relationships between observable variables and unknown hidden states [Ver21].

2 Methods

To understand Bayesian and dynamic Bayesian networks and Hidden Markov models, it is essential to understand Bayes theorem. Bayes theorem can be defined as the probability of an event based on knowledge of prior events. The probability of event A occurring given that event B has already occurred is represented as below.

$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

A and B are two different events and their probabilities are represented as P(A) and P(B) respectively. $P(A|B)$ represents the conditional probability of A occurring provided B occurred.

A Bayesian network is a graphical model that is based on the conditional probabilities among variables of interest along with statistical techniques.see Fig. 1

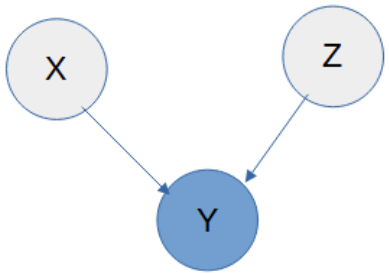


Figure 1: Bayesian Network Representation

Say X and Z are two predictor variables represented as nodes and there dependency on the output Y is represented as directed arcs.[Hec20]

Dynamic Bayesian Networks and Hidden Markov models are based on the Bayesian Network which relate variables to each other over adjacent time steps, which can be represented like Fig. 2:- Where X is the predictor variable and Y is the response variable.[Mur02]

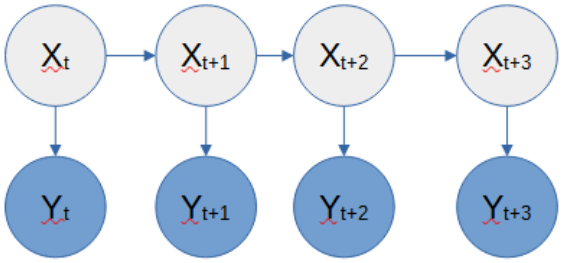


Figure 2: Dynamic Bayesian Network Representation

Let X_t represent the value of predictor variable X at given time t where $0 \leq t \leq T$. Bayesian network representation for a particular time say t=0 is $P[X_0]$ which is the initial state.[BG08] For time t+1, the probability distribution is given by, $P(X[t+1]|X[t]) = \Pi_{i=1}^{N} P(X_t^i|\pi(X_t^i))$, and its joint probability distribution will be, $P(Y_{1:t}) = \Pi_{t=1}^{T}\Pi_{i=1}^{N} P(X_i^t|\pi(X_t^i))$.

3 Illustrative examples with R and Python

3.1 Example 1: Bayesian network of plant biomass, matrix type, and plant growth factors

Understanding the factors that contribute to the aggregation of plant biomass is essential to issues such as global change, agriculture, and human nutrition. With a steady growing human population and global issues, such as poverty, malnutrition, and lack of housing, our ability to understand the factors that influence plant biomass is critical to human subsistence. Therefore, a Bayesian network approach was applied to a plant biomass data set in order to understand the factors that contribute to plant growth.

The plant biomass data set was acquired from the website https://www.kaggle.com/abtabm/plant-growthhydroponics-and-soil-compound-dataset/version/1 and may be accessed through the UK Environmental Information Data Centre [HAN+19]. The data set was filtered to the subset of observations containing plant biomass data. The Bayesian network was learned using R package bnlearn [SSM19] and then visualized with bnviewer [Fer19].

The variables included in learning the Bayesian network were matrix, worms, ade_ug/ g (adenosine), zeatin_ug/g (zeatin), iP_ug/g (isopentenyladenosine), ABA_ug/g (abscisic acid), and plant biomass. Matrix was converted to a factor with two levels: soil or hydroponic, and worms was converted to a factor with two levels: yes or no. The variables for plant growth hormone levels of adenosine, zeatin, isopentenyladenosine, and abscisic acid were all numeric variables that were measured post-experiment time. Plant biomass was also a numeric variable that was measured post-experiment.

```
# converting character strings into factors
plant_biomass<-as.data.frame(plant_biomass)
plant_biomass$worms <- as.factor(plant_biomass$worms)
plant_biomass$matrix <- as.factor(plant_biomass$matrix)
str(plant_biomass)

# description of data frame
#'data.frame': 16 obs. of 7 variables:
# $ worms      : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 1 1 1 1 ...
# $ matrix     : Factor w/ 2 levels "Hydroponic","Soil": 2 2 2 2 2 2 2 2 2 2 ...
# $ ade_ug/g   : num 0.0365 0.0257 0.0199 0.0247 0.0282 ...
# $ zeatin_ug/g : num 0.000615 0.000592 0.000413 0.000869 0.001753 ...
# $ iP_ug/g    : num 3.59032 4.59154 5.84602 0.00112 0.00866 ...
# $ ABA_ug/g   : num 8.0e-06 8.0e-06 8.0e-06 2.9e-03 8.0e-06 ...
# $ plant_biomass: num 2.51 3.46 5.57 1.68 3.6 4.12 4.74 5.68 6.11 3.7 ...
```

My objective was to learn the structure and fit the parameters for a Bayesian network in order to understand the biological interactions among matrix type (soil vs hydroponic), presence or absence of earthworms, plant growth hormone levels, and plant biomass.

