Latent Class Poisson Regression

1 Introduction

1.1 General Description

In this report, we'll perform latent class Poisson regression analysis using GAM and EM algorithm. We use GAM method to regress independent observations of count data on explanatory variables. In terms of count data being incomplete data set, we have a corresponding latent class variable and we use EM algorithm derive the parameters' update rules in the regression functions regarding to the missing information on data clusters. And also, we use monomial basis functions to expand the explanatory variables for the conditional GAMs.

1.2 Notation Interpretation

- $N :=$ number of data points
- $D :=$ The original dimension of input data
- $x_{id} :=$ the i-th data point in d-th dimension
- $C :=$ number of clusters
- $z_{ic} :=$ the probability that i-th data point belongs to cluster c

1.3 Data Preparation

The data set has three components: X, Y and Z. X denotes the $N \times D$ explanatory variables, Y denotes the independent observations and Z denotes the label information associated with data. Here we assume there are total C classes for all the observations Y. In details, Y_1, ..., Y_n are consisted of C classes and the i_{th} observation Y_i given it comes from the c_{th} class has a Poisson distribution with a rate $\lambda_{i|c}$ as:

$$f_c(y_i|\lambda_{i|c}) = \frac{e^{-\lambda_{i|c}}(\lambda_{i|c})^{y_i}}{y_i!},\ i = 1, ..., n,\ c = 1, ..., C.$$

The log density function is:

$$log(f_c(y_i|\lambda_{i|c})) = y_i log(\lambda_{i|c}) - \lambda_{i|c} - log(y_i!) \tag{1}$$

Z contains information for labels in which Z_i denotes the label associated with data point i. For Z = 1 or 2, the corresponding data point belongs to the first or second cluster. For Z = 0, the corresponding data point is denoted as having missing label. Using a latent variable Z, we can create a responsibility matrix $[\mathbf{z}]_{N\times D}$ to represent the label information with each entry z_{ic} denote the probability that the i_{th} data point belongs to the c_{th} cluster. To be specific:

For any data point y_i whose label is unknown, we assume the probability that a data point with missing label has even probabilities being in each of all the clusters C:

$$z_{ic} = z_c(y_i) = \begin{cases} 1, & \text{if } y_i \in c_{th} \text{ cluster, with probability = } 1/C \\ 0, & \text{otherwise} \end{cases}$$

For data point y_i whose label is known, we assume:

$$z_{ic} = z_c(y_i) = \begin{cases} 1, & \text{if } y_i \in c_{th} \text{ cluster, with probability = } 1 \\ 0, & \text{otherwise} \end{cases}$$

In this project, N = 500, D = 2.

2 Model Setup

2.1 Poisson Regression with GAM

In Poisson regression within a mixture of General Additive Models, we have:

$$\begin{aligned} log(E[Y_i|X_i = x_i, Z_i = z_i]) &= log(E[Y_i|X_i = x_i, z_{ic}]) \\ &= log(\lambda_{i|c}) = \beta_{0c\bullet} + \sum_{d=1}^{D}[\beta_{dc0} + \beta_{dc1}x_{id} + \beta_{dc2}{x_{id}}^2 + ... + \beta_{dcP}{x_{id}}^P] \end{aligned} \quad (2)$$

In monomial basis, β has a dimension of $C \times (D \times (P+1))$, where P denotes the number of degree in the polynomials. Also, let $\alpha = \alpha_1, ..., \alpha_C$ and let α_c denote the proportion of the c_{th} class with $\sum_{c=1}^{C} \alpha_c = 1$. Thus, a latent class Poisson regression model is a mixture of Poisson log-linear regression distribution with a latent variable C given by:

$$f(y_i|\alpha, \beta) = \sum_{c=1}^{C} \alpha_c f_c(y_i|\lambda_{i|c}),\ i = 1, ..., N$$

