GRA 4152 Object Oriented Programming

Title: Generalized Linear Models and Data Loaders Report

Wise flow ID: 1101390

Date: 2024.11.17

1, Generalized Linear Models (GLM)

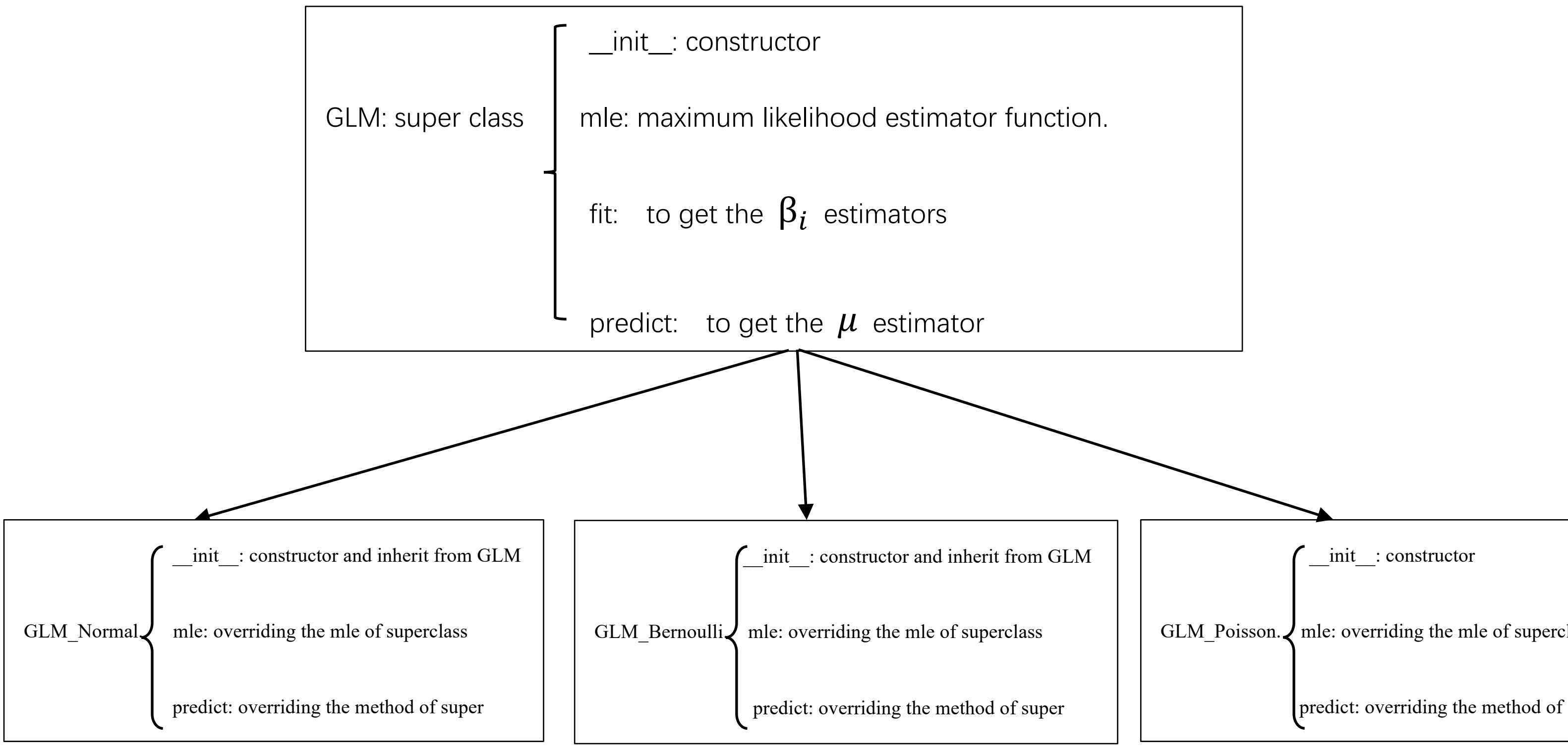
1.1 Math theories involved

GLM takes the form: $\eta_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \cdots + \beta_p x_{i,p} = x_i{}^T \beta$,
where $\beta = (\beta_0, \beta_1, \ldots \beta_p)^T$ and $x_i = (1, x_{i,1}, x_{i,2}, \ldots x_{i,p})^T$, for i=1,....T

Note that the first element of x_i is 1, it is used to get β_0. Thus, the matrix of variable x should add a one column first. To get the β_i, minimizing the negative likelihood is used.

