Probabilistic Graphical Models and Inference Methods for Handwriting

Danyang Chen
danyangc@buffalo.edu

Duc Thanh Anh Luong
ducthanh@buffalo.edu

Abstract

In this report, we present our approach for generating and learning probabilistic graphical models (PGMs) from a dataset of handwriting of word "*and*". We give a comparison between different PGMs of different generations involving in this dataset. We also use inference methods to determine the mean and entropy of each distribution. With the learned PGMs, we determine the rarest and most common handwriting in each dataset. We also show an application PGMs in solving the problem of missing value imputation.

1 Introduction

In this report, we solve the problem of learning PGMs from the given data and perform inferences from the learned models. In particular, we constructed the Bayesian Networks and Markov Networks from the dataset of handwriting and performed inferences on these learned models. We also present our approach of solving the problem of missing value imputation using these learned PGMs.

In section 2, we give a brief description about the dataset. We also give an explanation on dividing the dataset into different groups and draw a roadmap for further experiments in section 5. In section 3, we present our approach for constructing Bayesian Network, learning its parameters, and perform inferences on it. In section 4, we present our approach for constructing Markov Network and learning its parameters. In section 5, we give experimental results with Bayesian Network and Markov Network. Finally, in section 6, we conclude our works.

2 Dataset

We are given a dataset of children handwriting from a large number of students as they are learning (2nd grade) or have just learned (3rd and 4th grade) how to produce cursive and printed writing for 3 consecutive years 2012 - 2013 - 2014.

From the writing samples, the word "*and*" is extracted and features are generated from it. All features extracted from this process are discrete with maximum 5 distinct values. Figure 2 in project description gave a detailed explanation for each feature and its corresponding value. In short, there are totally 12 features for cursive handwriting and 12 features for handprint writing. In the dataset, there are some files with only 11 features which don't match the data description so we simply ignore those files.

In the dataset, there are missing and inconsistent values which are encoded with value -1 (missing value) and 99 (inconsistent). In our experiments when building the PGMs from dataset, we simply ignore the missing value whenever it appears and consider inconsistent value (99) as another value for that corresponding feature.

In order to compare the difference between handwriting for different generations, we group our dataset in the following:

- Cursive: for cursive handwriting, we construct 3 distinct groups of student which study in the same class in 2 consecutive years. The detail of each group is as follow:
  - Generation 1: consists of samples from students of 2011-2012 Grade 4 and 2012-2013 Grade 5 (471 samples)
  - Generation 2: consists of samples from students of 2011-2012 Grade 3 and 2012-2013 Grade 4 (559 samples)
  - Generation 3: consists of samples from students of 2012-2013 Grade 3 (176 samples)
- Handprint: for handprint handwriting, we construct 4 distinct groups of student which study in the same class in 3 consecutive years. The detail of each group is as follow:
  - Generation 1: consists of samples from students of 2011-2012 Grade 4 and 2012-2013 Grade 5 (503 samples)
  - Generation 2: consists of samples from students of 2011-2012 Grade 3, 2012-2013 Grade 4 and 2013-2014 Grade 5 (742 samples)
  - Generation 3: consists of samples from students of 2011-2012 Grade 2, 2012-2013 Grade 3 and 2013-2014 Grade 4 (1030 samples)
  - Generation 4: consists of samples from students of 2011-2012 Grade 1 (170 samples)

In section 3 and 4, we will construct PGMs for each generation mentioned above. We also construct PGMs for the whole dataset for both cursive and handprint data. With PGMs learned from the whole dataset, we can identify the common writing and rare writing in the dataset.

3 Bayesian Networks for handwriting dataset

3.1 Independence test

Since the Bayesian Network captures the set of independencies between variables, the first step that we need to do is to have an independence test, which we used Pearson's Chi-squared test [2]. The subroutine to perform this independence test can be found in the file *chi_square.m* of our code.

In brief, the Chi-squared test receive input including X_1, X_2 and X_3 where X_1 and X_2 are random variables while X_3 is a set of random variables. It checks whether $(X_1 \perp X_2|X_3)$ by using hypothesis testing. In particular, for each assignment x_3 of X_3, we perform the test $(X_1 \perp X_2|x_3)$. The formula for computing the chi-square value is as follow:

$$\chi^2 = \sum_{x_1} \sum_{x_2} \frac{[(O_{x_1,x_2}|x_3) - (E_{x_1,x_2}|x_3)]^2}{(E_{x_1,x_2}|x_3)} \tag{1}$$

where the first sum loops for all values x_1 of X_1, the second sum loops for all values x_2 of X_2, $(O_{x_1,x_2}|x_3)$ is the total number of occurrences of x_1, x_2, x_3 and $(E_{x_1,x_2}|x_3)$ is the expected value of number of occurrences of x_1, x_2, x_3 with an assumption that $(X_1 \perp X_2|x_3)$. In brief, the equation [1] computes the score of deviation from independence of our variables X_1, X_2 given values of X_3. If the score is too big, we then reject our hypothesis $(X_1 \perp X_2|x_3)$. More precisely, we convert the χ^2 score into probability and reject the hypothesis if the false rejection probability is more than 5%.

Chi-squared test accepts the hypothesis $(X_1 \perp X_2|X_3)$ when it doesn't reject any sub-hypotheses $(X_1 \perp X_2|x_3)$ for any value x_3.

3.2 Structure Learning

In this section, we explain our approach for constructing Bayesian Network from a given dataset. The approach that we chose follows closely the textbook [1, p. 78-92].

Before going to explain our algorithm, we define some terms that we will use in the subsequent sections.

Definition 3.1 *We say that a graph K is a perfect map (P-map) for a set of independencies I if we have that $I(K) = I$. We say that K is a perfect map for a distribution P if $I(K) = I(P)$.*

Definition 3.2 *If X and Y are not adjacent in G^* (an I-map of a distribution P) then we can find a set U such that $P \models (X \perp Y|U)$. We call this set U a witness of their independence.*

Definition 3.3 *A v-structure $X \rightarrow Z \leftarrow Y$ is an immorality if there is no direct edge between X and Y.*

Definition 3.4 *Let G be a DAG. A chain graph[1] K, is a class $PDAG$ of the equivalence class[2] of G if shares the same skeleton as G and contains a directed edge $X \rightarrow Y$ if and only if all G' that are I-equivalent to G contain the $X \rightarrow Y$.*

3.2.1 Building a skeleton for Bayesian Network

Assume that our distribution P in handwriting dataset has a perfect map. Let G^* be a perfect map for our distribution. The subroutine Build-PMap-Skeleton recovers the undirected skeleton for a distribution P that has a P-map. The pseudo-code for this subroutine can be written as follow:

Algorithm 1 Build-PMap-Skeleton

Input: $X = \{X_1, \cdots, X_n\}$ (set of random variables), P (distribution over X), d (bound on witness set)
Output: H (undirected graph over X), U (the set of witness)

```
Let H be a complete undirected graph over X
for X_i, X_j ∈ X do
    U_{X_i,X_j} = ∅
    for U ∈ Witness(X_i, X_j, H, d) do
        if P ⊨ (X_i ⊥ X_j|U) then
            U_{X_i,X_j} ← U
            Remove X_i − X_j from H
            break
        end if
    end for
end for
return H, {U_{X_i,X_j} : i, j ∈ {1, ··· , n}}
```

In the given pseudo-code of Algorithm 1, it sequentially detects an edge that is not in the skeleton and removes it by checking whether there exists a set U such that $(X_i \perp X_j|U)$.

One assumption that we make when running Build-PMap-Skeleton is that we limit the size of witness to d. Moreover, we can efficiently search for the set of witness U by enumerating it in increasing order of size.

3.2.2 Finding immoralities in skeleton

Having found the skeleton of P-map for distribution P, we now can add direction to some of the edges of the undirected graph returned in Build-PMap-Skeleton by checking the v-structure in the skeleton. Notice that, in a v-structure $X \rightarrow Z \leftarrow Y$, the information can flow from X to Y given Z. Therefore, by checking a path $X - Z - Y$ in the skeleton, if Z is not in witness set of X and Y, we can add the orientation for this v-structure as $X \rightarrow Z \leftarrow Y$. The formal description for above observation is given in the Algorithm 2

3.2.3 Build PDAG

Having constructed the skeleton of the perfect map G^*, and after adding the orientation for all immoralities that we have found in the skeleton, we can add some more orientation for this resulted graph by 3 rules given in Figure 1. A formal specification for this is given in Algorithm 3.

