Determining Probabilities of Handwriting Formations using PGMs

Shraddha Dwivedi
Computer Science and Engineering
University at Buffalo
sdwivedi@buffalo.edu

Abstract

This project describes the approach to develop probabilistic graphical models (PGMs) to determine probabilities of observations which are described by several variables. In this project we have used and worked with handwriting patterns which are described by document examiners and later on analyzed the result obtained. The probabilities obtained using Bayesian Networks are used to determine whether a sample was written by a certain individual or not. We can also determine that whether a particular sample is common or rare.

1 Introduction

In this project, handwriting data sets are analyzed using Bayesian networks and the obtained results are used to find the most common and rare handwriting sample as well as determine whether a sample was written by a certain individual or not. For performing the above mentioned task, we have used two data-sets 'th' data set and 'and' data set. The 'th' data-set contains six feature vectors and the 'and' data-set contains nine feature vectors. Various tasks performed to reach the goal are as follows:

1. Calculated the pairwise correlations and dependencies that exist in the data between feature variables using the conditional probability provided for 'th' data-set. This task has been performed on 'th' data sets. Calculated all the dependencies using the provided 'th' data set consisting of 6 tables.
2. Constructed several Bayesian Networks using the results obtained from first step. A data-set is generated using Bayesian sampling. Then calculated the K2score for each of the Bayesian network.Based on the K2 score, selected the best model and afterwards, based on the best model, calculated and described the high probability and low probability of 'th'.
3. In this task, converted the Bayesian Network into a Markov Network using moralization.
4. In this task, 'AND' image data set is used and then constructed the Bayesian Network and Markov Network using this 'AND' image data set.Here, I have generated the Conditional Probability using the PGMPY library.

2 Task 1: Calculation of Pairwise correlations and independence

In this first task, we have calculated the pairwise dependencies between the each of two features of the given 'th' data sets. For calculating the dependencies, we have used the following formula:

$$\sum_{i} \sum_{j} \mid p(xi, xj) - p(xi)p(xj) \mid \tag{1}$$

We have used this formula to calculate the pairwise relation. Output has been represented in Table 1. In certain field in table, values are not present. By using this information we can are assuming

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

Table 1: Conditional Dependencies

Features	X1	X2	X3	X4	X5	X6
X1	-	0	0	0.1195	0	0.1603
X2	0.159	-	0.2187	0.1157	0.1293	0.1753
X3	0	0.2185	-	0	0.1159	0.0943
X4	0.1194	0	0	-	0	0.1430
X5	0	0.1324	0.1155	0	-	0
X6	0.1601	0	0.0965	0.1434	0	-

that those two variables are independent from each other. In the table, rows and columns values represents the dependency of a variable on each other. From Table 1, we can infer that there are strong dependencies between X2 -> X5, X2 -> X3 and weak dependencies between X6 -> X3, X3 -> X6.

3 Task 2: Construction of Bayesian Network

In Task 2, we have used the dependencies calculated in the previous Task. Using that dependencies values, we have created the five different Bayesian Network. We have used the PGMPY Library to create Bayesian network. The below described 5 different Bayesian models are constructed using checking the dependencies presented in the Table 1 and checking which one variable is dependent to which another variable having lesser value of conditional probability. We have tried to select the variables having less conditional probability, so that we can get the near good Bayesian Network. Following are the five different Bayesian models:

```
from pgmpy.models import BayesianModel
model1 = BayesianModel([('x3', 'x6'), ('x3', 'x5'),
                        ('x6', 'x2'), ('x6', 'x1'), ('x6', 'x4')])
model2 = BayesianModel([('x6', 'x1'), ('x6', 'x2'),
                        ('x2', 'x3'), ('x2', 'x5'), ('x1', 'x4')])
model3 = BayesianModel([('x6', 'x1'), ('x6', 'x2'),
                        ('x6', 'x3'), ('x2', 'x5'), ('x1', 'x4')])
model4 = BayesianModel([('x4', 'x1'), ('x4', 'x2'),
                        ('x2', 'x3'), ('x3', 'x5'), ('x1', 'x6')])
model5 = BayesianModel([('x1', 'x6'), ('x1', 'x4'),
                        ('x6', 'x2'), ('x6', 'x3'), ('x3', 'x5')])
```