$g(\mu_i) = \eta_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \cdots + \beta_p x_{i,p} = x_i{}^T \beta$
After getting the estimators of β_i, through the $g(\mu_i)$ and link function, we can get the mean expressions as follows:
Normal: $\mu = \eta$
Bernoulli: $\mu = 1/(1 + e^{-\eta})$
Poisson: $\mu = e^{\eta}$

1.2 Hierarchy map



1.3 super class (#solutions to '1-a-Superclass')

The super class is named GLM and the code is as follows:

```python
class GLM():
    def __init__(self,x,y):
        column_ones=np.ones((x.shape[0],1))
        self.x=np.concatenate((column_ones,x),axis=1) #add one column in matrix x
        self.y=y


    def mle(self,beta,x,y):  #mle is maximum likelihood estimator funcntion
        raise NotImplementedError     #use abstract method,let subclass will specify it later.


    def fit(self,x,y):
        init_beta=np.repeat(0.1,self.x.shape[1])
        mini=minimize(self.mle,init_beta,args=(self.x,self.y))
        self.beta=mini.x
        print(f'Estimated beta are:{self.beta}')


    def predict(self,x):    # according to different link funtion, it is implemented in subclass.
        raise NotImplementedError
```

(1) def __init__(self, x,y):

It uses constructor __init__ to initialize the super class. According to the math theory, to get β_0, the x matrix needs a one column. Thus, the first two lines code is to implement it. 'self.y=y' is used to define 'self.y', which will be used later.

(2) def mle (self, beta, x, y): **(#solutions to 1-b-'Abstract methods')**

It uses an abstract method, because it will not be implemented in super class, but in subclass. This method is used to get a log-likelihood value, which is used as a parameter to the fit method.

(3) def fit (self, x, y):

It is used to estimate the β_i. Firstly, It sets an initial value of 0.1 to all β_i. Then, utilize the mle value got from mle method, initial β_i and implement the function of minimize in scipy to calculate β_i. The fit method is generally used in the following three subclasses. Thus, the three subclasses do not need to override or rewrite it, only need to inherit it.

(4) def predict (self,x):

It is used to get the estimated mean μ. According to math theory, every subclass has its own way to get μ. Consequently, it uses an abstract method not to implement it, but it is implemented in the subclass.

1.4 Sub Class: GLM_Normal, GLM_Bernoulli, GLM_Poisson(#solutions to '1-a-subclass')

Here, it has three subclasses, and the code is as follows:

```python
class GLM_Normal(GLM):
    def __init__(self,x,y):
        super().__init__(x,y)

    def mle(self,beta,x,y):
        eta=np.matmul(self.x,beta)
        mu=eta
        log_lik=-np.sum(norm.logpdf(y,mu))
        return log_lik

    def predict(self,x):
        eta=np.matmul(self.x,self.beta)  # self.beta is calculate from fit function
        mu=eta  # use the identity function
        return mu


class GLM_Bernoulli(GLM):
    def __init__(self,x,y):
        super().__init__(x,y)

    def mle(self,beta,x,y):
        eta=np.matmul(self.x,beta)
        mu=1/(1+np.exp(-eta))  # getting from Bernoulli link function
        log_lik=-np.sum(bernoulli.logpmf(y,mu))
        return log_lik

    def predict(self,x):
        eta=np.matmul(self.x,self.beta)  # self.beta is calculate from fit function
        mu=1/(1+np.exp(-eta))  # getting from the link function
        return mu

class GLM_Poisson(GLM):
    def __init__(self,x,y):
        super().__init__(x,y)

    def mle(self,beta,x,y):
        eta=np.matmul(self.x,beta)
        mu=np.exp(eta)  # getting from Poisson link function
        log_lik=-np.sum(poisson.logpmf(y,mu))
        return log_lik

    def predict(self,x):
        eta=np.matmul(self.x,self.beta)  # self.beta is calculate from fit function
        mu=np.exp(eta)  # getting from the link function
        return mu
```

(1) def __init__ (self, x, y):

It uses ‘super( ).__init__(x,y)’ to inherit all from super class GLM.