At this point, we have found the PDAG that captures all independencies of a distribution P. All remaining undirected edges can be oriented arbitrarily without losing any independencies in the

[1] a chain graph is graph which may have both directed and undirected edges but without any directed cycles
[2] 2 graphs that are in the same equivalent class if they capture the same set of independencies

Algorithm 2 Find-Immoralities

Input: $X = \{X_1, \cdots, X_n\}$ (set of random variables), S (Skeleton), $\{U_{X_i,X_j} : i, j \in \{1, \cdots, n\}\}$ (witness found by Build-PMap-Skeleton)
Output: K (Skeleton with edges involving in immoralities being directed)

$K \leftarrow S$
for X_i, X_j, X_k such that $X_i - X_j - X_k \in S$ and $X_i - X_k \notin S$ **do**
    //$X_i - X_j - X_k$ is a potential immorality
    if $X_j \notin U_{X_i,X_k}$ **then**
        Add orientation $X_i \rightarrow X_j$ and $X_j \leftarrow X_k$ to K
    end if
end for
return K

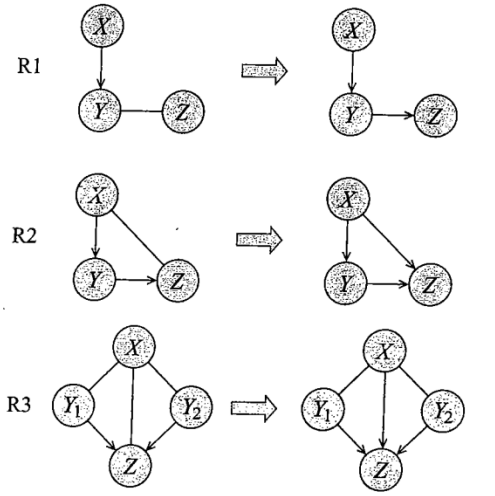


Figure 1: Rules for orienting edges in PDAG

Algorithm 3 Build-PDAG

Input: $X = \{X_1, \cdots, X_n\}$ (set of random variables), P (distribution of interest)
Output: K (PDAG of P)

$S, \{U_{X_i,X_j}\} \leftarrow$ Build-PMap-Skeleton(X, P)
$K \leftarrow$ Find-Immoralities$(X, S, \{U_{X_i,X_j}\})$
while not converged **do**
    Find a subgraph in K matching the left-hand side of a rule R1-R3
    Replace the subgraph with the right-hand side of the rule.
end while
return K

original distribution P. Theorem 3.10 in the textbook [1, p. 90] verifies the correctness of the algorithm.

3.3 Parameters Learning

The parameter set of Bayesian Network given its directed graph is the set of conditional probability distribution (CPD) which capture the local independencies of the distribution. In order to compute these CPDs, we follow the Maximum Likelihood Estimation (MLE) approach.

Let G be the Bayesian Network and θ be the set of parameters in G. For a dataset D with M samples $\{\xi[1], \cdots, \xi[M]\}$, the likelihood of parameter set θ given the dataset D is computed as follow:

$$
\begin{aligned}
L(\theta : D) &= \prod_{m=1}^{M} P_G(\xi[m] : \theta) && (2)\\
&= \prod_{m=1}^{M} \prod_{i} P(x_i[m] | pa_{X_i}[m] : \theta) && (3)\\
&= \prod_{i} \prod_{m=1}^{M} P(x_i[m] | pa_{X_i}[m] : \theta) && (4)\\
&= \prod_{i} L_i(\theta_{X_i|pa_{X_i}} : D) && (5)
\end{aligned}
$$

where $L_i(\theta_{X_i|pa_{X_i}} : D) = \prod_{m=1}^{M} P(x_i[m]|pa_{X_i}[m] : \theta)$ captures the local likelihood given the dataset.

At this point, our MLE formulation turns out to be the set of local MLE formulations which can be solved independently.

The local MLE is governed by a set of parameters $\{\theta_{x|u}$ for $x \in Val(X_i)$ and $u \in Val(pa_{X_i})\}$ of multinomial distribution. Formally, we can write the local likelihood as follow:

$$
\begin{aligned}
L_i(\theta_{X_i|pa_{X_i}} : D) &= \prod_{m=1}^{M} \theta_{x_i[m]|pa_{X_i}[m]} && (6)\\
&= \prod_{u \in Val(pa_{X_i})} \left[\prod_{x \in Val(X_i)} \theta_{x|u}^{M[u,x]} \right] && (7)
\end{aligned}
$$

where $M[u, x]$ is the occurrences of u and x in the dataset D.

By simple calculation, we can show that MLE parameters of multinomial distribution can be computed as follow:

$$
\hat{\theta_{x|u}} = \frac{M[u, x]}{M[u]} \tag{8}
$$

where $M[u]$ is the number of occurrences of u in the dataset D.

3.4 Sampling for Bayesian Network

3.4.1 Ancestral sampling

When sampling from Bayesian Network with ancestral sampling, we firstly order all random variables in topological order and sample each random variable in this order. The formal specification for this algorithm is given in Algorithm 4

3.4.2 Gibbs sampling

When sampling Bayesian Network with Gibbs sampling, we first determine the order of random variables that we want to sample. The first sample can be obtained in some arbitrary initial distribution. The next sample will be obtained by sampling each random variable in the pre-determined

Algorithm 4 Ancestral-Sample

Input: B (Bayesian Network over X - set of all random variables)
Output: $(x_1, \cdots, x_n)$
1: Let $X_1, \cdots, X_n$ be a topological ordering of X
2: **for** $i = 1, \cdots, n$ **do**
3: $u_i \leftarrow x(Pa_{X_i})$ //Assignment to Pa_{X_i} in $x_1, \cdots, x_{i-1}$
4: Sample x_i from $P(X_i|u_i)$
5: **end for**
6: **return** $(x_1, \cdots, x_n)$

order given the previous value of all other random variables. The formal description of Gibbs sampling can be given in Algorithm 5.

Algorithm 5 Gibbs-Sample

Input: X (set of all random variables), Φ (set of factors defining P_Φ), $P^{(0)}(X)$ (initial state distribution), T (number of samples)
Output: $x^{(0)}, \cdots, x^{(T)}$
1: Sample $x^{(0)}$ from $P^{(0)}(X)$
2: **for** $t = 1, \cdots, T$ **do**
3: $x^{(t)} \leftarrow x^{(t-1)}$
4: **for** each $X_i \in X$ **do**
5: Sample $x_i^{(t)}$ from $P_\Phi(X_i|x_{-i})$ // Change X_i in $x^{(t)}$
6: **end for**
7: **end for**
8: **return** $x^{(0)}, \cdots, x^{(T)}$

3.5 Inference Methods

In inference for Bayesian Network, we mostly have to deal with 2 problems:

- Approximate the solution by using samples generated from the distribution of Bayesian Network, which we have explained 2 methods: ancestral sampling and Gibbs sampling.
- Compute the joint probability of a sample in Bayesian Network, which we will explain in the next section.

3.5.1 Compute the joint probability in Bayesian Network

Given a Bayesian Network B and a data sample x, the joint proability of x can be computed as follow:

$$P(x) = \prod P(x_i|Pa_{X_i} = Pa_{x_i}) \tag{9}$$

3.5.2 Mean of a distribution

The mean of a distribution in a Bayesian Network can be approximately computed with M samples $\{x_1, \cdots, x_M\}$ as follow:

$$\hat{E}[p(x)] = \frac{1}{M} \sum_{k=1}^{M} x_k \tag{10}$$

3.5.3 Entropy of a distribution

The entropy of a distribution in a Bayesian Network can be approximately computed with M samples $\{x_1, \cdots, x_M\}$ as follow:

$$\hat{H}[p(x)] = -\frac{1}{M}\sum_{k=1}^{M}\ln(p(x_k)) \tag{11}$$

3.5.4 Relative entropy between two distributions

The relative entropy (KL divergence) between 2 distributions can be approximately computed with M samples $\{x_1, \cdots, x_M\}$ as follow:

$$\hat{KL}[p||q] = -\frac{1}{M}\sum_{k=1}^{M}[\ln(q(x_k)) - \ln(p(x_k))] \tag{12}$$

4 Markov Networks for handwriting dataset

4.1 Structure Learning

For a distribution P to have a corresponding Markov Network, the distribution must be positive (no entry in the joint probability can be zero).

