CSE674 Project 1: Determining Probabilities of Handwriting Formations using PGMs

Nikhil Srihari
UB Person ID : **50291966**
Department of Computer Science,
SUNY – UB, Buffalo, NY
nikhilsr@buffalo.edu

Abstract

In this report, I document the output and analysis of Determining Probabilities of Handwriting Formations using PGMs for 'th' and 'and' dataset. This project contains 4 tasks. Let us discuss them by one by one.

1 Introduction

The purpose of this project is to document the output and analysis of Determining Probabilities of Handwriting Formations using PGMs for 'th' and 'and' dataset. Task 1 consists of finding which features are independent of which features for 'th' dataset. Task 2 involves creating different Bayesian networks and finding the one with the best K2 score for 'th' dataset. Task 4 consists of the same task as Task 2, but with 'and' dataset. Finally, Task 3 consists of converting the above found, best Bayesian networks to Markov Models and comparing their results. Let us begin our discussion with Task 1

2 Task 1

The 'th' images can be describes by 6 manually recognized features. We are given the prior probabilities of all these 6 features, for all their possible values. Apart the prier distribution, we have also been provided with some 2 feature Condition Probability Distributions(CPDs). Ideally we would need all the possible CPDs to completely create the best Bayesian Network. However this leaves us with around 400k parameters. Thus, we concentrate here on only a few CPDs. The CPDs that aren't mentioned here, are assumed to be independent. The first task, requires us to find whether the feature pair (xi,xj) are independent for all values of i and j. This can be done by using the calculating the below quantity for multiclass features, and considering all (xi, xj) pairs with this value over some threshold to be non-independent. Else they are independent:

$$S = \sum abs(P(xi,xj)-P(xi)P(xj))$$

There are also other approximations of this quantity available.

2.1 S values for all (xi,xj) pairs:

This is expressed as a 6x6 matrix.

```
[[0.       0.15977  0.       0.11943  0.       0.160155]
 [0.15977  0.       0.218525 0.1157   0.13246  0.175315]
 [0.       0.218525 0.       0.       0.11552  0.11324 ]
 [0.11943  0.1157   0.       0.       0.       0.14347 ]
 [0.       0.13246  0.11552  0.       0.       0.      ]
 [0.160155 0.175315 0.11324  0.14347  0.       0.      ]]
```

2.2 Thresholds and Independent pairs

Here are the different threshold values I considered and the corresponding independence matrix. If the value = 1, the pair are independent. Else they are dependent.

```
[{'threshold': 0.11, 'independence': array([[1., 0., 1., 0., 1., 0.],
       [0., 1., 0., 0., 0., 0.],
       [1., 0., 1., 1., 0., 0.],
       [0., 0., 1., 1., 1., 0.],
       [1., 0., 0., 1., 1., 1.],
       [0., 0., 0., 0., 1., 1.]])}, {'threshold': 0.12, 'independence': array([[1., 0., 1., 1., 1., 0.],
       [0., 1., 0., 1., 0., 0.],
       [1., 0., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 0.],
       [1., 0., 1., 1., 1., 1.],
       [0., 0., 1., 0., 1., 1.]])}, {'threshold': 0.13, 'independence': array([[1., 0., 1., 1., 1., 0.],
       [0., 1., 0., 1., 0., 0.],
       [1., 0., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 0.],
       [1., 0., 1., 1., 1., 1.],
       [0., 0., 1., 0., 1., 1.]])}, {'threshold': 0.14, 'independence': array([[1., 0., 1., 1., 1., 0.],
       [0., 1., 0., 1., 1., 0.],
       [1., 0., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 0.],
       [1., 1., 1., 1., 1., 1.],
       [0., 0., 1., 0., 1., 1.]])}, {'threshold': 0.15, 'independence': array([[1., 0., 1., 1., 1., 0.],
       [0., 1., 0., 1., 1., 0.],
       [1., 0., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.],
       [0., 0., 1., 1., 1., 1.]])}, {'threshold': 0.16, 'independence': array([[1., 1., 1., 1., 1., 0.],
       [1., 1., 0., 1., 1., 0.],
       [1., 0., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.],
       [0., 0., 1., 1., 1., 1.]])}, {'threshold': 0.17, 'independence': array([[1., 1., 1., 1., 1., 1.],
       [1., 1., 0., 1., 1., 0.],
       [1., 0., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.],
       [1., 0., 1., 1., 1., 1.]])}]
```

