Report On

Learning Bayesian Networks: An Application to Kinect Data

Zahid Hassan Tushar & Mohammad Mohaiminul Islam

1 Introduction

This assignment is aimed to classify the body movements from the MSRC-12 Kinect gesture data set of Microsoft Research Cambridge [1]. Although the dataset consists of hundreds of sequences of skeletal body movements, the scope of this assignment is confined in classifying only four body positions namely crouch, right arm extended and both arms lifted. Two types of Bayesian models were implemented for the classification task. The first model we explored, was Naïve Bayes (NB) which considers only dependencies from the class variable to the other random variables. In the second model, we used Linear Gaussian Model (LGM) which also considers the dependencies between the positions of the different joints of the body. The comparative result shows that LGM outperforms NB as expected.

2 Method

Bayes' Theorem is a simple mathematical formula used for calculating conditional probabilities. One of the applications of Bayes theorem is the statistical inference. The basic form of Bayes theorem is expressed in (1).

$$P(B|E) = P(B) \cdot P(E|B)/P(E) \tag{1}$$

where P standing for probability, B for belief and E for evidence. P(B) is the probability that B is true, and P(E) is the probability that E is true. P(B|E) means the probability of B if E is true, and P(E|B) is the probability of E if B is true.

We will demonstrate in the subsequent section the adaption of Bayes theorem in NB and LGM. In both cases we used maximum likelihood estimation.

2.1 Naïve Bayes

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances. Given a problem instance to be classified, represented by a vector $\mathbf{x}=[x_1, x_2, \ldots\ldots, x_K]$ representing K features, it assigns to this instance probabilities $P(C_C|\boldsymbol{x})$ for each of C classes. These conditional probabilities are calculated using the formula in (2).

$$P(C_C|x) = P(C_C) \cdot P(x|C_C)/P(x) \tag{2}$$

Where $P(C_C)$ is the class prior probability, $P(x|C_C)$ is the likelihood and $P(x)$ is the normalization factor. In NB, we assume all the variables are mutually independent. Since the variables are continuous in nature, we assume a Gaussian conditional probability distribution for our likelihood. In (3) we denote the general expression of the likelihood estimation formula and in (4) we express the likelihood in its full form. However in practice, we used Log of likelihood to reduce the computational cost as shown in (5).

$$P(x|C_C) = Normal(x; \mu, \sigma^2) \quad (3)$$

$$L(\mu, \sigma^2 : x) = P(x|\mu, \sigma^2, C) = \prod_{i=1}^{K}\left(1/\sqrt{2\pi\sigma_i^2}\right) \cdot \exp(-(x_i - \mu_i)^2/2\sigma_i^2) \quad (4)$$

$$l(\mu, \sigma^2 : x) = Log(L) = -\sum_{i=1}^{K}(0.5 \cdot \log(2\pi\sigma_i^2) + (x_i - \mu_i)^2/2\sigma_i^2) \quad (5)$$

Before computing the likelihood, we need to estimate the parameters i.e. the mean and variance of each K variables. To calculate the parameters for the i^{th} variable we used the formula in (6) and (7). And finally for classifying a new instance, we need class prior probabilities for which we used the formula in (8).

$$\mu_i = (1/N)\sum_{j=1}^{N} x_i[j] \quad (6)$$

$$\sigma_i^2 = (1/N)\sum_{j=1}^{N}(x_i[j] - \mu_i)^2 \quad (7)$$

$$P(c_c) = Total\ number\ of\ instances\ with\ class\ label\ C_c/N \quad (8)$$

Where N represents the total number of instances in the dataset.

2.2 Linear Gaussian Model

In the linear Gaussian model we assumed a more complex model where the continuous variables have continuous parent variables. The LGM adaption of Bayes theorem can be found in (9). The probability of an instance to belong in Class c is given by (9). Each instance has K features or in other word, K variables.

$$P(C = c|instance) \propto P(C = c) \cdot \prod_{i=1}^{K} P(x_i|x_{p(i)}, C) \quad (9)$$

Where $x_{p(i)}$ is the parent of variable x_i.