Assuming that our distribution P is positive, let H be the Markov Network corresponding to P. We also denote that $X = \{X_1, \cdots, X_n\}$ be the set of random variables in distribution P.

In order to construct Markov Network, there are 2 naive ways to do it:

- Pairwise independencies: if the edge $\{X_i, X_j\}$ is not in H, then X_i and X_j must be independent given all other nodes in the graph regardless of which other edges the graph contains. Thus, at the very least, to guarantee that H is an I-map, we must add direct edges between all pairs of nodes X_i and X_j such that

  $$P \not\models (X_i \perp X_j | X - \{X_i, X_j\}) \tag{13}$$

  We can now define H to include an edge $X_i - X_j$ for all X_i, X_j for which equation [13] holds.

- Markov blanket: in this approach, we use the local independencies and the notion of minimality. For each variable X, we define the neighbors of X to be a minimal set of Y that render X independent of the rest of the nodes. More precisely, we define:

  Definition 4.1 *A set U is a Markov blanket of Y in a distribution P if $Y \notin U$ and if U is a minimal set of nodes such that*

  $$(Y \perp X - \{Y\} - U | U) \in I(P) \tag{14}$$

  We then define a graph H by introducing an edge $\{Y, Z\}$ for all Y and all $Z \in MB_P(Y)$ where $MB_P(Y)$ is a set of Markov blanket of Y.

Although 2 methods above give an intuition how Markov Network can be constructed, they are not tractable because both involve the entire set of variables X and hence require measuring the probability of exponentially many events. Moreover, independencies that involve many variables lead to fragmentation of the data, and are much harder to evaluate without error. To estimate the distribution sufficiently well as to evaluate these independencies reliably, we would need exponentially many data points.

Suppose that in the Markov Network that we are looking for, the size of the set of Markov blanket for every node is bounded above by d. Given our assumption and perfect independence test, the Build-PMap-Skeleton procedure of algorithm 1 reconstructs the correct Markov structure H.

4.2 Parameters Learning

In order to find parameters for Markov Network, we need to have a finer representation for our Markov Network. In this section, we first explain the Log-linear model to represent Markov Network. Then, we define the set of features associate with our Markov Network. Finally, the weight of those features will be determined by gradient descent.

4.2.1 Log-linear models

First, we give a formal definition of log-linear model which we will use later on to compute the parameters of Markov Network.

Definition 4.2 *A distribution P is a log-linear model over a Markov network H if it is associated with:*

- *a set of features $F = \{f_1(D_1), \cdots, f_k(D_k)\}$ where each D_i is a complete subgraph in H*
- *a set of weights $\theta_1, \cdots, \theta_k$ such that*

$$P(X_1, \cdots, X_n) = \frac{1}{Z} \exp\left[-\sum_{i=1}^{k} \theta_i f_i(D_i)\right]$$

4.2.2 Choosing set of features

From above definition, given a Markov Network structure, for every clique in the Markov Network, there is at least a feature that includes all variables in that clique. A question is how many features for each clique and how to define them. One easy approach that we choose is binary indicator feature. For every clique, the number of features will be the number of possible assignments for all random variables in that clique and each feature can be computed by using indicator function.

For example, if we have a clique of 2 random variables A and B, where $Val(A) = \{a_0, a_1\}$ and $Val(B) = \{b_0, b_1\}$. So the number of features for this clique is 4 and each feature can be defined as follow:

$$f_{a_i,b_j}(A, B) = 1(A = a_i)1(B = b_j) \text{ for } i, j \in \{0, 1\}$$

4.2.3 Gradient descent

Finally, we need to find the set of weight $\theta = \{\theta_1, \cdots, \theta_k\}$ for our distribution associated with log-linear model.

Let D be a set of samples $\{\xi[1], \cdots, \xi[M]\}$. The likelihood of our parameter θ given the dataset D is:

$$L(\theta : D) = \prod_{m=1}^{M} \frac{1}{Z(\theta)} \exp\left[-\sum_{i=1}^{k} \theta_i f_i(\xi[m])\right] \tag{15}$$

The log-likelihood can be written as follow:

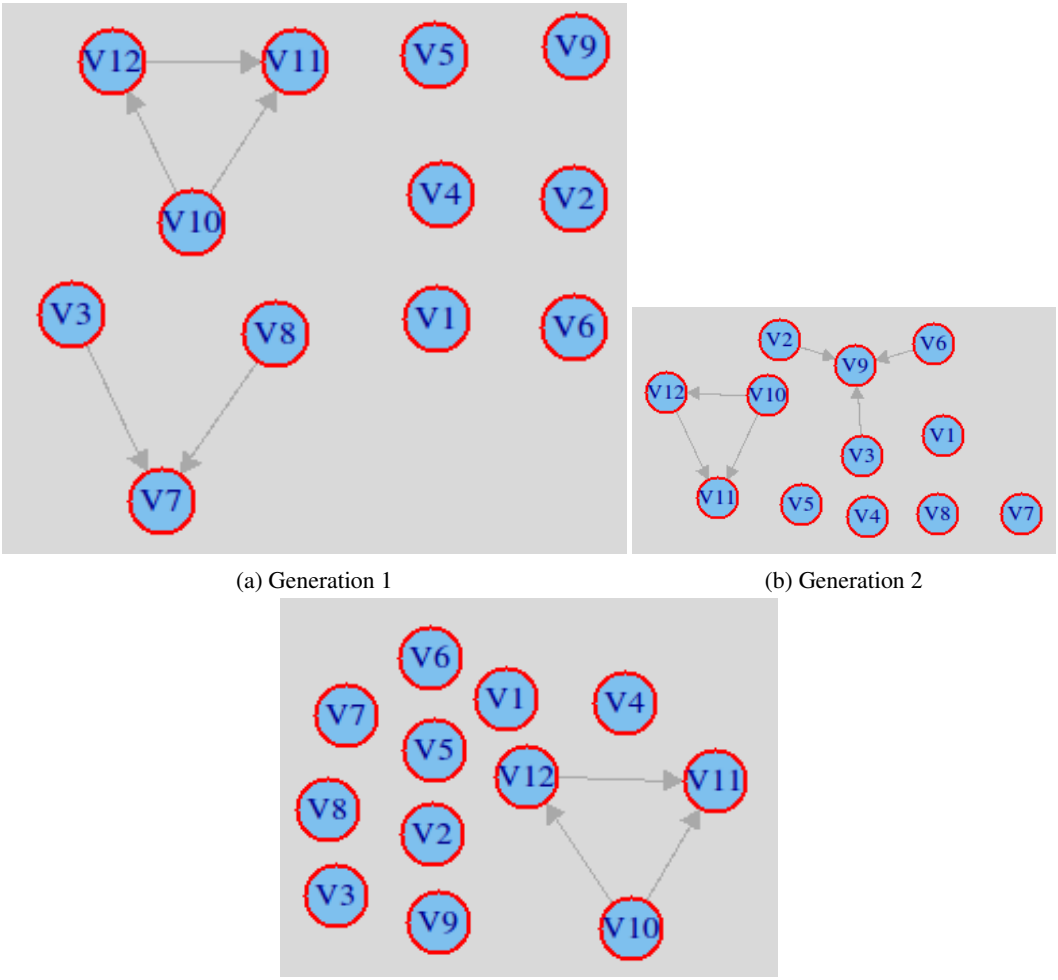
$$\log L(\theta : D) = \sum_i \theta_i \left(\sum_{m=1}^{M} f_i(\xi[m])\right) - M \ln Z(\theta) \tag{16}$$

In order to maximize the above log-likelihood, we can use the gradient descent to find the optimal parameter θ. In gradient descent, the derivative with respect to parameters can be shown as follow:

$$\frac{\partial}{\partial \theta_i} \frac{1}{M} \log L(\theta : D) = E_D[f_i] - E_\theta[f_i] \tag{17}$$

5 Experimental results

All the code that is used to perform experiments in this section can be found in the following link: `https://github.com/luongthanhanhduc/cse674`



(a) Generation 1

(b) Generation 2

(c) Generation 3

Figure 2: Bayesian Network for each generation in cursive dataset

5.1 Bayesian Networks

5.1.1 Structure learning

Figure 2 and 3 show the Bayesian Network for different generations in cursive and handprint dataset.