Here ('x3', 'x6') implies that x6 is dependent on x3 i.e x3 is parent of x6. In the below mentioned code, Conditional Probabilities which we have calculated in Task 1 has been used here as parameters in each of the five different created Bayesian Network. In the below code snippet, evidence describes the number of parameter that parent variable can take. 'values' tells the conditional probability of the variable which has been listed in Table 1.

```
from pgmpy.factors.discrete import TabularCPD
cpd_x4 =TabularCPD(variable='x4', variable_card=4,
values=[[x / 100 for x in p[3]]])
cpd_x1 =TabularCPD(variable='x1', variable_card=4,
values=[[float(j)/100 for j in i] for i in px14],
evidence=['x4'],evidence_card=[4])
...
cpd_x6= TabularCPD(variable='x6', variable_card=5,
values=[[float(j)/100 for j in i] for i in px61],
evidence=['x1'],evidence_card=[4])

model4.add_cpds(cpd_x3, cpd_x6, cpd_x5, cpd_x2, cpd_x4, cpd_x1)

model4.check_model()
```

Table 2: K2 Scores Values

Score	Model1	Model2	Model3	Model4	Model5
K2 Scores	-6444.3877	-6436.8686	-6510.8176	-6371.8956	-6449.9019

In the below mentioned code snippet we have done the sampling to generate the random data sets. We have used PGMPY library to do this task. The data generated in the below code is of size 1000. In the next defined code snippet, we have calculated the the K2score, this K2 score is used to find the best model. We have directly used the pre-defined 'K2Score' function to calculate the K2 score. Higher the K2 score value, better the Bayesian Network model.

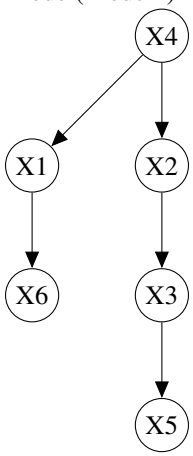
```
from pgmpy.sampling import BayesianModelSampling
inference = BayesianModelSampling(model1)
bayesianData1=inference.forward_sample(size=1000)

from pgmpy.estimators import K2Score
k1 = K2Score(bayesianData1)
print('K2 Score of Model 1: ' + str(k1.score(model1)))
```

By looking at values of Table 2, we can conclude that Model4 is the best Bayesian Network Model for the given 5 different Bayesian Model. The formula to calculate the joint probability distribution for the Model4, which is selected as best model by looking into the K2Score value, is given below.

$$p(x1, x2, x3, x4, x5, x6) = p(x4)p(x1|x4)p(x2|x4)p(x3|x2)p(x5|x2)p(x5|x2) \quad (2)$$

In this formula x4 is independent variable.

Figure 1: Best Model(Model4) for 'th' data-set



This below mentioned table(Table 3) is presenting the values of most probable 'th' pattern. This

Table 3: High Probability 'th' pattern

Features	Parameters
X1: Height Relationship of t to h	1: t shorter than h
X2: Shape of Loop of h	3: both sides curved
X3: Shape of Arch of h	2: pointed
X4: Height of Cross on t staff	1: upper half of staff
X5: Baseline of h	4: no set pattern
X6: Shape of t	4: closed

below mentioned table(Table 4) is presenting the values of rare probable 'th' pattern.

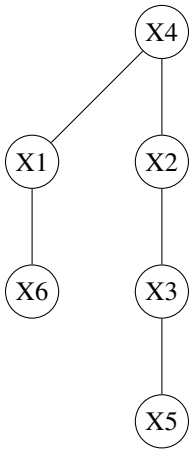
Table 4: Low probability 'th' pattern

Features	Parameters
X1: Height Relationship of t to h	1: t shorter than h
X2: Shape of Loop of h	1: retraced
X3: Shape of Arch of	1: rounded arch
X4: Height of Cross on t staff	3: above staff
X5: Baseline of h	1: slant- ing upward
X6: Shape of t	1: tented

4 Task 3: Converting the best obtained Bayesian Network into Markov Network

In this Task, we have constructed the Markov Network from the best obtained Bayesian Network i.e Model4 for us using the moralization. In this project for 'th' data set we have created the Bayesian Network of only one parent, so when we convert out best Bayesian model i.e Model4 into Markov Network, we observe that the constructed Markov Network will be same as the Bayesian Network. In

Figure 2: Best Model obtained for 'th' data-set



the below mentioned code snippet, we have calculated the the edges and factors from the Markov Network which is constructed from the best Bayesian Model i.e Model4.

```
markovModel_th = model4.to_markov_model()
print(markovModel_th.edges())
print(markovModel_th.get_factors())
```

```
»[[('x4', 'x1'), ('x4', 'x2'), ('x1', 'x6'), ('x2', 'x3'), ('x3', 'x5')]
<DiscreteFactor representing phi(x1:4) at 0x7fc37ef78b00>,
<DiscreteFactor representing phi(x6:5, x1:4) at 0x7fc37ef78048>,
<DiscreteFactor representing phi(x4:4, x6:5) at 0x7fc37f315160>,
<DiscreteFactor representing phi(x2:5, x6:5) at 0x7fc37f315e10>,
<DiscreteFactor representing phi(x3:3, x2:5) at 0x7fc37f315470>,
<DiscreteFactor representing phi(x5:4, x2:5) at 0x7fc37f315fd0>]
```