The general expression for the likelihood can be seen in (10). The full equation form to compute the likelihood and log-likelihood are same as NB and are given in (4) and (5). To estimate the class prior probabilities, the formula is also same as NB and can be accessed in (6).

$$P(x|C_C) = Normal(x; \mu, \sigma^2) \quad (10)$$

Where mean, $\mu = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots\ldots\ldots + \beta_K x_K$

To estimate likelihood, we need to compute the mean and variance of our linear Gaussian model. The number of betas is related to the number of parents a variable has. To compute betas, we need to solve (11) which is in the form of Ax=b. In (11) we assumed each variable has K parent variables. Here we express variable with Y and parent variable with $X=[X_1, X_2, \ldots, X_K]$. For the variance, we use the formula given in (12).

$$\begin{pmatrix} E_D[Y] \\ E_D[Y.X_1] \\ \vdots \\ E_D[Y.X_K] \end{pmatrix} = \begin{pmatrix} 1 & E_D[X_1] & \cdots & E_D[X_K] \\ E_D[X_1] & E_D[X_1.X_1] & \cdots & E_D[X_1.X_K] \\ \vdots & \vdots & \ddots & \vdots \\ E_D[X_K] & E_D[X_K,X_1] & \cdots & E_D[X_K,X_K] \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_K \end{pmatrix} \quad (11)$$

Where $E_D[Y]$ is the average of all Y instances in the dataset and $E_D[Y.X]$ is the average of dot product of all Y and X instances in the dataset.

$$\sigma^2 = COV_D[Y,Y] - \sum_i \sum_j \beta_i \cdot \beta_j \cdot COV_D[X_i,X_j] \quad (10)$$

Where

$$COV_D[X,Y] = E_D[X.Y] - E_D[X] \cdot E_D[Y] \quad (13)$$

3 Implementation

This assignment has been written on Python language with numpy, sklearn, matplotlib, mlxtend and scipy packages. A framework has been provided from the lab with some functions implemented. We build our model using the framework with some slight variations and addition of a few functions for smooth program flow.

We have implemented two classes for each model: NB and LGM. Each class contains three methods *train, predict* and *compute_logprobs.* When *train* method of NB class is executed, it sorts the instances based on their classes and uses *calculatePrior* function to compute prior probabilities for each class. Then *fit_gaussian* method is applied to compute model paramters i.e. mean and variance for each variable. In our dataset, we have 20 joints with each having (x, y, z) positions, providing us a total of 60 variables. Since the mini dataset is consisted of 4 classes only, we receive two matrixes for mean and variance each having a dimension of 20x3x4. When NB model is executed using the *predict* method, it utilizes the *compute_logprobs* function to compute the log probabilities for each class given the instance. Then the result is normalized and converted back to linear space from log space using the function *normalize_logprobs*.

In case of LGM model, the methods are similar to NB. However, the estimation of parameters is different from NB. For this model, we have assumed that out of 20 joints from the dataset, 19 joints have parent joint except joint X_0. Therefore we compute mean for this joint separately which results in a 1x3x4 matrix. For other joints we estimate the mean using the betas. Betas are computed based on the parent joint's variables. The adaptive formulas for likelihood of each position per joint are shown in (14), (15) and (16).

$$P(x_i | x_{p(i)}, y_{p(i)}, z_{p(i)}, C) = Normal(\beta_{01} + \beta_{11}x_{p(i)} + \beta_{21}y_{p(i)} + \beta_{31}z_{p(i)}; \sigma^2) \quad (14)$$

$$P(y_i | x_{p(i)}, y_{p(i)}, z_{p(i)}, C) = Normal(\beta_{02} + \beta_{12}x_{p(i)} + \beta_{22}y_{p(i)} + \beta_{32}z_{p(i)}; \sigma^2) \quad (15)$$

$$P(z_i | x_{p(i)}, y_{p(i)}, z_{p(i)}, C) = Normal(\beta_{03} + \beta_{13}x_{p(i)} + \beta_{23}y_{p(i)} + \beta_{33}z_{p(i)}; \sigma^2) \quad (16)$$

Where p(i) indicates the index of the joint that is the parent of the i-th joint.