Figure 4 and 5 show the Bayesian Network for entire cursive and handprint dataset.

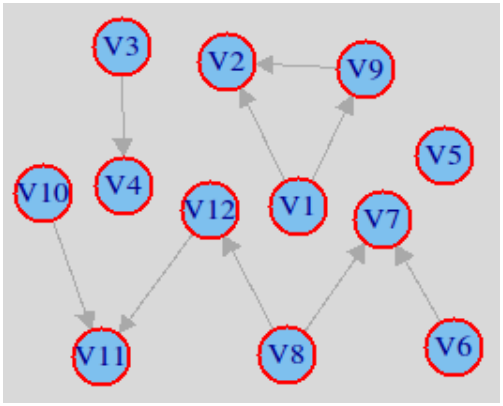
5.1.2 Comparison between ancestral sampling and Gibbs sampling

In order to compare the performance of Gibbs sampling and ancestral sampling for Bayesian Network, we compare the following values:

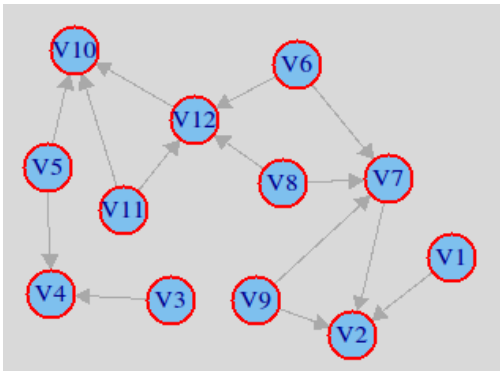
- $L2$ norm of difference between true mean of Bayesian Network and mean of 10000 samples drawing from Bayesian Network using Gibbs sampling,
- $L2$ norm of difference between true mean of Bayesian Network and mean of 10000 samples drawing from Bayesian Network using ancestral sampling.

Figure 6 shows the comparison of Gibbs sampling and ancestral sampling for Bayesian Network in different datasets.

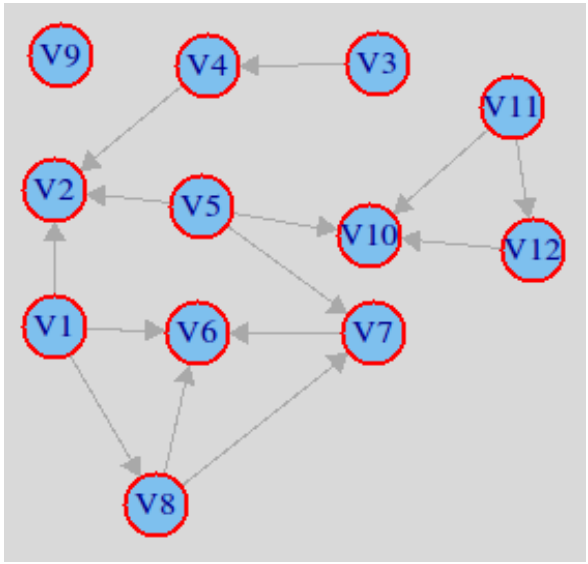
In figure 6, we denote *Difference* as the $L2$ norm of difference between true mean of Bayesian