3.1.1 Bayesian network: Structure learning

The structure learning of a Bayesian network follows two basic types. One method of structure learning is *score-based*, using a learning algorithm and a scoring method for the probabilities. The other main type of Bayesian network structure learning is *constraint-based* algorithm. In this paper, I applied a score-based structure learning method employing a hill-climbing algorithm for searching and Bayesian Information Criterion (BIC) for scoring of the candidate Bayesian networks. The hill-climbing algorithm is a greedy search that learns one arc at a time, performs removal and reversal of arcs, and performs random restarts to avoid local optima. The hill-climbing algorithm produces a directed graphical representation of the network with no undirected arcs [Scu09].

Since the system has some inherent rules based on study design, I utilized a blacklist in structure learning. Items included in the blacklist in one direction, for instance, from a plant growth hormone to matrix type, are never present in the DAG. However, the interaction (directed arc) in the other direction, from matrix type to growth hormone, would be allowed.

```
# creating a blacklist
library(bnlearn)
bl_pb = tiers2blacklist(list("matrix","worms",c("ade_ug/g","zeatin_ug/g",
"iP_ug/g","ABA_ug/g")))
bl_pb

# directed arcs blacklisted
#     from         to
# [1,] "worms"     "matrix"
# [2,] "ade_ug/g"  "matrix"
# [3,] "zeatin_ug/g" "matrix"
# [4,] "iP_ug/g"   "matrix"
# [5,] "ABA_ug/g"  "matrix"
```

```
# [6,] "ade_ug/g"  "worms"
# [7,] "zeatin_ug/g" "worms"
# [8,] "iP_ug/g"   "worms"
# [9,] "ABA_ug/g"  "worms"
```

A whitelist was not included as to not force an interaction that may not be there. A directed arc that is whitelisted must be in the network and would not be allowed in the other direction [Scu09]. For instance, if we assume matrix type must have an impact on plant hormones, which in turn impacts plant biomass, then we are potentially ignoring hidden (latent) variables not included in the study. Hence, omission of a whitelist in learning the Bayesian network for the plant biomass data would lead to relying on the data, conditional probability scores, and learning algorithm in order to learn the structure of the network.

For network structure learning, a score-based method of learning utilized a hill-climbing algorithm, BIC (cond. Gauss.), and blacklist as described.

```
# learning Bayesian network structure
dag = hc(plant_biomass, blacklist = bl_pb, debug = TRUE)
dag

# Bayesian network learned via Score-based methods
#
#  model:
#   [worms][matrix][ade_ug/g][ABA_ug/g][zeatin_ug/g|matrix]
#   [plant_biomass|matrix][iP_ug/g|matrix:zeatin_ug/g]
#  nodes:                               7
#  arcs:                                4
#    undirected arcs:                   0
#    directed arcs:                     4
#  average markov blanket size:         1.14
#  average neighbourhood size:          1.14
#  average branching factor:            0.57
#
#  learning algorithm:                  Hill-Climbing
#  score:                               BIC (cond. Gauss.)
#  penalization coefficient:            1.386294
#  tests used in the learning procedure: 57
#  optimized:                           TRUE
```

The resulting learned Bayesian network structure model

$$[worms][matrix][ade][ABA][zeatin \mid matrix][plant\_biomass \mid matrix][iP \mid matrix : zeatin] \tag{3}$$

contained seven nodes (corresponding to the variables) and four directed arcs (connections) between the variables.