When a new instance is needed to be classified using LGM, we use *predict* method. It computes the posterior probabilities of each class for that instance using the *compute_logprobs.* The formulas involved in this computation can be found in (17). Finally, the transformation of the probabilities from log space to linear space is done similarly as NB.

$$P(C = c|instance) \propto P(C = c) \prod_{j=1}^{20} P\left(x_i, y_i, z_i \middle| x_{p(i)}, y_{p(i)}, z_{p(i)}, C = c\right) \quad (17)$$

Where

$$\begin{aligned} &P\left(x_i, y_i, z_i \middle| x_{p(i)}, y_{p(i)}, z_{p(i)}, C = c\right) \\ &= P\left(x_i \middle| x_{p(i)}, y_{p(i)}, z_{p(i)}, C = c\right) \times P\left(y_i \middle| x_{p(i)}, y_{p(i)}, z_{p(i)}, C = c\right) \\ &\times P\left(z_i \middle| x_{p(i)}, y_{p(i)}, z_{p(i)}, C = c\right) \end{aligned}$$

For robust performance evaluation of the model, we have implemented k-fold cross validation. To implement this, we need to call *cv* method by providing the model name, dataset and number of folds. Finally, we used a confusion matrix generator to visualize the response of our model.

4 Dataset

In this assignment we are using a mini version of the MSRC-12 Kinect gesture data set of Microsoft Research Cambridge [1]. This mini version was provided by the ABM lab. It contains instances for four body positions: crouch, right arm extended to one side, right arm extended to the front and arms lifted. The training set has 2045 instances while the validation set is composed of 120 instances. Our goal is to train two models: NB and LGM with this dataset and evaluate the comparative response of these models using the validation set. The body positions are encoded using a 20x3 matrix where each row is the position in space (x, y, z) of each of the 20 joints. In Fig. 1 we have shown the order in which the joints appear for one instance of the class both arms lifted.

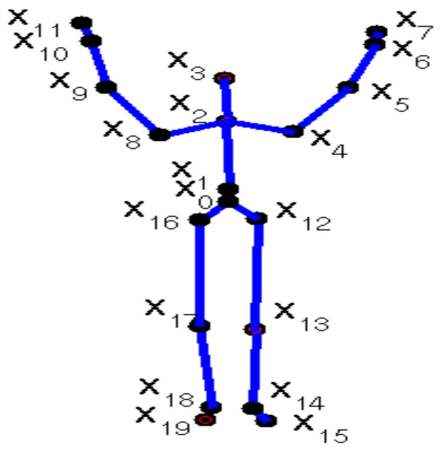

Fig. 1. Order of the joints from the class both arms lifted.

The lab provided a *template_data.zip* which included 3 files: *data.mat*, *validation_data.mat* and *ejemplolineargaussian.mat.* The *data.mat* file contained the following three elements:

1. Data: 3D matrix with dimensions #joints x 3 x #instances. In our case 20x3x2045.
2. labels: a vector with class label for each instance
3. individuals: a vector with the identification of each individual.

The *validation.mat* file contains the similar elements as above but for 120 instances. The *ejemplolineargaussian.mat* file contains demo results for betas and sigma calculation required by LGM.

5 Results and Analysis

In this experiment we have implemented two models Naive Bayes and Linear Gaussian Model for classifying the gesture data. Linear Gaussians Model considers the relation between the parent and child joint whereas Naive Bayes does not. According to the result it can be seen, that the Linear Gaussian Model yields better result than Naive Bayes which implies prediction is better when we are considering the relation between the parent and child joint

.