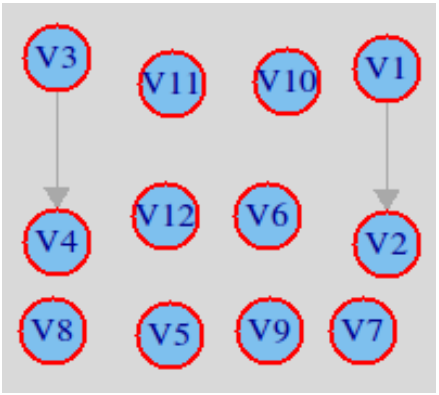
(a) Generation 1



(b) Generation 2



(c) Generation 3



(d) Generation 4

Figure 3: Bayesian Network for each generation in handprint dataset

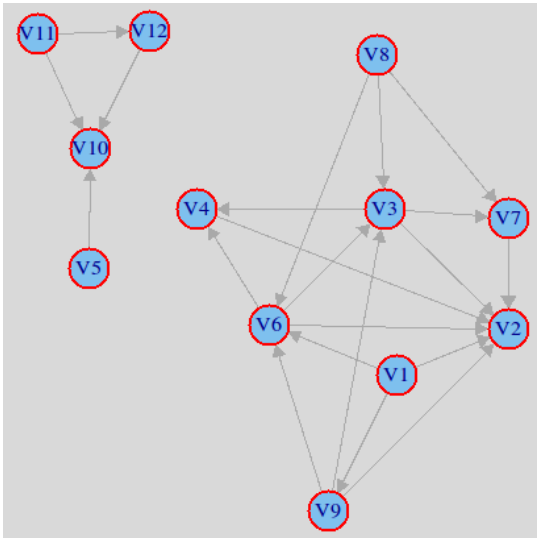


Figure 4: Bayesian Network for the entire cursive dataset

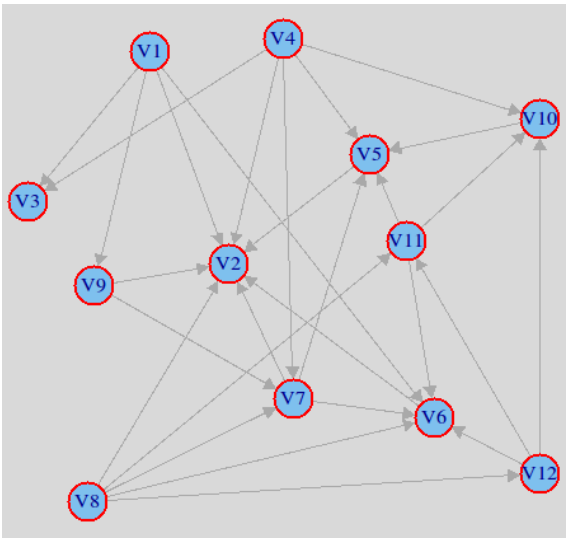


Figure 5: Bayesian Network for the entire handprint dataset

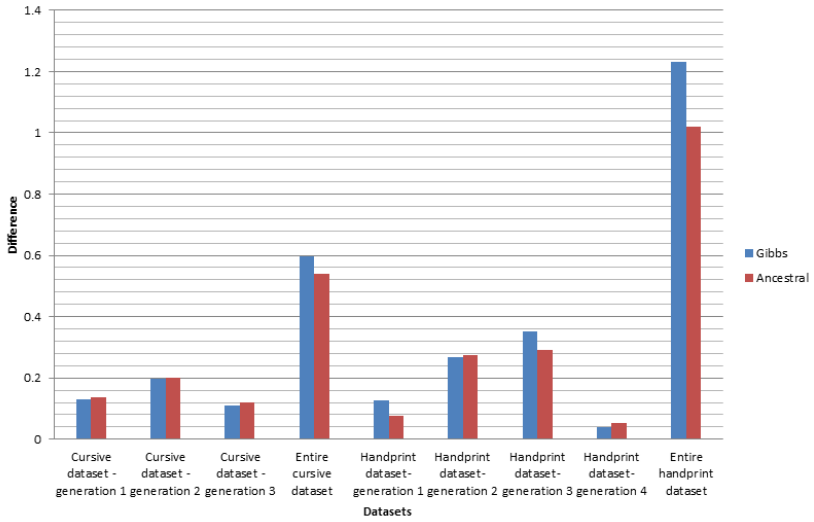
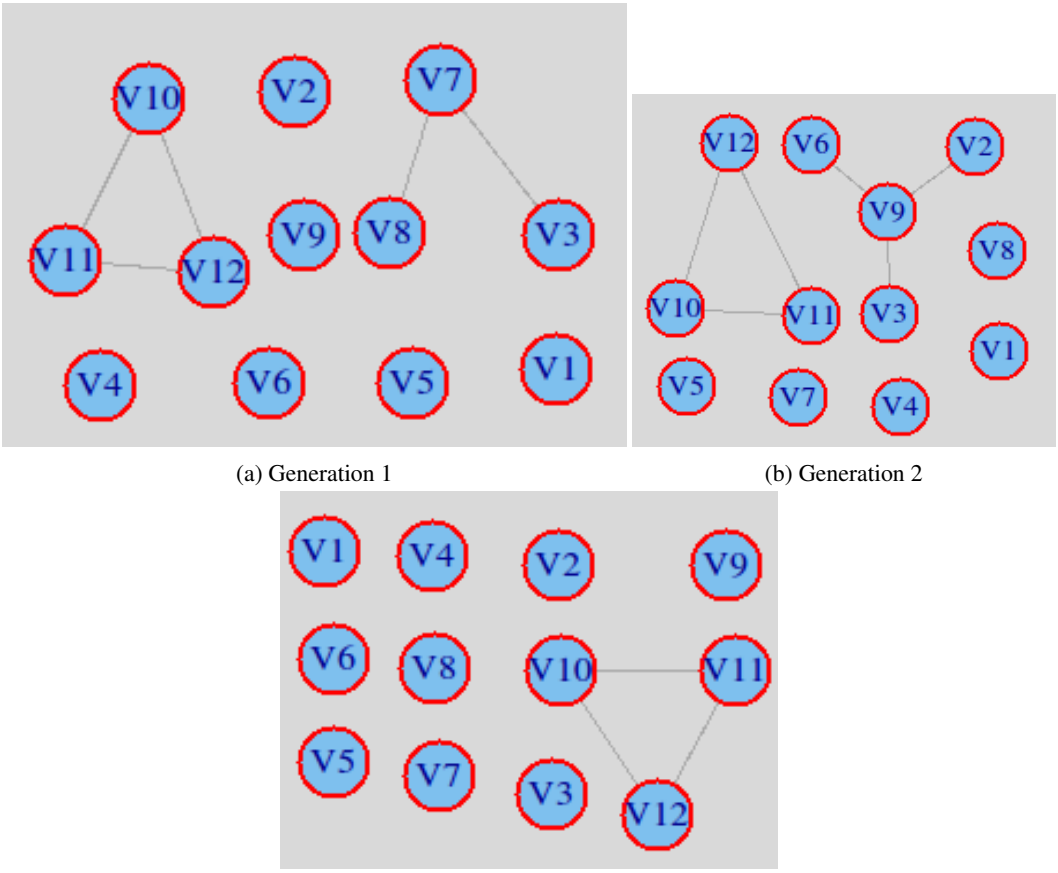


Figure 6: Comparison between Gibbs sampling and ancestral sampling



(a) Generation 1

(b) Generation 2

(c) Generation 3

Figure 7: Markov Network for each generation in cursive dataset

Network and mean of 10000 samples drawing from Bayesian Network using respective sampling method. From figure 6, we observe that Gibbs sampling and ancestral sampling have comparable performance in children handwriting dataset.

5.2 Markov Network

5.2.1 Structure learning

With the approach presented in earlier section, from the given dataset, we can construct the Markov Network and learn its parameters using gradient descent.

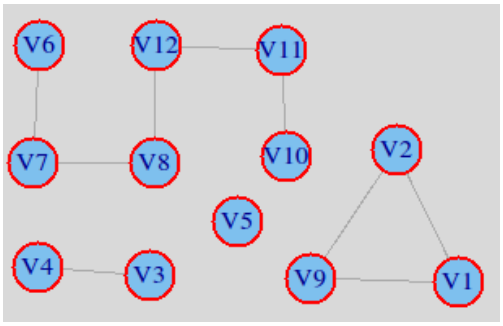
Figure 9 and 10 show the Markov Network for the entire cursive and handprint dataset.

Figure 7 and 8 show the Markov Network for each generation in each dataset (cursive and handprint).

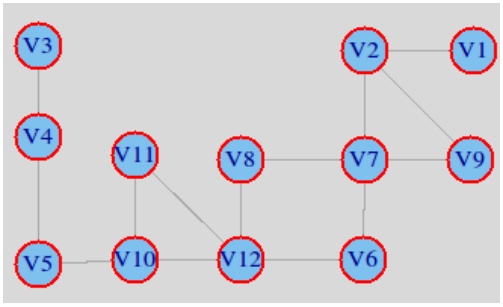
5.3 Comparison between Markov Network and Bayesian Network

5.3.1 Mean of distribution

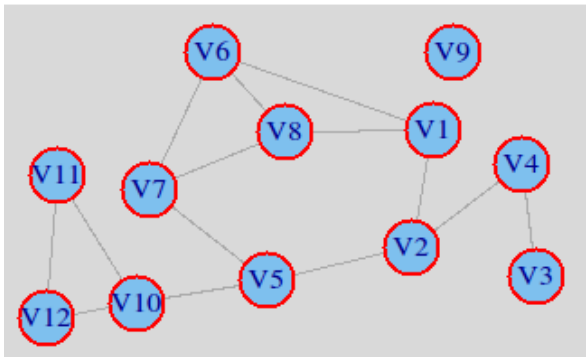
For the mean of the distribution, we compute the approximate mean using formula given in section 3.5.2. We first compute the mean from data set. Then we compare them with the mean of samples drawing from Bayesian Network using ancestral sampling (10000 samples) and Gibbs sampling (10000 samples). We also compute the mean of samples drawing from Markov Network using Gibbs sampling (10000 samples).



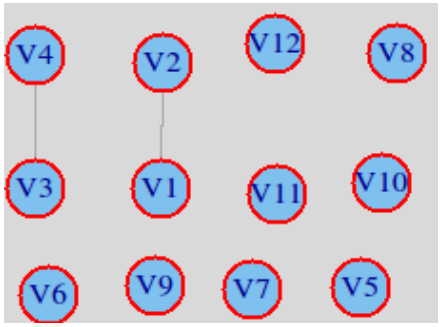
(a) Generation 1



(b) Generation 2



(c) Generation 3



(d) Generation 4

Figure 8: Markov Network for each generation in handprint dataset

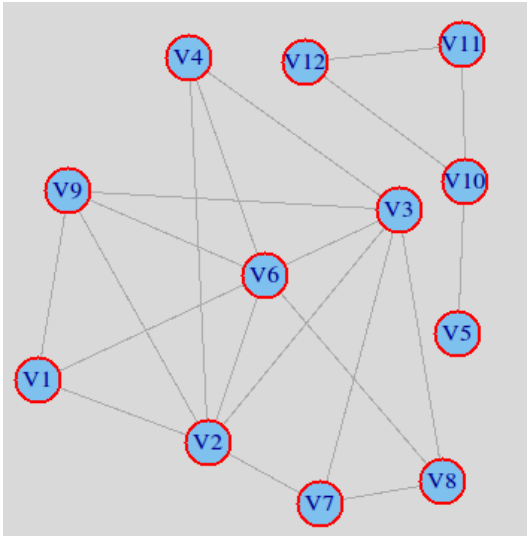


Figure 9: Markov Network for the entire cursive dataset

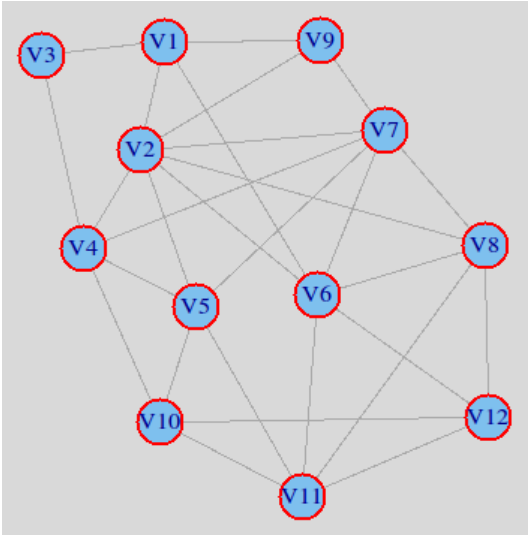


Figure 10: Markov Network for the entire handprint dataset

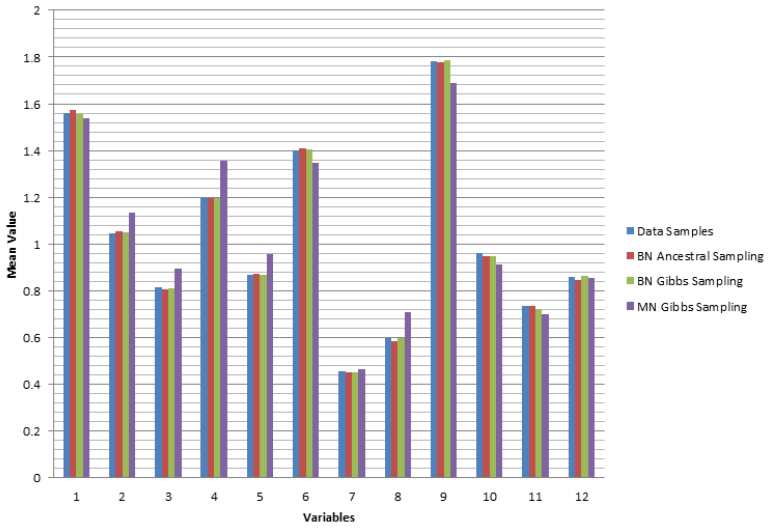


Figure 11: Mean of distribution of Bayesian Network obtained from cursive dataset - generation 1

Figure 11, 12 and 13 show the mean value for each random variables and give comparison between different mean values obtained from data set and samples drawing from both Bayesian Network and Markov Network in cursive dataset.