(2) def mle (self, beta, x, y): **(#solutions to 1-b-‘Inheritance’)**

Three distributions have its own way to get the loglikelihood value, so here it overrides or rewrites the mle method belonging to super class GLM.

(3) def predict (self, x):

Three distributions have its own link function to get the estimator. Three subclasses inherit and implement fit method from super class to get β_i estimators. Utilizing the β_i and its own predict method to get the estimation of μ. Thus, it overrides or rewrites the predict method in the super class GLM. **(#solutions to 1-b-‘Overriding’)**

Base on inheritance and overriding, it shows polymorphism. **(#solutions to 1-b-‘Polymorphism’)**

1.5 Public interface (#solutions to 1-c-‘Testing codes’ and 1-d-‘User Interface’)

Public interface is separated in ‘main_GLM.py’ because it takes a different role. In this part, firstly, use method in pandas to get the relevant data, which will be used to test later. In the test part, it defines three instances of three subclasses respectively.
Secondly, define main function to implement methods in argparse library. It will show information or description of the program.

1.6 Results (#solutions to 1-c-‘Testing results’)

By contrasting the results through GLM.py and statsmodels, it can help to check the code.
Normal Distribution results from GLM.py :

```
Estimated beta are:[10.42636062  0.03226315  0.62372386]
The prediction of Normal Distribution Model is:[64.34634793 64.64744031 69.5297177  60.73305735 69.33613882 67.40044094
 71.43315243 69.78782288 45.66689523 68.08869109 48.36609362 69.09957272
 74.05710047 49.93617282 58.89414891 35.22481436 59.28121571 62.75472964
 39.91351287 70.45453395 37.07453783 33.40744501 38.2897224  22.01909977
 31.75130043 44.74198802 53.11922746 47.01104143 27.35297026 29.32093128
 17.59923051 20.00806057 23.11595582 17.27659904 19.01871803 26.04099624
 23.73967968 15.69579578 12.84600571 21.11573165 15.13659821 18.09390178
 16.06141446 37.51540686 17.69601995]
```

Normal Distribution results from statsmodels :

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.702
Model:                            OLS   Adj. R-squared:                  0.688
Method:                 Least Squares   F-statistic:                     49.55
Date:                Wed, 13 Nov 2024   Prob (F-statistic):           8.88e-12
Time:                        21:36:34   Log-Likelihood:                -179.90
No. Observations:                  45   AIC:                             365.8
Df Residuals:                      42   BIC:                             371.2
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         10.4264      4.164      2.504      0.016       2.024      18.829
x1             0.0323      0.132      0.244      0.808      -0.234       0.299
x2             0.6237      0.125      5.003      0.000       0.372       0.875
==============================================================================
Omnibus:                        9.200   Durbin-Watson:                   2.053
Prob(Omnibus):                  0.010   Jarque-Bera (JB):               21.265
Skew:                           0.075   Prob(JB):                     2.41e-05
Kurtosis:                       6.364   Cond. No.                         168.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[64.34634847 64.64744085 69.52971826 60.7330579  69.33613936 67.40044148
 71.43315299 69.78782345 45.66689577 68.08869164 48.36609415 69.09957329
 74.05710103 49.93617336 58.89414946 35.22481482 59.2812162  62.75473015
 39.91351334 70.45453449 37.07453834 33.4074455  38.28972291 22.01910025
 31.75130088 44.74198849 53.11922792 47.0110419  27.3529707  29.32093173
 17.59923096 20.00806099 23.11595626 17.27659947 19.01871846 26.04099668
 23.73968013 15.69579622 12.84600614 21.11573208 15.13659866 18.09390222
 16.06141489 37.51540734 17.6960204 ]
```

Bernoulli Distribution results from GLM.py :