3.1.2 Bayesian network: Estimating parameters

After learning the structure, the parameters of each node (variable) were estimated using bn.fit from the bnlearn package [SSM19]. According to the estimated parameters for plant biomass, there was an increase in plant biomass for the matrix type soil (1.426) versus the matrix type hydroponic (0.572). The parameters for matrix were 0.375 for hydroponic and 0.625 for soil. The presence of earthworms had no apparent influence on plant biomass or any other variable.

```
# estimating the parameters of the local probability distributions
dag.trained = bn.fit(dag, plant_biomass)
dag.trained
```

```
#  Bayesian network parameters
#
#  Parameters of node worms (multinomial distribution)
#
# Conditional probability table:
#  No Yes
# 0.5 0.5
#
#  Parameters of node matrix (multinomial distribution)
#
# Conditional probability table:
# Hydroponic     Soil
#    0.375     0.625
#
#  Parameters of node ade_ug/g (Gaussian distribution)
#
# Conditional density: ade_ug/g
# Coefficients:
# (Intercept)
#  0.0261965
# Standard deviation of the residuals: 0.007482876
#
#  Parameters of node zeatin_ug/g (conditional Gaussian distribution)
#
# Conditional density: zeatin_ug/g | matrix
# Coefficients:
#                      0           1
# (Intercept) 0.0002279097 0.0007956838
# Standard deviation of the residuals:
#          0           1
# 9.218237e-05 4.038192e-04
# Discrete parents' configurations:
#      matrix
# 0  Hydroponic
# 1       Soil
#
#  Parameters of node iP_ug/g (conditional Gaussian distribution)
#
# Conditional density: iP_ug/g | matrix + zeatin_ug/g
# Coefficients:
#                       0            1
# (Intercept)  4.919131e-04  8.474563e+00
# zeatin_ug/g  2.148556e+00 -5.972772e+03
# Standard deviation of the residuals:
#           0           1
# 0.0004399629 2.8221295764
# Discrete parents' configurations:
#      matrix
# 0  Hydroponic
# 1       Soil
#
#  Parameters of node ABA_ug/g (Gaussian distribution)
#
# Conditional density: ABA_ug/g
# Coefficients:
```

```
# (Intercept)
# 0.001331787
# Standard deviation of the residuals: 0.002155222
#
#   Parameters of node plant_biomass (conditional Gaussian distribution)
#
# Conditional density: plant_biomass | matrix
# Coefficients:
#                0     1
# (Intercept) 1.755 4.117
# Standard deviation of the residuals:
#       0         1
# 0.5720052 1.4262854
# Discrete parents' configurations:
#      matrix
# 0  Hydroponic
# 1       Soil
```

3.1.3 Bayesian network: Visualization

The Bayesian network structure was visualized using bnviewer [Fer19] and can be seen in Figure 3.

```
# visualization of Bayesian network
library(bnviewer)
viewer(dag,
       edges.smooth = TRUE,
       bayesianNetwork.height = "400px",
       node.colors = list(background = "#f4bafd",
                          border = "#2b7ce9",
                          highlight = list(background = "#97c2fc",
                                           border = "#2b7ce9")),
       bayesianNetwork.layout = "layout_with_sugiyama")
       # viewer code from https://github.com/robson-fernandes/dbnlearn
```