Figure 15, 16, 17 and 18 show the mean value for each random variables and give comparison between different mean values obtained from data set and samples drawing from both Bayesian Network and Markov Network in handprint dataset.

Figure 14 and 19 show the mean value for each random variables and give comparison between different mean values obtained from data set and samples drawing from both Bayesian Network and Markov Network in the entire dataset of cursive data and handprint one.

5.3.2 Entropy of a distribution

For the entropy of the distribution, we compute the approximate entropy using formula given in section 3.5.3. Figure 20 shows the entropy value for each probability distribution learnt from each dataset using Bayesian Network and Markov Network.

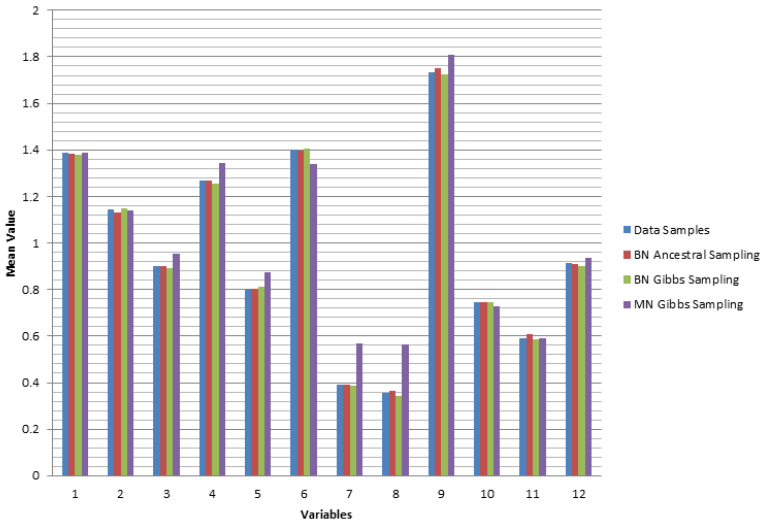


Figure 12: Mean of distribution of Bayesian Network obtained from cursive dataset - generation 2

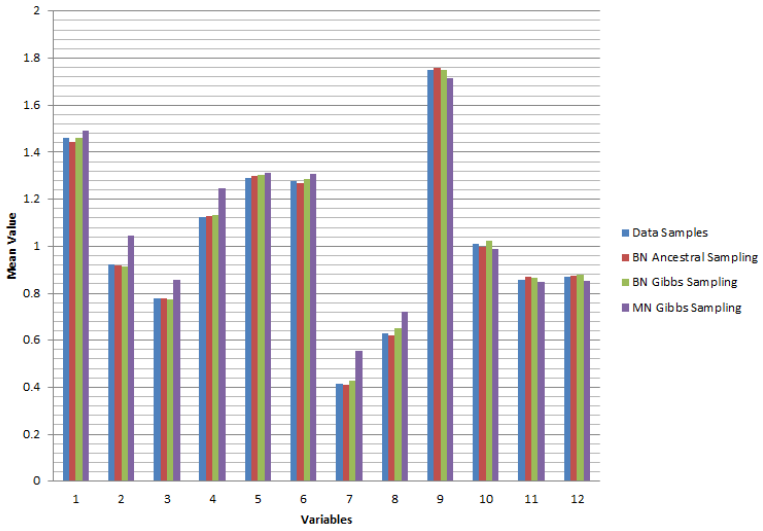


Figure 13: Mean of distribution of Bayesian Network obtained from cursive dataset - generation 3

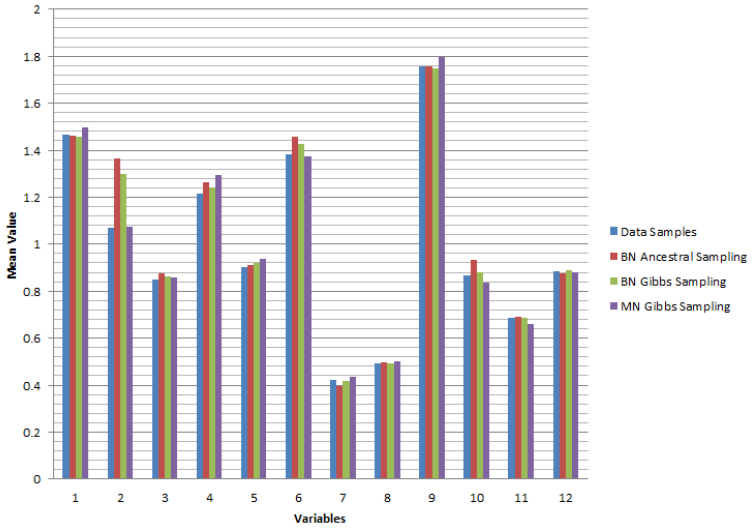


Figure 14: Mean of distribution of Bayesian Network obtained from entire cursive dataset

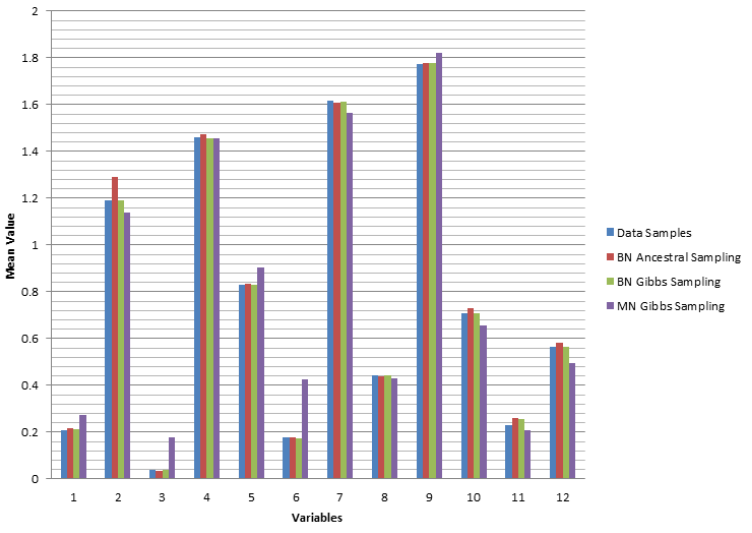


Figure 15: Mean of distribution of Bayesian Network obtained from handprint dataset - generation 1

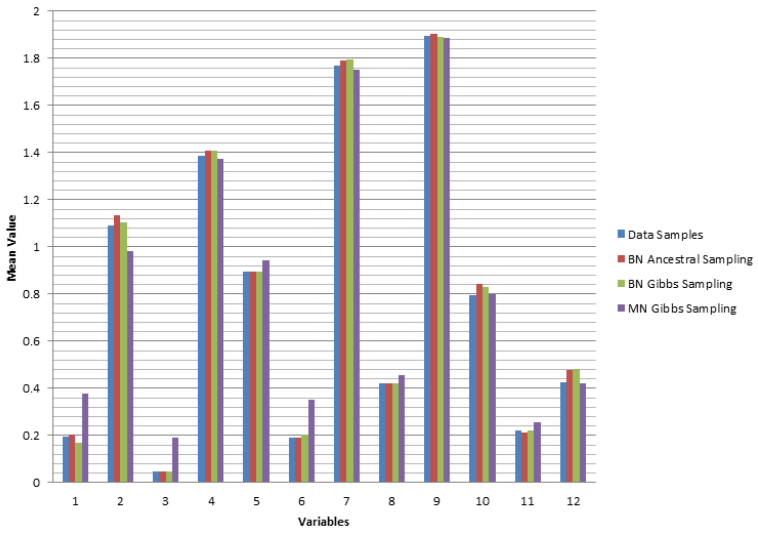


Figure 16: Mean of distribution of Bayesian Network obtained from handprint dataset - generation 2