```
Estimated beta are:[-13.02134202   2.82611292   0.09515741   2.37868728]
The prediction of Bernoulli Distribution Model is:[0.02657801 0.05950127 0.18725991 0.02590171 0.56989316 0.03485832
 0.02650409 0.05155902 0.11112661 0.69351106 0.02447039 0.18999746
 0.3222396  0.19321111 0.36098979 0.03018378 0.05362638 0.03858837
 0.58987242 0.66078566 0.06137583 0.90484715 0.2417728  0.85209077
 0.83829045 0.48113269 0.63542089 0.30721851 0.84170418 0.94534025
 0.52911725 0.11103089]
```

Bernoulli Distribution results from statsmodels :

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   32
Model:                            GLM   Df Residuals:                       28
Model Family:                Binomial   Df Model:                            3
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -12.890
Date:                Wed, 13 Nov 2024   Deviance:                       25.779
Time:                        22:50:11   Pearson chi2:                     27.3
No. Iterations:                     5   Pseudo R-squ. (CS):             0.3821
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        -13.0213      4.931     -2.641      0.008     -22.686      -3.356
x1             2.8261      1.263      2.238      0.025       0.351       5.301
x2             0.0952      0.142      0.672      0.501      -0.182       0.373
x3             2.3787      1.065      2.234      0.025       0.292       4.465
==============================================================================
[0.02657799 0.05950126 0.18725993 0.02590164 0.56989295 0.03485827
 0.02650406 0.051559   0.11112666 0.69351131 0.02447037 0.18999744
 0.32223955 0.19321116 0.36098992 0.03018375 0.05362641 0.03858834
 0.58987249 0.66078584 0.06137585 0.90484727 0.24177245 0.85209089
 0.83829051 0.48113304 0.63542059 0.30721866 0.84170413 0.94534025
 0.5291172  0.11103084]
```

Poisson Distribution results from GLM.py :

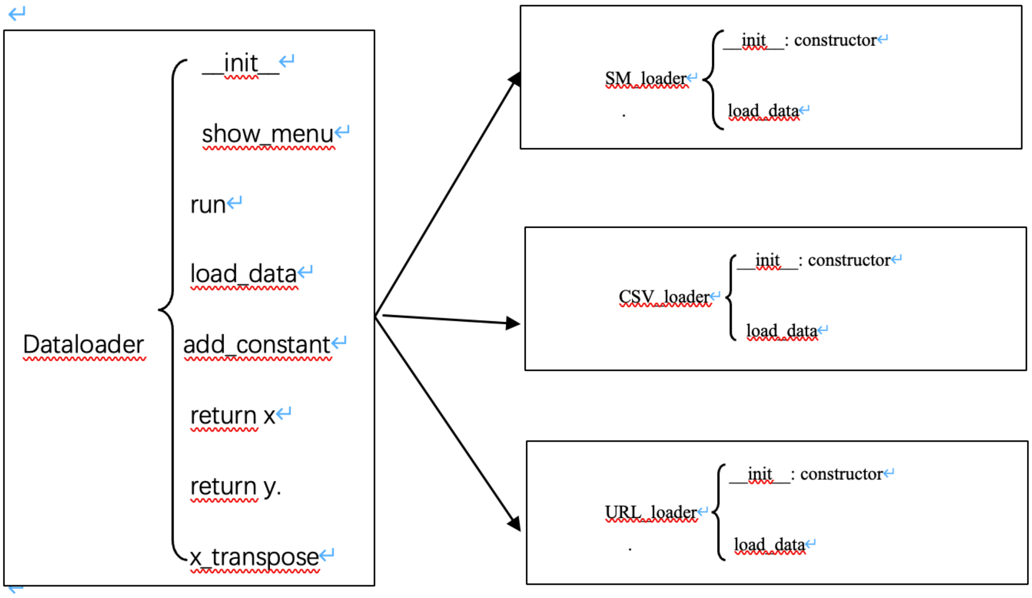
```
Estimated beta are:[ 3.67653953 -0.20598844 -0.26455303]
The prediction of Poisson Distribution Model is:[39.50943593 39.50943593 39.50943593 39.50943593 39.50943593 39.50943593
 39.50943593 39.50943593 39.50943593 30.3254259  30.3254259  30.3254259
 30.3254259  30.3254259  30.3254259  30.3254259  30.3254259  30.3254259
 23.27624869 23.27624869 23.27624869 23.27624869 23.27624869 23.27624869
 23.27624869 23.27624869 23.27624869 32.15445738 32.15445738 32.15445738
 32.15445738 32.15445738 32.15445738 32.15445738 32.15445738 32.15445738
 24.68011987 24.68011987 24.68011987 24.68011987 24.68011987 24.68011987
 24.68011987 24.68011987 24.68011987 18.94319999 18.94319999 18.94319999
 18.94319999 18.94319999 18.94319999 18.94319999 18.94319999 18.94319999]
```