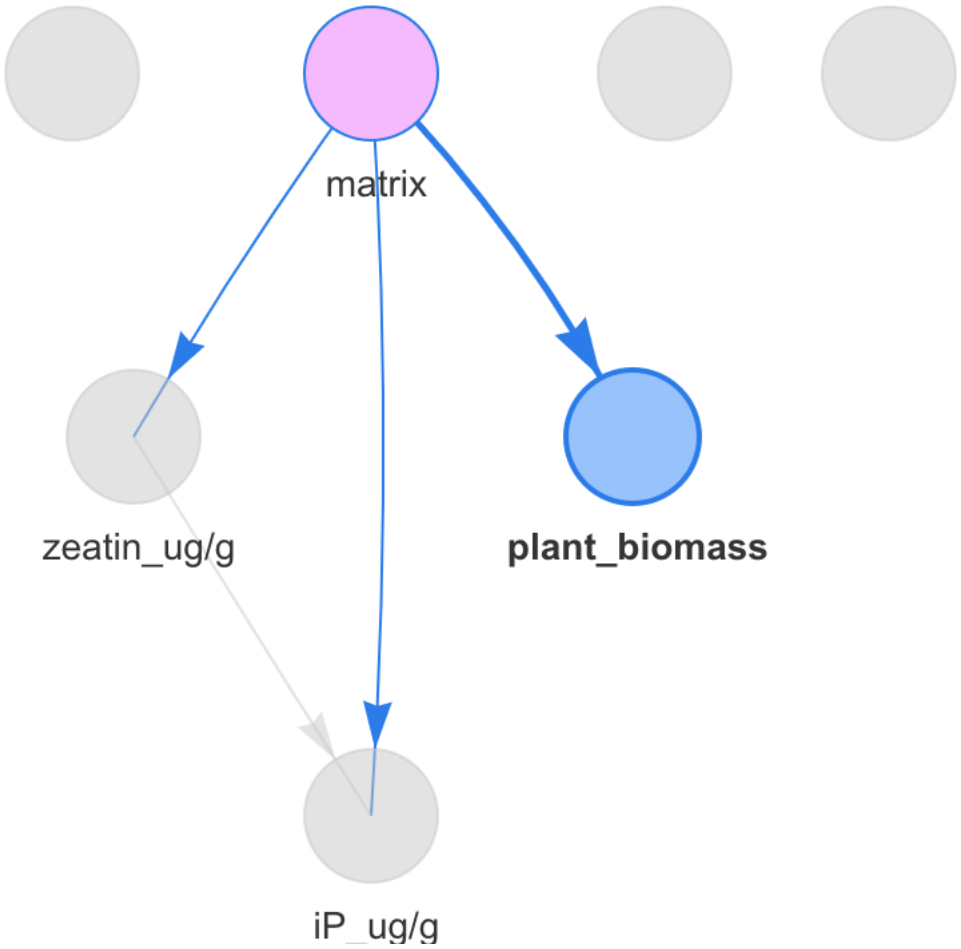


Figure 3: A Bayesian Network of factors impacting plant biomass. Based on the experiment and data, matrix type would influence plant biomass. The interaction between matrix and plant biomass did not seem to be directly related to plant hormone levels. The Bayesian network was visualized using R package bnviewer [Fer19]
.

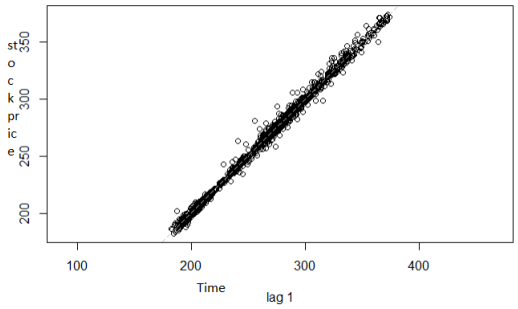
3.2 Example 2: Dynamic Bayesian network for predicting financial data

To predict a trend in the stock market or weather we need to know data from time to time - Time series data. Time series data is a series of data points ordered in time, where time will be the independent variable.

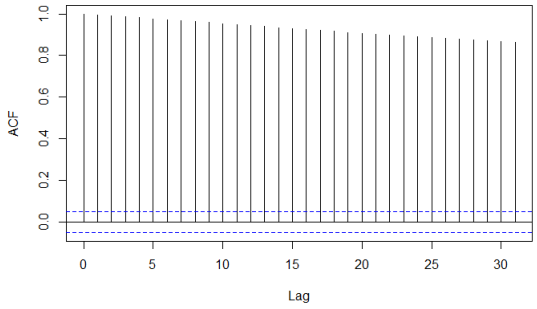
In this study we will design an univariate model using dynamic Bayesian networks and use that model to predict the stock price of a given stock. In other words, We are trying to predict future stock price based on the history stock price.

Yahoo Finance data is used for this study. The data is retrieved by using the package Quantmod. Getsymbol function in Quantmod retrieves the Open price, Volume, closing price and other details for the stock symbol and the time period that is passed.

A lag plot(See Fig. 4a) is used to Visulaize the data and that will help evaluate whether the values in a dataset or time series are random. If the data are random, the lag plot will exhibit no pattern. If the data are not random, the lag plot will exhibit a pattern. It also shows the outliers in the data. From the lagplot in 4a, we understand that the data is non random.



(a) Lag Plot for the stock Price

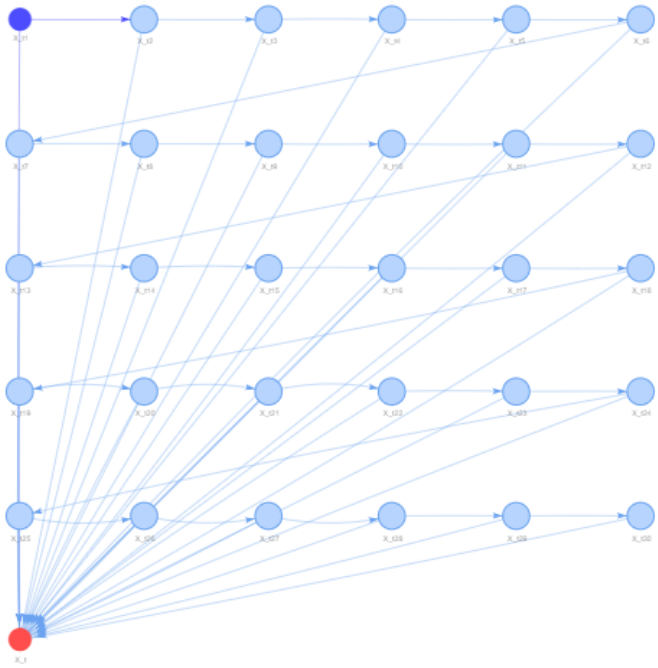


(b) Auto correlation Plot

Auto correlation plot is used to understand the correlation between the stock price at different time lags. If the co efficient at one or more lags are non zero then it means that the data is non random. In the plot, each bar represents the size and direction of the correlation. The co efficient for the stock price is more than zero and almost same with the time. It means that the data is non random. When the data are not random, it's a good indication that you need to use a time series analysis The auto correlation is moderately high at all lags and is dropping slowly indicating it is a non-stationary time series. And also we could see the previous lag is related to a lag. So it is possible to predict a future price based on the previous price. That is inline with the study where we are trying to predict the stock price based on the history price. 4b