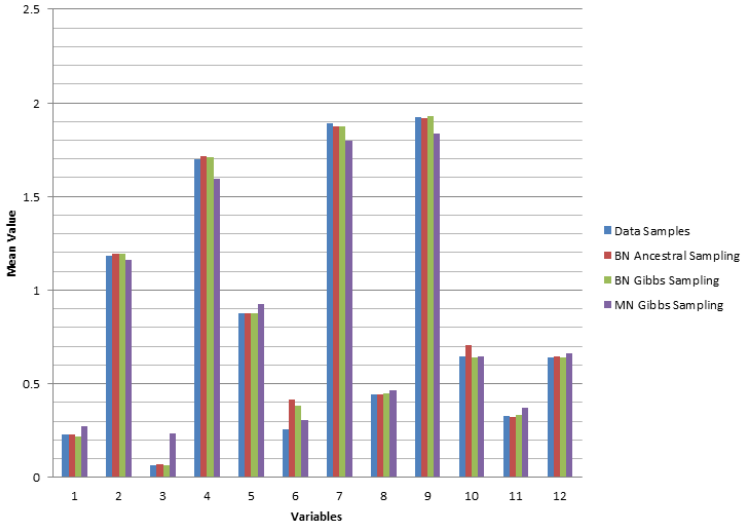


Figure 17: Mean of distribution of Bayesian Network obtained from handprint dataset - generation 3

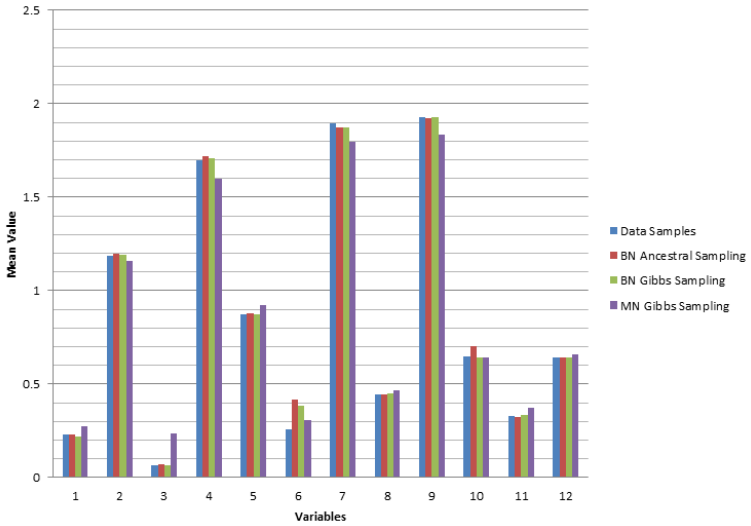


Figure 18: Mean of distribution of Bayesian Network obtained from handprint dataset - generation 4

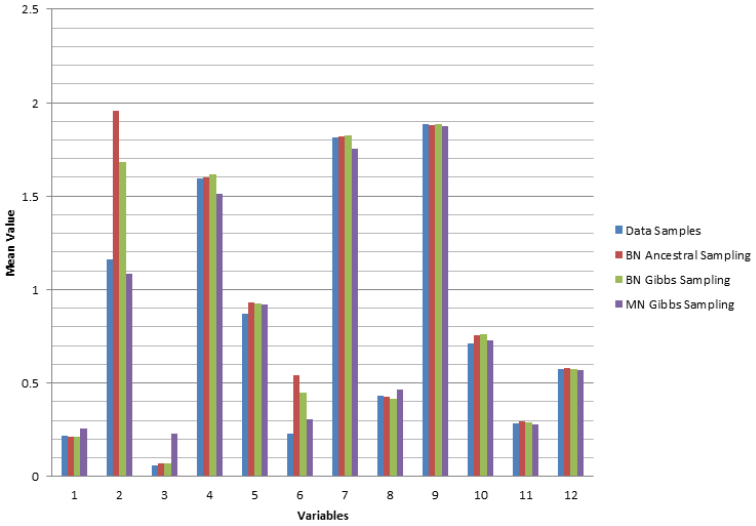


Figure 19: Mean of distribution of Bayesian Network obtained from entire handprint dataset

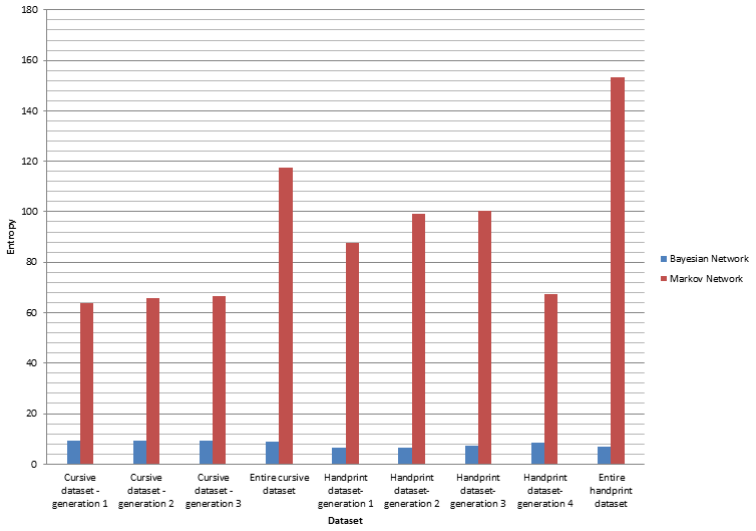


Figure 20: Comparison of entropy of distribution obtained from Bayesian Network and Markov Network

5.4 Rare writing and common writing

Most common writing in the dataset is defined as a data sample that has highest probability using any probability distributions inferred from dataset. Similarly, rarest writing in the dataset is defined as a data sample that has lowest probability using any probability distributions inferred from dataset.

In our experiment, we determine the most common handwriting and rarest handwriting for cursive and handprint dataset using Bayesian Network and Markov Network.

Table 1 shows the most common and least common data sample identified by using Bayesian Network and Markov Network. From the table 1, we can observe that Bayesian Network and Markov Network agree on the most common handwriting. In identifying rarest handwriting, Bayesian Network and Markov Network agrees on the rarest handwriting in handprint dataset while disagree in cursive dataset.

Table 1: Most common and least common data samples identified by using Bayesian Network (BN) and Markov Network (MN)

	Rarest handwriting sample		Most common writing sample	
Dataset	BN	MN	BN	MN
cursive dataset	KAM2021MO	GaT2021Kl	SiP2020Fl	SiP2020Fl
handprint dataset	LAE2020JO	LAE2020JO	IsR2020Ch	IsR2020Ch

From above experiment, we can say there is a high correlation between results obtained from Bayesian Network and Markov Network, which indicates they all represent the same distribution in handwriting dataset.

Figure 21, 22, 23, 24 and 25 shows images of KAM2021MO[3] (rarest handwriting in cursive dataset identified by BN), GaT2021Kl (rarest handwriting in cursive dataset identified by MN), LAE2020JO (rarest handwriting in handprint dataset identified by both BN and MN), SiP2020Fl (most common handwriting in cursive dataset identified by both BN and MN), IsR2020Ch (most common handwriting in handprint dataset identified by both BN and MN).