Then, given the responsibility matrix represented by the latent class variable $z_i = (z_{i1}, ..., z_{iC})^T$ with its distribution function $f(z_i) = \prod_{c=1}^{C} {\alpha_c}^{z_{ic}}$, we have:

$$f(y_i|z_i) = \prod_{c=1}^{C} (f_c(y_i|\beta_c))^{z_{ic}}$$

and

$$f(y_1, ..., y_N, z_1, ..., z_N) = \prod_{i=1}^{N} f(y_i, z_i) = \prod_{i=1}^{N} f(y_i|z_i)f(z_i) = \prod_{i=1}^{N}(\prod_{c=1}^{C} (f_c(y_i|\beta_c))^{z_{ic}} {\alpha_c}^{z_{ic}}.$$

Thus, the log likelihood function is:

$$logL = \sum_{i=1}^{N}\sum_{c=1}^{C} z_{ic} log f_c(y_i|\beta_c) + \sum_{i=1}^{N}\sum_{c=1}^{C} z_{ic} log \alpha_c$$

Also, β is a matrix that can be represented as:

$$\tilde{\beta} \in R^{C \times (D \times P+1)}$$

where

$$\beta_{cdp} := \text{Regression Coefficient of } [x_{id}]^p, \quad \text{for } \forall i \in 1, ..., N$$

2.2 Parameters in Monomial Basis

With mononial basis function, we expand input data and rewrite it into the following input data matrix:

$$X = \begin{bmatrix} 1, & x_{11} & \dots & x_{11}^P, & x_{12} & \dots & x_{12}^p, & \dots \\ 1, & x_{21} & \dots & x_{21}^P, & x_{22} & \dots & x_{22}^p, & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1, & x_{N1} & \dots & x_{N1}^P, & x_{N2} & \dots & x_{N2}^p, & \dots \end{bmatrix}_{[N\times(D\times P+1)]}$$

2.3 Initial Guess Setup

Before implementing the EM algorithm, we need to perform initial guess as model setup. In our model, we make initial guess for the responsibility matrix $[\mathbf{z}]_{N\times L}$.

For each data point, if the label is given, we assign one to that label and 0 else; if the label is missing, we uniformly random select a label for it.

Once we have done the naive guessing, we can compute the mean for each label as our first naive λ value.

2.4 E-Step

In E-Step, if the data point y_i has label, we just keep the label and DO NOT update the responsibility throughout the iteration process. On the other hand, if the data has no label, we use $E(z_{ic}|y_i, \alpha, \beta_c)$ to estimate z_{ic}.

$$\begin{aligned} f(z_{ic}|y_i) &= \frac{f(y_i, z_{ic})}{f(y_i)} \\ &= \frac{f(yi|z_{ic})f(z_{ic})}{\sum_{s=1}^{C}\alpha_s f_s(y_i|\beta_s)} \\ &= \frac{(\alpha_c f_c(y_i|\beta_c))_{ic}^{z}\sum_{s=1,s\neq c}^{C}\alpha_s f_s(y_i|\beta_s)^{(1-z_{ic})}}{\sum_{s=1}^{C}\alpha_s f_s(y_i|\beta_s)} \end{aligned}$$

$$\begin{aligned} \hat{z}_{ic} &= E[z_{ic}|y_i, \alpha, \beta_c] \\ &= 1\cdot\frac{\alpha_c f_c(y_i|\beta_c)}{\sum_{s=1}^{C}\alpha_s f_s(y_i|\beta_s)} + 0 \\ &= \frac{\alpha_c f_c(y_i|\beta_c)}{\sum_{s=1}^{C}\alpha_s f_s(y_i|\beta_s)} \end{aligned}$$

Thus,

$$E(logL) = \sum_{i=1}^{N}\sum_{c=1}^{C}\hat{z}_{ic}logf_c(y_i|\beta_c) + \sum_{i=1}^{N}\sum_{c=1}^{C}\hat{z}_{ic}log\alpha_c$$