3.2.1 Visualization

The R package named BNviewer is used to view the conditional dependecies and relationships between timeslices and within the timeslice. In the below fig, the nodes (in blue) are represented as $X_1, X_2..X_{30}$ for 30 days in a month. The plot is developed based on the model and is used to visualize the model. The dependencies between the varipus timeslices are represented as directed arcs. The target is the predicted stock price and it is represented as Y node in red. This gives a uderstanding of the data and the bayesian networks.



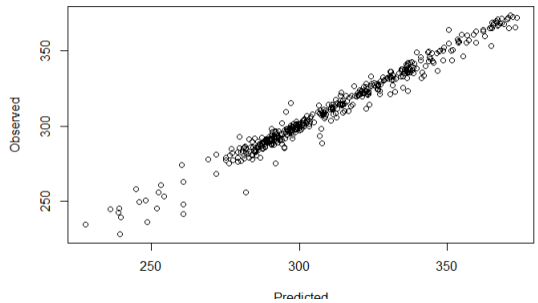
3.2.2 Data modelling and prediction

Dataset is split into train and test sets, so that there will be enough data in the training dataset for the model. Also, there will be enough data in the test set to effectively evaluate the model performance

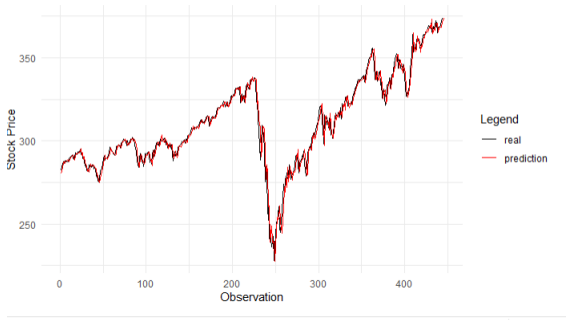
```
#70% Train 30% Test Data Set
percent = 0.7
n = nrow(X.ts)

trainIndex <- seq_len(length.out = floor(x = percent * n))
X.ts.train <- X.ts[trainIndex,]
X.ts.test <- X.ts[-trainIndex,]

#Train the model
ts.learning = dbn.learn(X.ts.train)

#Fit the data
ts.fit = dbn.fit(ts.learning, X.ts.train)
#Predict the stock price
stock.pred=dbn.predict(ts.fit, X.ts.test)
```

In this study, 70 percentage of the data will be used for Training and 30 percentage will be used for testing the model.

The package DBNLearn is used to fit the model, test the model and predict the future stock price. The training data is used to learn and fit a model. The testing data is used to test the model. The values that are predicted with the test data is plotted against the expected values. Below is the predicted data and the expected data 5a The model with real data and predicted data is shown in 5b. It shows that the



(a) Predicted Vs Observed values



(b) Predicted Vs Observed Plot

The mean absolute percentage error for the model is 0.009838961. Based on the percentage error and the comparison between real and predicted value looks good.

From the study it is evident that one can predict the future stock price based on the history using the Dynamic Bayesian network. The future work will include more predictor variables to the model (multivariate model) and compare it to the model that is developed here.

3.3 Example 3: Hidden Markov models for Speech Recognition

In this example we aim to highlight the effectiveness of Hidden Markov models in the domain of continuous speech recognition systems. We utilize the AudioMNIST dataset, which comprises a set of free spoken digits between 0 and 9, to train an HMM for each digit class to predict an appropriate sequence of phonemes. Additionally, we evaluate the classification accuracy of each HMM on a hold out set of the respective digits.