Poisson Distribution results from statsmodels :

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   54
Model:                            GLM   Df Residuals:                       51
Model Family:                 Poisson   Df Model:                            2
Link Function:                    Log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -243.15
Date:                Wed, 13 Nov 2024   Deviance:                       211.63
Time:                        22:52:12   Pearson chi2:                     214.
No. Iterations:                     4   Pseudo R-squ. (CS):             0.7956
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.6765      0.043     84.572      0.000       3.591       3.762
x1            -0.2060      0.052     -3.994      0.000      -0.307      -0.105
x2            -0.2646      0.032     -8.277      0.000      -0.327      -0.202
==============================================================================
[39.50943607 39.50943607 39.50943607 39.50943607 39.50943607 39.50943607
 39.50943607 39.50943607 39.50943607 30.32542611 30.32542611 30.32542611
 30.32542611 30.32542611 30.32542611 30.32542611 30.32542611 30.32542611
 23.27624893 23.27624893 23.27624893 23.27624893 23.27624893 23.27624893
 23.27624893 23.27624893 23.27624893 32.15445751 32.15445751 32.15445751
 32.15445751 32.15445751 32.15445751 32.15445751 32.15445751 32.15445751
 24.68012006 24.68012006 24.68012006 24.68012006 24.68012006 24.68012006
 24.68012006 24.68012006 24.68012006 18.9432002  18.9432002  18.9432002
 18.9432002  18.9432002  18.9432002  18.9432002  18.9432002  18.9432002 ]
```

2. Data_Loader

2.1 Hierarchy map



2.2 Super class: Dataloader (#solutions to 2-a-'Superclass') (#solutions to 2-b-'methods')

(1) def __init__(self):
Define a constructor and set initial data, x value and y value to be none.

(2) @staticmethod: **(#solutions to 2-b-'method'-'decorator')**
To use this code in terminal clearly, a decoration staticmethod is used here. The content of staticmethod is only to show the functions in the public interface, because it does not rely on any class or instance variables and methods. Setting staticmethod makes the codes more clearly and no error.

(3) def run(self):
This function matches the staticmethod and it activates corresponding function after users input the function number in public interface. In the first function load_data, input a parameter of 'data_info', which will be a local csv file path , a csv file in statsmodels, or a csv file url address respectively in the subclasses.

(4) def load_data(self): use an abstract method here and subclass will specify it.

(5) def add_constant(self): this method is to add an one value column to x matrix.

(6) def return_x(self): return the assumed x matrix.

(7) def return_y(self): return the assumed y value.

(8) def x_transpose(self): transpose the x matrix and return its value.

2.3 Subclass: SM_loader **(#solutions to 2-a-'subclass')**

(1) def __init__ (self): in the constructor, use super method to inherit all from super class.

(2) def load_data (self, data_info)[1]: load data from statsmodels built-in csv file, so it overrides the load_data method in super class. In this method, use an exception code as required. Moreover, it will load x values and y values, which will be used in the following return_x and return y methods. **(#solutions to 2-b-'methods'-'exceptions')**

2.4 Subclass: CSV_loader **(#solutions to 2-a-'subclass')**

(1) def __init__ (self): in the constructor, use super method to inherit all from super class.

(2) def load_data (self, data_info): load data from local csv file, and the data_info parameter is the file name or file path. It overrides the load_data in super class as well. It sets an exception, loads x values and y values too. **(#solutions to 2-b-'methods'-'exceptions')**

2.4 Subclass: URL_loader **(#solutions to 2-a-'subclass')**

(1) def __init__ (self): in the constructor, use super method to inherit all from super class.

(2) def load_data (self, data_info): load data from a url address, and the data_info parameter is the address. It overrides the load_data in super class as well. It sets an exception, loads x values and y values too. **(#solutions to 2-b-'methods'-'exceptions')**

2.5 Public interface

Define a main function to add argument of description and other relevant information. It creates three instances of the three subclasses and use the known datasets to implement these instances.

---

[1] Due to my reinstallation of my mac os, the new python3 cannot go through statsmodels and url address. The error shows it is certificate problem. I solve the problem by importing ssl and add a line code: ssl._create_default_https_context = ssl._create_unverified_context.

Reference: https://blog.csdn.net/doudou19930614/article/details/82824406