[3] we use student's identifier in dataset to indicate handwriting written by him/her

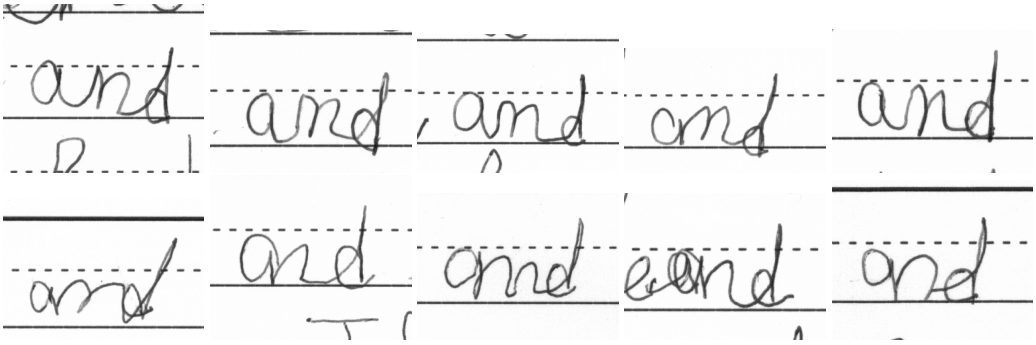


Figure 21: Rarest handwriting samples in cursive dataset identified by BN

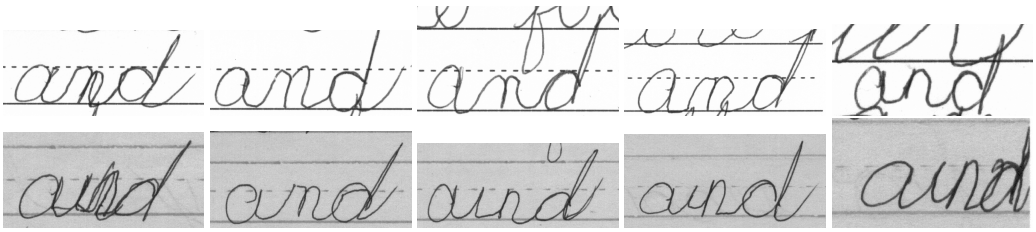


Figure 22: Rarest handwriting samples in cursive dataset identified by MN

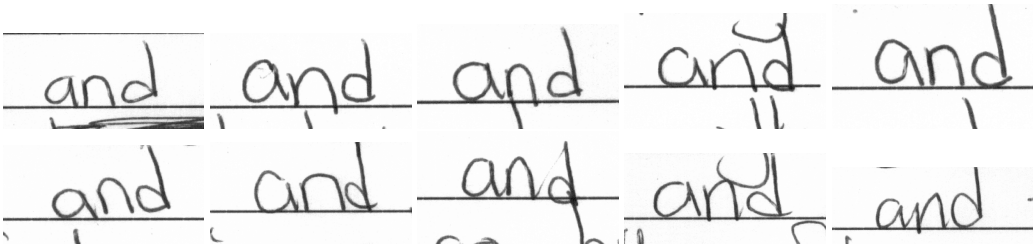


Figure 23: Rarest handwriting samples in handprint dataset identified by both BN and MN

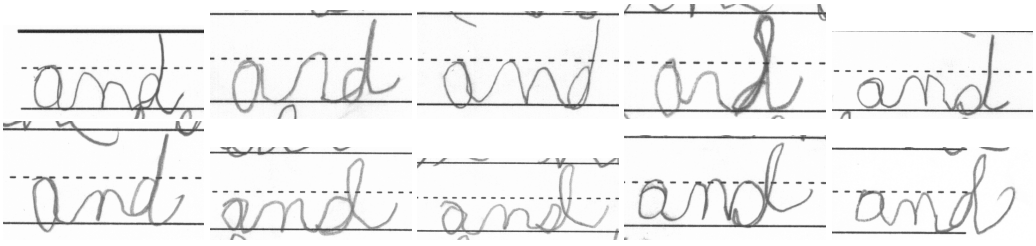


Figure 24: Most common handwriting samples in cursive dataset identified by both BN and MN

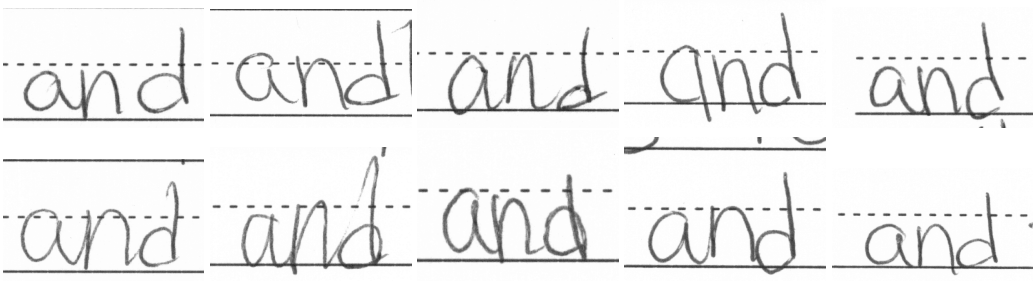


Figure 25: Most common handwriting samples in handprint dataset identified by both BN and MN

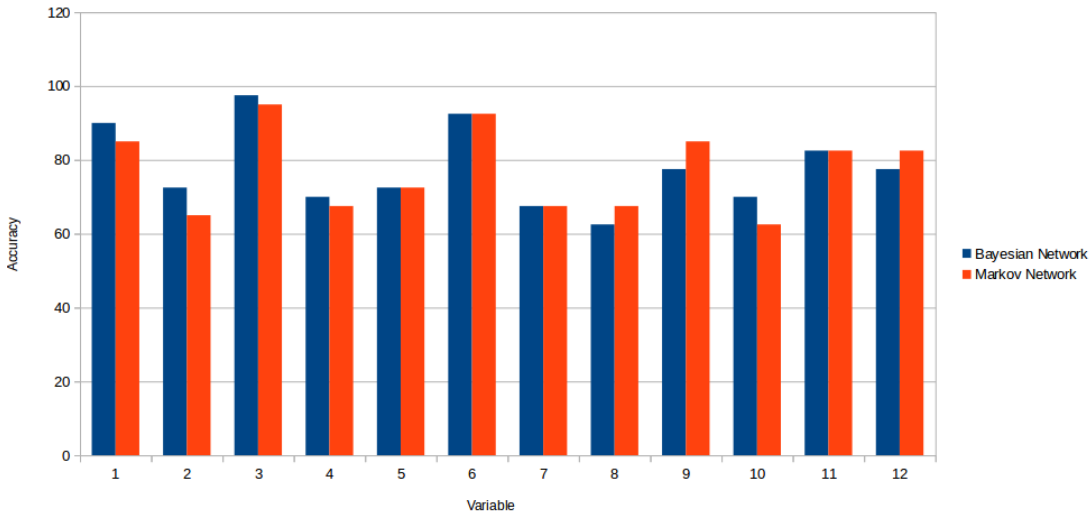


Figure 26: Accuracy of missing value imputation using Bayesian Network and Markov Network

5.5 Missing value imputation

One application of Bayesian Network and Markov Network in children handwriting dataset is missing value imputation. In particular, many samples in the dataset usually have missing values which can be inferred using probabilistic approach. In brief, we compute the all possible assignments for the missing value in that tuple and assign the most probable assignment (has highest probability) to the missing value.
In order to verify this approach, we design an experiment as follow:

- Consider the entire handprint dataset (we only consider data samples without missing values).
- Divide the dataset into training set and testing set, in which testing set contains 40 random samples and training set contains the remaining samples.
- Learn the Bayesian Network and its parameters using training set (similarly for Markov Network).
- In testing set, for each variable, we first assign missing value to that variable, and use the probability distribution learnt from Bayesian Network (and Markov Network respectively) to predict the missing value.
- We then compare the results obtained from prediction with the ground truth (values originally assigned to that variable). With this approach, we can compute the accuracy of our approach.

Figure 26 shows the accuracy of missing value imputation using Bayesian Network and Markov Network.

From figure 26, we can observe that the accuracy of missing value imputation using Bayesian Network and Markov Network is similar. This is a reinforcement for our claim that Bayesian Network and Markov Network are all good approximations for the same original distribution of the dataset.

6 Conclusion

In this project, we have applied the Bayesian Network and Markov Network learning into the dataset of children handwriting. In particular, we constructed the structure of Bayesian Network and Markov Network from given dataset. We also use Maximum Likelihood approach to find parameters for our Bayesian Network and Markov Network. With the learned PGMs, we perform different inferences

and answer many important questions involving children handwriting such as identifying rare writing as well as common writing. We also use the learned PGMs to solve the problem of missing value imputation.

References

[1] Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

[2] Pearson, K. (1900). *On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 50(302), 157-175.*