To evaluate our model we have employed Cross-Validation on both of our models. We have done a 7-fold Cross validation following Table. 1 summarizes the accuracy in each fold. We can also see the Cross Validation accuracy plotted in Fig. 2 and Fig. 3. We can tell from the following table and graph that For each fold, the Linear Gaussian Model showed a better result than Naive Bayes.

Table 1. Performance Evaluation by 7-Fold Cross Validation

Model	Folds							Average Accuracy
	01	02	03	04	05	06	07	
NB	94.19	93.49	93.83	95.20	96.57	96.91	96.23	95.20
LGM	99.31	99.65	99.65	99.65	100	99.31	99.65	99.56

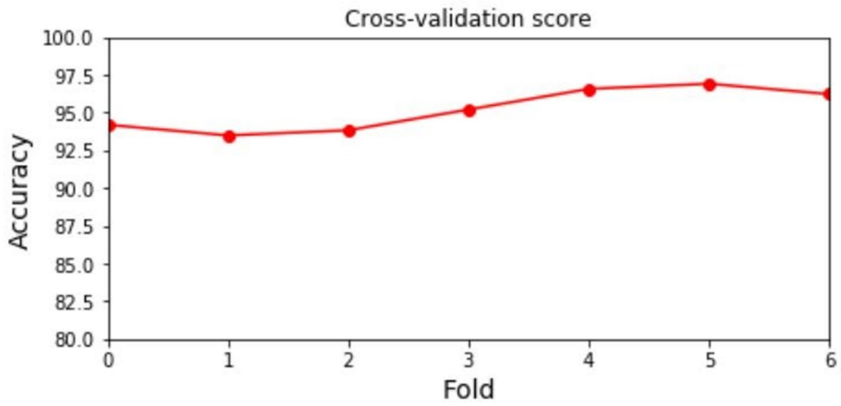

Fig. 2. Cross validation Score for Naive Bayes Model

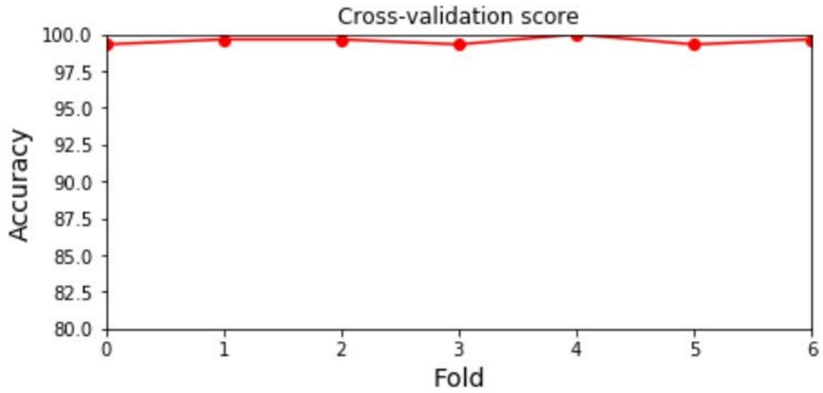

Fig. 3. Cross validation Score for Linear Gaussian Model

We have also tested our model sets on the test set that was provided with the assignment. In this test set there are 40 instances consisting of 4 classes. In this case both of our models performed slightly poorer than what we have seen for cross validation. Table 2 summarizes the complete class wise result of NB and LGM with some of

the measures commonly used to measure the performance of machine learning algorithms.