3 Task 2

In this task we try to create Bayesian Network Models using the S values calculated and the independent pairs discovered in Task 1. For the sake of simplicity, it is assumed that any node can have only one parent, as we have only CPDs for 2 feature pairs. Thus we manually create different BNs and assign them their CPDs. We then sample a 1000 samples from this model and calculate the K2 score for this model. The model with the highest K2 score is best performing.

```
Task 2 - For 'th' data:
WARNING:root:Replacing existing CPD for x1
WARNING:root:Replacing existing CPD for x6
WARNING:root:Replacing existing CPD for x2
WARNING:root:Replacing existing CPD for x3
WARNING:root:Replacing existing CPD for x5
    Bayesian Model 1 K2 Accuracy Score is -6470.107263584804
WARNING:root:Replacing existing CPD for x4
WARNING:root:Replacing existing CPD for x6
WARNING:root:Replacing existing CPD for x2
WARNING:root:Replacing existing CPD for x3
WARNING:root:Replacing existing CPD for x5
    Bayesian Model 2 K2 Accuracy Score is -6422.68423237832
WARNING:root:Replacing existing CPD for x1
WARNING:root:Replacing existing CPD for x6
WARNING:root:Replacing existing CPD for x3
WARNING:root:Replacing existing CPD for x2
WARNING:root:Replacing existing CPD for x5
    Bayesian Model 3 K2 Accuracy Score is -6383.527082095525
WARNING:root:Replacing existing CPD for x3
WARNING:root:Replacing existing CPD for x2
WARNING:root:Replacing existing CPD for x1
WARNING:root:Replacing existing CPD for x6
WARNING:root:Replacing existing CPD for x4
    Bayesian Model 4 K2 Accuracy Score is -6442.7889970661045
WARNING:root:Replacing existing CPD for x3
WARNING:root:Replacing existing CPD for x2
WARNING:root:Replacing existing CPD for x1
WARNING:root:Replacing existing CPD for x4
WARNING:root:Replacing existing CPD for x6
    Bayesian Model 5 K2 Accuracy Score is -6409.137922321888
    Best Bayesian Model with the highest accuracy score is thus Model 3
```

The network architecture of the best performing model from above can be simply described by the edge pairs:

```
[('x4','x1'), ('x1','x6'), ('x6','x3'), ('x3','x2'), ('x2','x5')]
```

4 Task 4

In this task we try to create Bayesian Network Models for the ‘and’ data. This is basically Task 2 but for ‘and’ data. The difference here being that with ‘and’ dataset doesn’t provide any probability values. It’s just a list of the the different features for different images. Hence, we try to find the best Bayesian structure first by using the Hill Climb algorithm, Then we use Bayesian estimators on this data with this model in mind, to calculate all the necessary CPDs. We repeat this procedure for multiple models, where these models are slight variation of the local best model produced by Hill Climb algorithm.

```
Task 4 - For 'and' data:

    Model 1: Model through HillClimbSearch is : [('f3', 'f4'), ('f3', 'f9'), ('f3', 'f8'), ('f5', 'f9'), ('f5', 'f3'), ('f9', 'f8'), ('f9', 'f7'), ('f9', 'f1'), ('f9', 'f6'), ('f9', 'f2'), ('f9', 'f4')]
    Model 1: K2 Accuracy Score is -9305.069314371063
    Model 2: Manual Model based on HillClimbSearch is : [('f3', 'f4'), ('f3', 'f8'), ('f4', 'f9'), ('f9', 'f8'), ('f9', 'f1'), ('f9', 'f6'), ('f9', 'f2'), ('f1', 'f7'), ('f1', 'f6'), ('f5', 'f3')]
    Model 2: K2 Accuracy Score is -9491.989375840923
    Model 3: Manual Model based on HillClimbSearch is : [('f3', 'f4'), ('f3', 'f8'), ('f4', 'f9'), ('f9', 'f8'), ('f9', 'f1'), ('f9', 'f6'), ('f9', 'f2'), ('f5', 'f7'), ('f5', 'f3'), ('f1', 'f2')]
    Model 3: K2 Accuracy Score is -9330.623362759818
    Model 4: Manual Model based on HillClimbSearch is : [('f3', 'f4'), ('f4', 'f9'), ('f9', 'f1'), ('f9', 'f6'), ('f9', 'f8'), ('f5', 'f7'), ('f5', 'f3'), ('f1', 'f2')]
    Model 4: K2 Accuracy Score is -9426.247507740312
    Model 5: Manual Model based on HillClimbSearch is : [('f3', 'f4'), ('f4', 'f9'), ('f4', 'f7'), ('f9', 'f6'), ('f9', 'f8'), ('f1', 'f2'), ('f8', 'f5')]
    Model 5: K2 Accuracy Score is -9491.557927596972
    Best Bayesian Model with the highest accuracy score is thus Model 1
```