2.5 M-Step

In M-Step, we assume that all data points has correct labels, we maximize the likelihood by updating each class density parameters using the data points assigned to that density. To maximize $E(logL)$ with the restriction $\sum_{c=1}^{C}\alpha_c = 1$. That is:

$$\frac{\partial}{\partial \alpha_c}(E(logL) - \gamma(\sum_{c=1}^{C} \alpha_c - 1)) = 0,$$

We have that

$$\hat{\alpha}_c = \frac{\sum_{i=1}^{N} \hat{z}_{ic}}{N}.$$

Our objective function is the log likelihood

$$L(\beta) = \sum_{i=1}^{N}\sum_{c=1}^{C} \hat{z}_{ic} log f_c(y_i|\beta_c) + \sum_{i=1}^{N}\sum_{c=1}^{C} \hat{z}_{ic} log \hat{\alpha}_c.$$

In each M-step, we are maximizing the objective function with respect to β. Since

$$log f_c(y_i|\lambda_{i|c}) = y_i log(\lambda_{i|c}) - \lambda_{i|c} - log(y_i!)$$

Then

$$\frac{\partial}{\partial \beta_{cdp}} L(\beta) = \sum_{i=1}^{N} \hat{z}_{ic} \frac{\partial log f_c(y_i|\lambda_{i|c})}{\partial \lambda_{i|c}} \frac{\partial \lambda_{i|c}}{\partial \beta_{cdp}} \tag{3}$$

$$= \sum_{i=1}^{N} \hat{z}_{ic} (\frac{y_i}{\lambda_{i|c}} - 1) \times x_{id}^{p} \tag{4}$$

$$-\mathbb{E}\left[\frac{\partial^2 L(\beta)}{\partial \beta_{cd'p} \partial \beta_{cdp}}\right] = \sum_{i=1}^{N} \hat{z}_{ic} \lambda_{i|c} [x_{id'} x_{id}]^p = [(X^p)^T W_c X^p]_{[dd']} \tag{5}$$

where

$$W_c = \begin{bmatrix} \hat{z}_{c1}\lambda_{1|c} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \hat{z}_{cN}\lambda_{N|c} \end{bmatrix}$$

Note that given class c, $\mathbb{E}(y_i) = \lambda_{i|c}$ and $\lambda_{i|c}$ is good point to approximate y_i. By first order Taylor Expansion, we have

$$\gamma_{i|c} = log(y_i) \approx log(\lambda_{i|c}) + (y_i - \lambda_{i|c}) \times \frac{1}{\lambda_{i|c}}.$$

Then we have

$$X\beta_c = log(\lambda_{i|c}) \approx log(y_i) = \gamma_{i|c}$$

It implies

$$\begin{aligned} X^T W_k X \beta_c &= X^T W_k \gamma_{i|c} \\ X^T (W_k)^{1/2} (W_k)^{1/2} X \beta_c &= X^T (W_k)^{1/2} (W_k)^{1/2} \gamma_{i|c} \\ ((W_k)^{1/2} X)^T ((W_k)^{1/2} X) \beta_c &= ((W_k)^{1/2} X)^T \gamma_{i|c} \end{aligned}$$

Define matrix A as $A = (W_k)^{1/2} X$. According to the singular value decomposition (SVD), for any matrix $A \in R^{m \times n}$, A can be factorized as the product of three matrices $A = U\Sigma V^T$, where:

$$\Sigma = \begin{bmatrix} \sigma_1 & & & & 0 \\ & \ddots & & & \\ & & \sigma_r & & \\ & & & \ddots & \\ 0 & & & & 0 \end{bmatrix}, \quad \text{where } \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$$

Then the β_c can be written as

$$\beta_c = V\Sigma U^T W^{1/2} \gamma_{i|c}$$

2.6 EM Algorithm

The whole procedure for EM algorithm can be summarized as follows:

Algorithm 1: EM Algorithm

Result: $\tilde{\boldsymbol{\beta}}, \boldsymbol{\alpha}, \tilde{\boldsymbol{Z}}, \tilde{\boldsymbol{\lambda}}$
initialization $\tilde{\boldsymbol{Z}}, \tilde{\boldsymbol{\beta}}_{\boldsymbol{0}}$;
while *Run Long Enough* **do**
  $log(\lambda_{i|c}^{(t)}) \leftarrow \boldsymbol{x_i \beta_c^{(t-1)}}$
  $\gamma_{i|c}^{(t)} \leftarrow log(\lambda_{i|k}^{(t-1)}) + \frac{y_i}{\lambda_{i|c}} - 1$
  $\hat{\alpha}_c^{(t)} \leftarrow \frac{\sum_{i=1}^{N} \hat{z}_{ic}^{(t-1)}}{N}$
  $\boldsymbol{W_c}^t \leftarrow \hat{z}_{ci} \lambda_{i|c}^{(t)}$
  $\tilde{\boldsymbol{\beta}}^{(t)} \leftarrow (\tilde{\boldsymbol{X}}^T \boldsymbol{W_c}^{(t)} \tilde{\boldsymbol{X}})^{-1} \tilde{\boldsymbol{X}}^T \boldsymbol{W_k}^{(t)} \boldsymbol{\gamma_c}^{(t)}$ //use SVD
  $\hat{z}_{ic} \leftarrow \frac{\alpha_c f_c(y_i|\beta_c)}{\sum_{s=1}^{C} \alpha_s f_s(y_i|\beta_s)}$;
end

The condition of While Loop stop rule is that we run the algorithm until we see that the change in responsibility probability becomes very small. That is

$$||\tilde{\boldsymbol{Z}}^{(t)} - \tilde{\boldsymbol{Z}}^{(t-1)}|| \to \epsilon, \quad \text{for any } \epsilon > 0,$$

where the distance measure can be any proper norm function.

3 Output and Visualization

First of all, we try to get a idea about the general behavior of data. The scatter plots for given data are as follows:

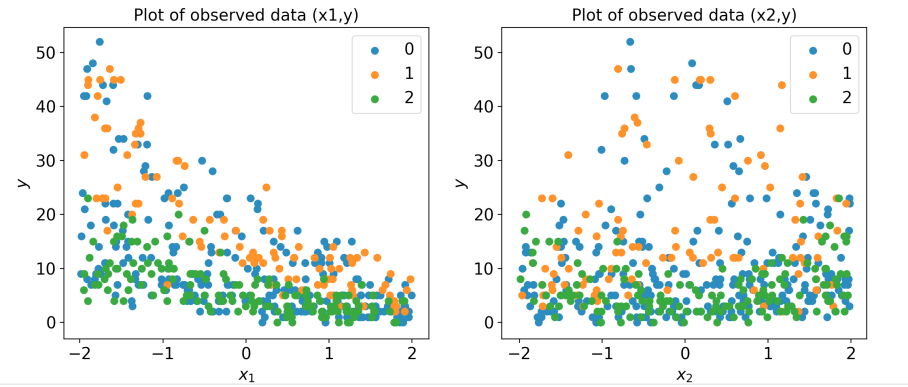


Figure 1: Scatter-Plot of $X1 \sim Y$ and $X2 \sim Y$:

3.1 Model Validation

As we have pointed out, the model has a large number of parameters required to be updated at each iteration; however, there are also two kinds of hyper parameters needed to be tuned before we start to train.

The value of C indicates how many clusters we believe that can capture the underlying structure of data. If the C is chosen such that larger than the true; we will see that eventually this cluster will vanish or contain very few points. In the extreme case, when $C \approx N$, we might see that each cluster contain only one point or few points, our model is too complex and will potentially over-fit. On the other hand, $C \ll N$, it over simplifies the data structure and under-fit the model. Similarly, the value P: maximum degree of polynomial also plays an important role. If $P = 1$, we are simply doing linear regression, the model will be unable to learn any nonlinear function; but if P is too larger such that expanding the dimension of $D * P + 1 \gg N$, terrible thing happens that, when solving for β, the matrix $X^T W X$ has rank at most $N \ll D * P + 1$, some rows of the matrix reduced to zero leads to zero determinants and we will have infinitely many solutions to β.