Table 2. Accuracy Measure on Test Set

Model	Class	Accuracy	Precision	Recall	F1-score
NB	Crouch	0.95	0.90	1.00	0.95
	Right arm extended		1.00	1.00	1.00
	Left arm extended		1.00	0.91	0.95
	Both arms lifted		0.90	0.90	0.90
LGM	Crouch	0.97	1.00	1.00	1.00
	Right arm extended		1.00	1.00	1.00
	Left arm extended		1.00	0.91	0.95
	Both arms lifted		0.90	1.00	0.95

Also Fig.4 and Fig.5 depicts the confusion matrix for NB and LGM where it can be seen that Naive Bayes model misclassified the one instance from crouch and both arms lifted class. On the other hand LGM misclassified just one instance from both arms lifted class. Though it can be determined which classes are having the most error for both of these models due to the lack of testing data.

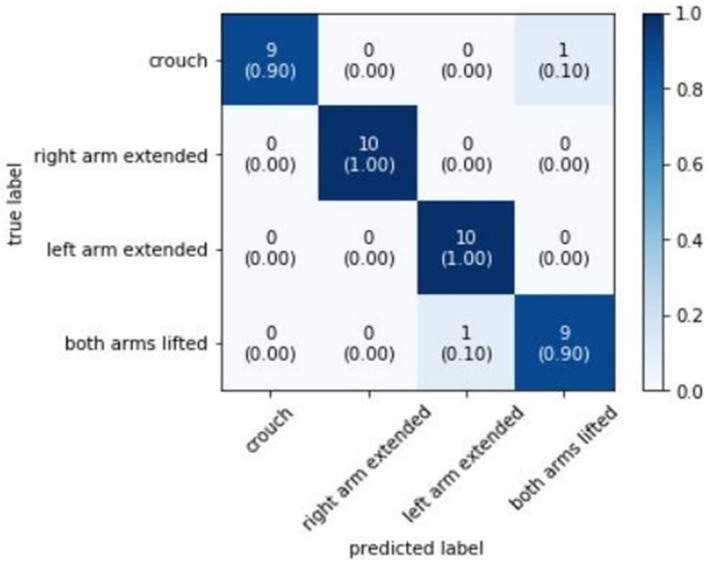

Fig. 4. Confusion matrix for test data on NB model

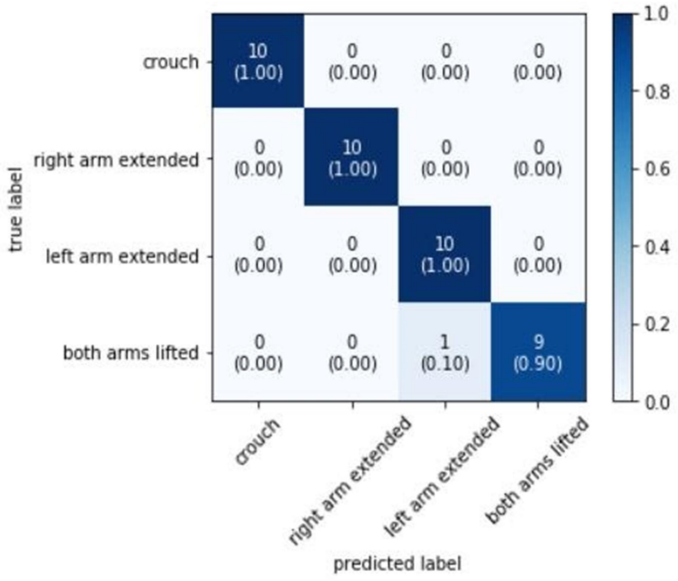

Fig. 5. Confusion matrix for test data on LGM model

6 Conclusion

In conclusion, NB is a simple model which assumes all the variables are mutually independent while LGM also considers the dependencies between the several joints of the body. LGM being a complex model compared to NB demonstrates better accuracy. The cross-validation scheme indicates a slight increase in the accuracy of the both model. Finally, the accuracy of the both model may be further improved if the full dataset was used.

References

1. https://www.microsoft.com/en-us/download/details.aspx?id=52283&from=http%3A%2F%2Fresearch.microsoft.com%2Fen-us%2Fum%2Fcambridge%2Fprojects%2Fmsrc12%2F