3.3.1 Preprocessing Audio

Because the AudioMNIST dataset is composed of 3,000 individual .wav files, we must convert the dataset into digital representations of each audio file in order to train the HMMs. An example of the digitized representation after this conversion is shown below in Figure 6.

```
import os
import numpy as np
import librosa
from librosa import display
from hmmlearn import hmm
import matplotlib.pyplot as plt

# Define file path to .wav files + list directory
FILE_PATH = "/Users/ConnorBaugh/Documents/R Files/AudioMNIST/recordings/"
WAV_ARR = os.listdir(FILE_PATH)

# Create lists to store sorted sequence data + lengths
df_0, df_1, df_2, df_3, df_4, df_5, df_6, df_7, df_8, df_9 = ([] for i in range(10))
len_0, len_1, len_2, len_3, len_4, len_5, len_6, len_7, len_8, len_9 = ([] for i in range(10))
dict_d = {'0': df_0, '1': df_1, '2': df_2, '3': df_3, '4': df_4,
          '5': df_5, '6': df_6, '7': df_7, '8': df_8, '9': df_9}
dict_l = {'0': len_0, '1': len_1, '2': len_2, '3': len_3, '4': len_4,
          '5': len_5, '6': len_6, '7': len_7, '8': len_8, '9': len_9}
filtered = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

# Loop through each .wav file, digitize + sort into appropriate digit list
for i in range(len(WAV_ARR)):
    audio, sample_rate = librosa.load(FILE_PATH + WAV_ARR[i])
    mfcc = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=12).transpose().tolist()
    for j in range(len(filtered)):
        if WAV_ARR[i][0] == filtered[j]:
            current_df = dict_d[filtered[j]]
            current_len = dict_l[filtered[j]]
            current_df.extend(mfcc)
            current_len.append(len(mfcc))
            break
```

```
# Convert .wav file into digitized representation and plot
audio_6, sample_rate_6 = librosa.load(FILE_PATH + WAV_ARR[-1])
librosa.display.waveshow(y=audio_6, sr=sample_rate_6)
plt.ylabel('Amplitude')
plt.title('Waveplot of the digit 6')
plt.show()
```

In order to get the data into a format appropriate for the HMMs, it must be further processed into *Mel Frequency Cepstral Coefficients* (MFCCs). MFCCs are better suited for HMMs because each MFCC roughly represents each phoneme we are trying to model [Nai18]. The same digit 6 is displayed in Figure 7 in MFCC form.

```
# Process the previous waveplot into MFCC format and plot
mfcc_6 = librosa.feature.mfcc(y=audio_6, sr=sample_rate_6, n_mfcc=12)
librosa.display.specshow(mfcc_6)
plt.colorbar()
plt.xlabel('Samples')
plt.ylabel('Coefficients')
```

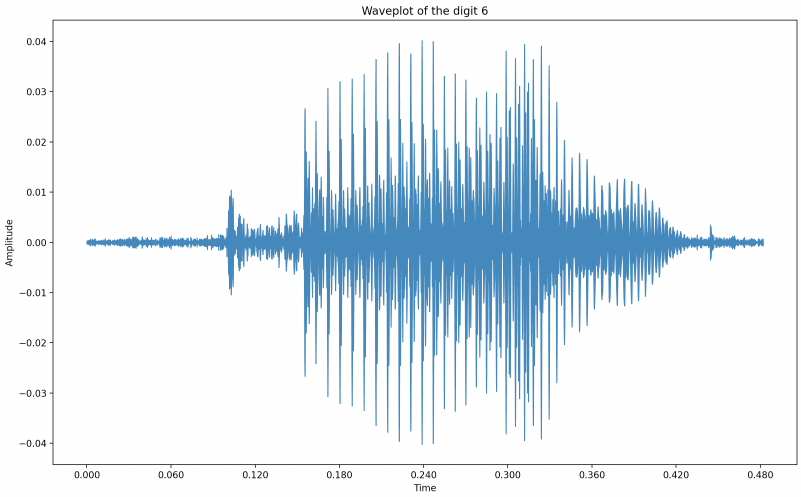


Figure 6: Waveplot of the digit 6

```
plt.title('MFCC of the digit 6')
plt.show()
```