5 Task 4: Constructing the Bayesian Network and evaluating the goodness score for 'and' image data-set

In this task, we have used the "and" image data-set to construct a Bayesian network and evaluate the goodness score (likelihood of a data-set) of several Bayesian networks. For the 'and' image

Figure 3: Markov Model for 'and' image data-set

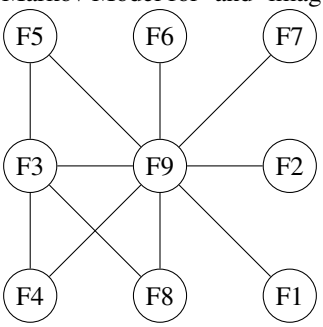


Table 5: Most Common 'and' pattern

Features	Parameters
F1: Initial stroke of formation of a	3: Center of Staff
F2: Formation of staff of a	2: Retraced
F3: Number of arches of n	2: Two
F4: Shape of arches of n	1: pointed
F5: Location of mid-point of n	3: At baseline
F6: Formation of staff of d	3: Looped
F7: Formation of initial stroke of d	1: Overhand
F8: Formation of terminal stroke of d	4: No obvious ending stroke
F9: Symbol in place of the word and	2: Symbol

data-set, we have given a data sets. So we have directly used the greedy hill climb search to get the best Bayesian Network. The below code snippet is showing the usage of HillClimbSearch to evaluate the best Bayesian Network model for 'and' image data sets.

```
from pgmpy.estimators import HillClimbSearch
from pgmpy.estimators import K2Score
hillclimb = HillClimbSearch(andFeatures_data,
scoring_method=K2Score(andFeatures_data))
bestModel = hillclimb.estimate()
oneParent=hillclimb.estimate(max_indegree=1)
print(bestModel.edges())
print(oneParent.edges())
```

»[('f3', 'f4'), ('f3', 'f9'), ('f3', 'f8'), ('f5', 'f9'), ('f5', 'f3'), ('f9', 'f8'), ('f9', 'f7'), ('f9', 'f1'), ('f9', 'f6'), ('f9', 'f2'), ('f9', 'f4')]
»[('f3', 'f4'), ('f5', 'f9'), ('f5', 'f3'), ('f9', 'f8'), ('f9', 'f7'), ('f9', 'f1'), ('f9', 'f6'), ('f9', 'f2')]

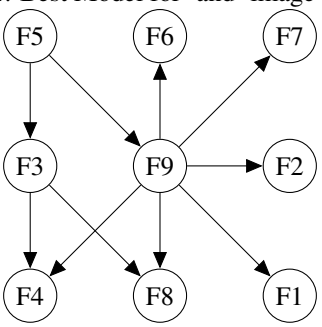
Now we have calculated the K2 score to determine the best Bayesian Model. This process/step

Table 6: Most Rare 'and' pattern

Features	Parameters
F1: Initial stroke of formation of a	4: No fixed pattern
F2: Formation of staff of a	4: No staff
F3: Number of arches of n	1: One
F4: Shape of arches of n	5: No fixed pattern
F5: Location of mid-point of n	2: below baseline
F6: Formation of staff of d	1: tented
F7: Formation of initial stroke of d	4: No fixed pattern
F8: Formation of terminal stroke of d	4: No obvious ending stroke
F9: Symbol in place of the word and	1: Formation

is same as the previous mentioned code snippet used to calculate the K2Score for the 5 different

Figure 4: Best Model for 'and' image data-set



Bayesian Model for 'th' data sets. This below mentioned code snippet is used to calculate the conditional probabilities for the 'and' image data sets

```
from pgmpy.estimators import BayesianEstimator
estimator = BayesianEstimator(bestModel, andFeatures_data)
for node in bestModel.nodes():
    cpd = estimator.estimate_cpd(node)
    print(cpd)
    cpdofAnd.append(cpd.get_values())
```

Once, we have the conditional probabilities for each node, we find the most common and most rare occurring 'and' pattern. This is shown in Table-5 and Table-6 respectively.

6 Conclusion

In this project, Probabilistic Models were constructed for the 'th' data-set and 'and' data-set. Several probabilistic graphical model (PGM) were constructed so that we can calculate the joint probability distribution of the six features of 'th' and nine features of 'and'. This process of evaluation is called the process of inference. From the 'and' data set, we generated conditional probabilities and obtained the best model possible as shown in Figure 4. The model was used to report the most probable and least probable pattern of both 'th' and 'and' in Table-3,4,5 and, 6 respectively. This project will be useful for the next project regarding handwriting comparison using explainable.

References

[1] PGMPY Documentation: http://pgmpy.org/index.html

[2] Determining Probabilities of Handwriting Formations using PGMs - Project Description