Hence, how to choose appropriate values of hyper parameters critically decides the performance of our model. In this project, we perform cross-validation for several values of P, and C; the sum weighted residual scores is used for evaluation. As we minimize:

$$Q_{wls} = \sum_{i}^{N} w_i (y_i - X_i \beta)$$

Here are the outputs for the case when $C = 2$, in which for different P we have:

- $P = 1$:
$$Q_{wls} = 1461.1492762413068$$
- $P = 2$:
$$Q_{wls} = 1042.866400095587$$
- $P = 3$:
$$Q_{wls} = 1047.2026428783674$$
- $P = 4$:
$$Q_{wls} = 1062.7265478221814$$

Apparently, the optimal value is obtained when $P = 2$. Note that the model is too simple for any nonlinear when $P = 1$.

Now we analyze C by fixing the effect of P. Let $P = 3$ and then we have:

- $C = 1$:
$$Q_{wls} = 2640.286906865586$$
- $C = 2$:
$$Q_{wls} = 1042.8664000956103$$
- $C = 3$:
$$Q_{wls} = 4438.555909544021$$
- $C = 4$:
$$Q_{wls} = 4438.555909883889$$

Accordingly, the model believes that there are only two clusters exists for given data.

3.2 Optimal Model

The convergence performance is shown as in the following plot, with the horizontal axis representing the number of iterations and the vertical axis representing the log-likelihood. We can see from the plot that the model converges after approximately 5 iterations which can be visualized with the log-likelihood being maximized.

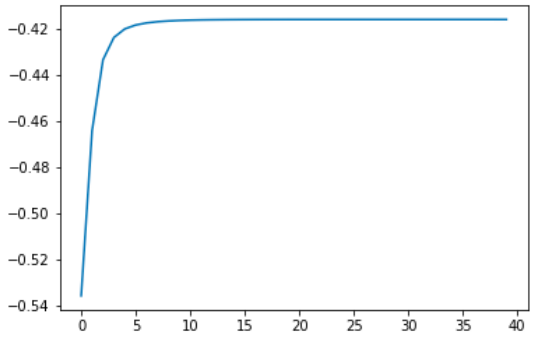
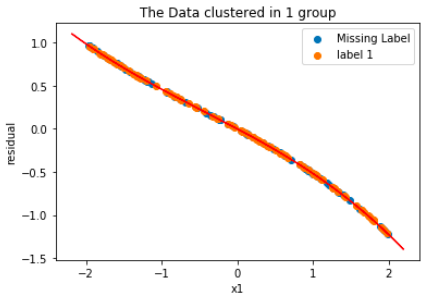
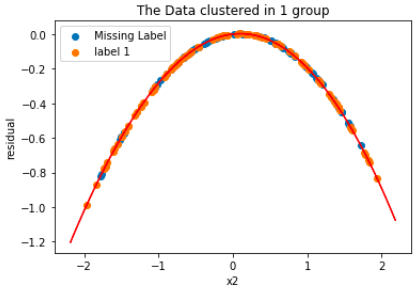


Figure 2: convergence performance of simulated data: x-axis: number of iterations, y-axis: log-likelihood

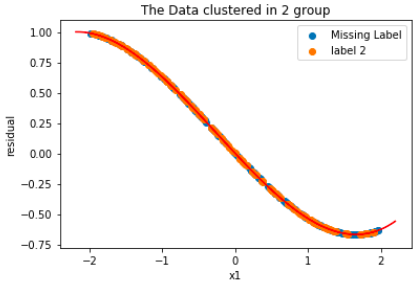
When we train the given data, the data points whose labels are missing will gradually be assigned into the other two clusters when performing iterations, so the cluster containing data points with missing labels will be squeezed gradually. Thus, when the model finally converges, the the proportion of data points with missing labels compared to the total number of data points will be very close to zero, which is consistent with the result that the corresponding α value is very close to zero. The results for residuals are shown as below for visualization:



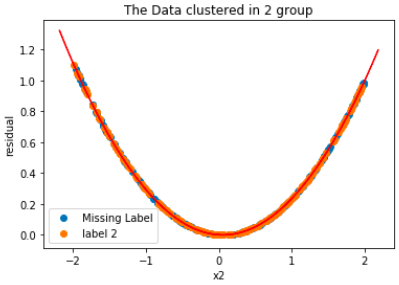
(a)



(b)



(c)



(d)

4 Simulator Results

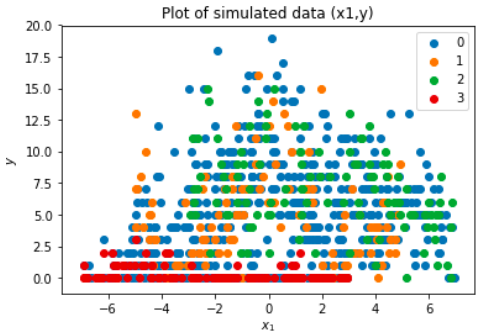
The data is simulated by defining: $x = (x_1, x_2) \sim \mathcal{N}(0, 1)$. Consider the function $f_c : \mathbb{R}^2 \to \mathbb{R}$, where:

$$f_1 = |\frac{1}{1+e^{-x_1}} + 2 * cos(x_2) + \epsilon|$$

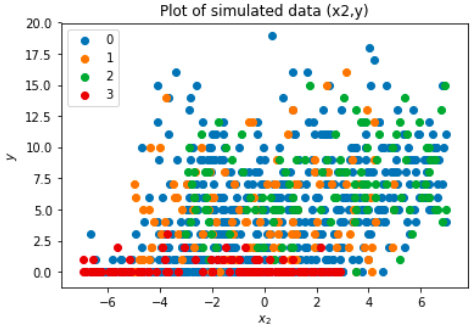
$$f_2 = |\sqrt{c - {x_1}^2} + log(x_2) + \epsilon|$$

$$f_3 = |0.3 * [(\mathcal{N}(0,1) + 5) * 7] + 0.7 * [(\mathcal{N}(0,1) + 2) * 2]\,|$$

We generate 1000 points from above, in which 333 data points from each individual f, then sample $y_i \sim Poi(f)$. The pair (x_i, y_i) will be our input data. The scatter plots for simulated data are as follows:



(e)



(f)

According to the convergence performance of our simulated data below, it shows the convergence speed of our algorithm is quite fast:

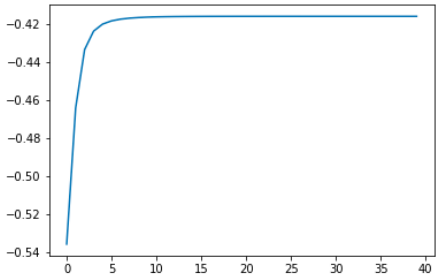
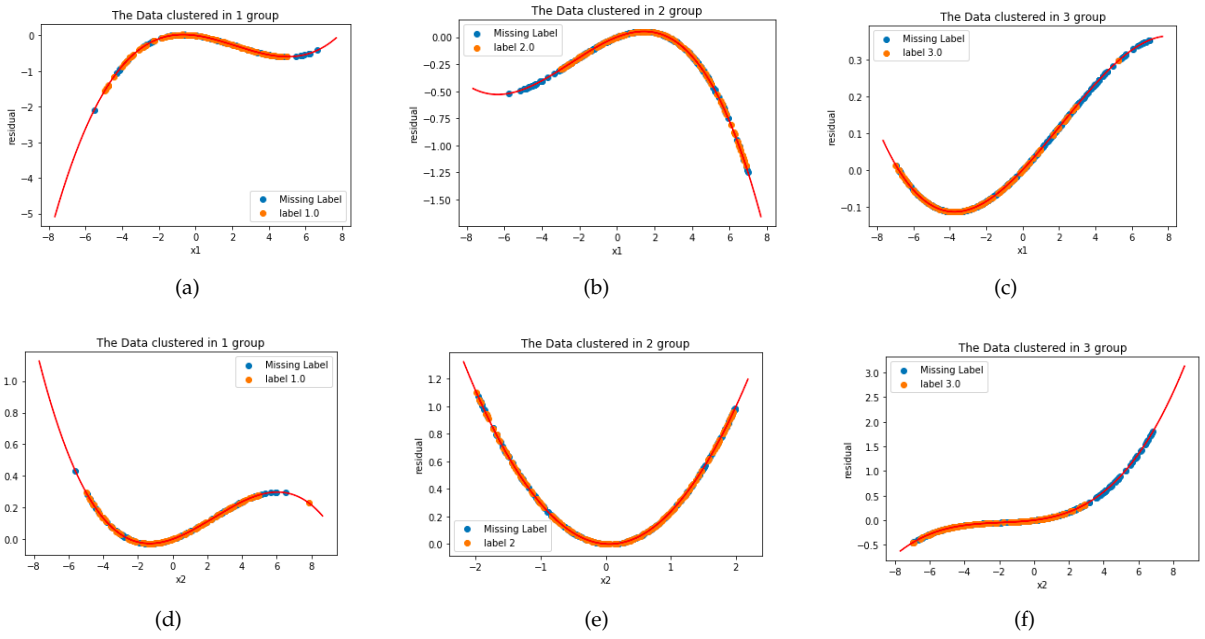


Figure 3: convergence performance of simulated data: x-axis: number of iterations, y-axis: log-likelihood

The results for residuals are shown as below:



5 Further Consideration of Algorithm

There are several important things needed to be kept in mind:

The first is about the numerical Issues. Note that Poisson density needs to take exponential of the linear combination of data and applying monomial basis function can make values even harder to control. Although in our simple data set with small counts and small values of X, the overflow/underflow numerical issues are not encountered. However, when dealing with more complicated data, straight execution of the algorithm will not give us good results. Several tricks are adopted in our project: (i) Instead of using Poisson density, we adopt log Poisson to keep values in lop space to avoid overflow. (ii) To update the responsibility matrix, $\frac{exp(A)}{\sum_{c'} A_{c'}} = \frac{exp(A_c - B)}{\sum_{c'} A_{c'} - B}$ where $B = max_j A_j$. Both measures keep our values in safety zone

during the iterations even when given very large value of P

Another thing is about the situation if we believe that true number of C is more than number of already known classes. In this case, we have no data points that are labeled with these mysterious class. We need to adopt some prior belief that each class contains at least one data point (or we would decrease the value of C); to do this by random sampling C data points from normal distribution and each of them assigned with label respectively. (There is very low probability that at our initial guess, there is an regressor that algorithm assigns any label to any one of data points to it). We assure that at our initial guess, each regressor has an intercept to begin with; even if our guess of the value C is wrong, it is still okay, as the algorithm will figure out which regressor a data point belongs to. And if there is no data belonging to it, it will eventually only contains that single noise fake data while other regressors can still well-capture the underlying structure of the data

6 Conclusion

From our model, using the given data and cross validation method for checking we conclude that there are only two clusters for our given data. The model converges well and the residual plot also fits the regression line very well. Then we build up a simulator to randomly generate some data points from known non-linear functions and also generate random independent observations from Poisson distribution. We run the regression model and again our model fits very well by looking at the convergence performance and residuals plot.

References

[1] Yang, MS. & Lai, CY.
Soft Comput (2005) 9: 519. https://doi.org/10.1007/s00500-004-0369-4

.