4 Task 3

In this task, we convert the best Bayesian networks we found for ‘th’ dataset and ‘and’ dataset, to Markov models. We then analyze the time taken by the 2 models to generate a 1000 samples (with GibbsSampling for Markov model and Bayesian sampling for Bayesian model). We end by comparing their accuracy scores.

Bayesian Network for ‘th’ has a poorer performance compared to the Markov model, in terms of accuracy. However, for the same dataset, in terms of time BN is faster than the MN. BN is faster than MN by a factor of 4.

For the ‘and’ data, Bayesian Nework performs considerably better than the one for ‘th’ data in terms of accuracy - Understandably as we made vast generalization in the BN of ‘th’ data. However, Markov model beats the BN in terms of accuracy. In terms of time, again the BN is considerably faster. BN is approximately 350 times faster.

The exact numerical values of accuracy and time taken are displayed below.

```
Task 3 - For 'th' and 'and' data:
C:\Users\nikhi\AppData\Local\Programs\Python\Python36\lib\site-packages\pgmpy\factors\discrete\DiscreteFactor.py:586: FutureWarning: Using a non-tuple sequence for multidimensional indexing is
deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  phi.values = phi.values[slice_]
C:\Users\nikhi\AppData\Local\Programs\Python\Python36\lib\site-packages\pgmpy\factors\discrete\DiscreteFactor.py:598: FutureWarning: Using a non-tuple sequence for multidimensional indexing is
deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  phi1.values = phi1.values[slice_]
C:\Users\nikhi\AppData\Local\Programs\Python\Python36\lib\site-packages\pgmpy\factors\discrete\DiscreteFactor.py:663: FutureWarning: Using a non-tuple sequence for multidimensional indexing is
deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  phi1.values = phi1.values[slice_]
C:\Users\nikhi\Documents\UB\CSE 674 - Advanced Machine Learning\Projects\Project 1\main.py:356: FutureWarning: Method .as_matrix will be removed in a future version. Use .values instead.
  data1 = task2_best_mm_samples[['x1']].as_matrix()
C:\Users\nikhi\Documents\UB\CSE 674 - Advanced Machine Learning\Projects\Project 1\main.py:357: FutureWarning: Method .as_matrix will be removed in a future version. Use .values instead.
  data2 = task2_best_bm_samples[['x1']].as_matrix()
    Bayesian Model for 'th' data : [('x1', 'x4'), ('x4', 'x6'), ('x6', 'x2'), ('x2', 'x3'), ('x3', 'x5')]
    Bayesian Model for 'th' data takes time : 0.12566351890563965
    Markov Model for 'th' data : [('x1', 'x4'), ('x4', 'x6'), ('x6', 'x2'), ('x2', 'x3'), ('x3', 'x5')]
    Markov Model for 'th' data takes time : 0.48863649368286133
    Markov Model for 'th' data has accuracy : 60.9
    Bayesian Model for 'and' data : [('f3', 'f4'), ('f3', 'f9'), ('f3', 'f8'), ('f5', 'f9'), ('f5', 'f3'), ('f9', 'f8'), ('f9', 'f7'), ('f9', 'f1'), ('f9', 'f6'), ('f9', 'f2'), ('f9', 'f4')]
    Bayesian Model for 'and' data takes time : 0.19983696937561035
    Markov Model for 'and' data : [('f1', 'f9'), ('f9', 'f2'), ('f9', 'f3'), ('f9', 'f4'), ('f9', 'f5'), ('f9', 'f6'), ('f9', 'f7'), ('f9', 'f8'), ('f3', 'f4'), ('f3', 'f8'), ('f3', 'f5')]
    Markov Model for 'and' data takes time : 78.48961925506592
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

4 Conclusion

In conclusion, the objective was achieved. Bayesian networks and Markov models were created for the 2 datasets – ‘th’ and ‘and’, and their results were compared in terms of their accuracy and time. We also got the chance to explore the pgm.py library in detail to build the above PGMs.