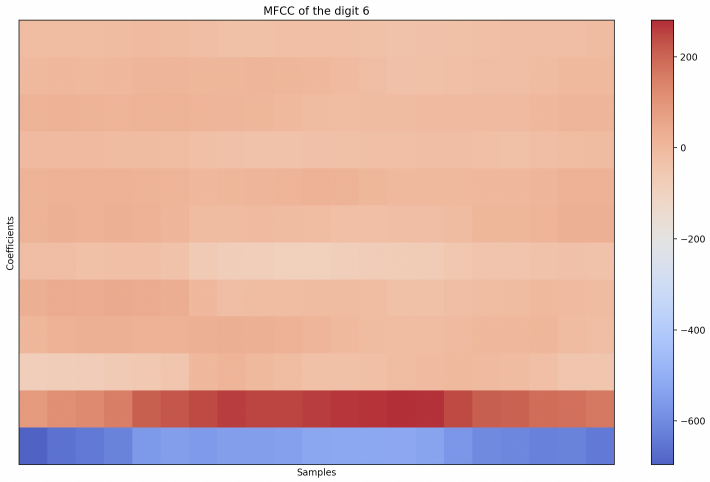


Figure 7: MFCC representation of the digit 6

3.3.2 Speech Recognition with HMMs

After converting all of the audio samples into MFCCs, we are finally ready to train the HMMs for each digit.

```
# Create lists to store validation sets + lengths
vdf_0, vdf_1, vdf_2, vdf_3, vdf_4, vdf_5, vdf_6, vdf_7, vdf_8, vdf_9 = ([] for i in range(10))
vlen_0, vlen_1, vlen_2, vlen_3, vlen_4, vlen_5, vlen_6, vlen_7, vlen_8, vlen_9 = ([] for i in
    range(10))
val_df = [vdf_0, vdf_1, vdf_2, vdf_3, vdf_4, vdf_5, vdf_6, vdf_7, vdf_8, vdf_9]
```

```
val_len = [vlen_0, vlen_1, vlen_2, vlen_3, vlen_4, vlen_5, vlen_6, vlen_7, vlen_8, vlen_9]

# Define HMMs for each digit class with states set to the number of phonemes present in each digit
    + 1 to account for silence
dict_hmm = {'0': hmm.GaussianHMM(n_components=5, random_state=8), '1':
    hmm.GaussianHMM(n_components=4, random_state=8),
           '2': hmm.GaussianHMM(n_components=3, random_state=8), '3':
               hmm.GaussianHMM(n_components=4, random_state=8),
           '4': hmm.GaussianHMM(n_components=4, random_state=8), '5':
               hmm.GaussianHMM(n_components=4, random_state=8),
           '6': hmm.GaussianHMM(n_components=5, random_state=8), '7':
               hmm.GaussianHMM(n_components=6, random_state=8),
           '8': hmm.GaussianHMM(n_components=3, random_state=8), '9':
               hmm.GaussianHMM(n_components=4, random_state=8)}

# Create validation sets for each digit class + use training sets to fit respective digit HMMs
for i in range(len(filtered)):
    len_split = int(0.8*len(dict_l[filtered[i]]))
    df_split = np.sum(dict_l[filtered[i]][:len_split])
    val_df[i].extend(dict_d[filtered[i]][df_split:])
    val_len[i].extend(dict_l[filtered[i]][len_split:])
    dict_hmm[filtered[i]].fit(dict_d[filtered[i]][:df_split],
        lengths=dict_l[filtered[i]][:len_split])
```

We have separated the samples into the appropriate training and validation sets for each of the respective digits and have fit 10 HMMs for each of the digit training sets with n states set depending on the number of phonemes present in the respective spoken digit. Note that for each HMM we include an additional state to account for any silence present in the audio recordings. Each digit and it's associated phonemes can be found in Table 1 [Hui19].

Digit	Phonemes
0	z iy r ow
1	w ah n
2	t uw
3	th r iy
4	f ao r
5	f ay v
6	s ih k s
7	s eh v ah n
8	ey t
9	n ay n

Table 1: Phonemes of each digit

Now that the models are fit, we can predict the states for each observation in the validation sets to get an idea of how well the models generalize to unseen samples. Figure 8 below shows the predicted states of the same spoken digit from the digit 6 validation set against the waveplot of the data.

```
# Get predicted state sequence from the digit 6 HMM + plot against the waveplot of the data
mfcc_6 = mfcc_6.transpose().tolist()
predict = dict_hmm['6'].predict(mfcc_6).tolist()
fig = plt.figure()
ax1 = fig.subplots()
ax2 = ax1.twiny()
```

```
ax1.margins(0)
ax2.margins(0)
ax1.set_xlabel('Samples')
fig.text(0.06, 0.5, 'Amplitude', ha='center', va='center', rotation='vertical')
plt.title('Predicted Phoneme Sequence over Waveplot of the digit 6')
librosa.display.waveshow(y=audio_6, sr=sample_rate_6, color='w', linewidth=1, ax=ax2)
colors = {0: 'g', 1: 'r', 2: 'y', 3: 'b', 4: 'c', 5: 'm'}
for i, state in enumerate(predict):
    ax1.axvspan(i, int(i+1), color=colors[state])
plt.show()
```

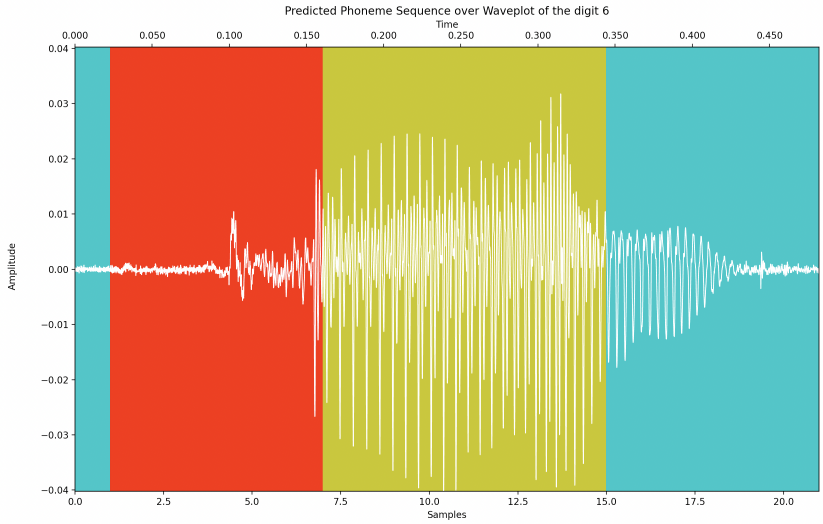


Figure 8: Predicted states vs Waveplot of digit 6 from validation set

3.3.3 Digit Classification and Results

To perform digit classification, we first calculate the log likelihood of each digit in the validation set fitting the distribution of each HMM. Then, we take whichever HMM receives the highest likelihood score and count that as the digit's assignment. Lastly, we use each digit's classification to record the accuracy of each HMM, defining accuracy as the number of digits correctly assigned to their respective class over the total number of digits in the validation set. The results of the digit classification are displayed in Table 2 below.

```
# For each sample in the validation sets, find the log likelihood of belonging to each class, if
    the max value is for the correct digit then the classification accuracy for that HMM increases
counts = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
accuracy = []
for i in range(len(val_len)):
    start = 0
    for j in range(len(val_len[i])):
        end = start + val_len[i][j]
        scores = []
        for k in range(len(filtered)):
            scores.append(dict_hmm[filtered[k]].score(val_df[i][start:end]))
        start += val_len[i][j]
        if scores.index(max(scores)) == i:
            counts[i] += 1
```

```
    accuracy.append(counts[i]/len(val_len[i]))
print(accuracy)
```

Digit	Val Acc
0	0.87
1	0.68
2	0.45
3	0.78
4	0.92
5	0.85
6	0.83
7	0.90
8	0.58
9	0.92

Table 2: Accuracy of each Digit HMM

After completing this example, we have seen that the digit HMMs are generally good predictors of audio sequences and reliable spoken digit classifiers. With the exception of the HMMs for digits 2, 3 and 8, each did an acceptable job at correctly classifying unseen sets of samples, with an average accuracy across all HMMs of approximately 78%.

References

[299] Dynamic bayesian networks (dbns), 1999.

[BG08] Irad Ben-Gal. Bayesian networks. *arXiv:2002.00498*, 2008.

[Fer19] Robson Fernandes. Bnviewer: Interactive visualization of bayesian networks. *R Package Version 0.1*, 4, 2019.

[HAN+19] K.E. High, P.D. Ashton, M. Nelson, E.L. Rylott, J.E. Thomas-Oates, and M.E. Hodson. Concentrations of plant growth promoting compounds in soils and hydroponics due to the interaction of plants and earthworms, 2019.

[Hec20] David Heckerman. A tutorial on learning with bayesian networks. *arXiv:2002.00498*, 2020.

[Hui19] Jonathan Hui. Speech recognition — kaldi, 2019.

[Lè13] Radhakrishnan NagarajanMarco ScutariSophie Lèbre. *Bayesian Networks in R*. SPRINGER Press, 2013.

[Mur02] Kevin P. Murphy. A tutorial on dynamic bayesian networks. *cs.ub.ca*, 2002.

[Nai18] Pratheeksha Nair. The dummy's guide to mfcc, 2018.

[Pea11] Judea Pearl. Bayesian networks. 2011.

[Pea22] Judea Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *Probabilistic and Causal Inference: The Works of Judea Pearl*, pages 129–138. 2022.

[PSD+20] Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Konstantinos Georgatzis, Paul Beaumont, and Bryon Aragam. Dynotears: Structure learning from time-series data. In *International Conference on Artificial Intelligence and Statistics*, pages 1595–1605. PMLR, 2020.

[Scu09] Marco Scutari. Learning bayesian networks with the bnlearn r package. *arXiv preprint arXiv:0908.3817*, 2009.

[SSM19] Marco Scutari, Maintainer Marco Scutari, and Hiton-PC MMPC. Package 'bnlearn'. *Bayesian network structure learning, parameter learning and inference, R package version*, 4(1), 2019.

[Ver21] Yugesh Verma. A guide to hidden markov model and its applications in nlp, 2021.