IE343: Statistical Machine Learning

Final Porject (Day: June 22, 2020)

Professor: Seyoung Yoon *TA : SeongYoon Kim*

1 Notice

(i) Deadline : 11 :59(PM), Tue, July 7, 2020

(ii) Final project has a large portion(25 percent) of total grading.
So, I will answer only to minor question(vague meaning at the pdf file, model description at the preliminary).
I'll not give support to implement source codes. See carefully the module(App/rbf.py).

(iii) You can complete the final project without any additional modules.

(iv) If you understand the assignment 4 and midterm project well, it's not hard to implement.

(v) Cheating is not allowed. Both those who showed and those who copied will get zero points.

(vi) Ask the question via Classum. I will not answer to the question via another way.

(vii) Handwriting is allowed for complex numerical expression. But if you write illegibly, that illegibly part will be not graded.

(viii) You can write your report within 5 pages.

2 Preliminary and Binary Kernel Logistic Regression

We implemented the logistic regression in the binary and multiple labels classification at the midterm project. In the final project, we will extend the logistic regression to the kernel logistic regression.
Let's N and d denote the total number of the training data points and dimension of each data input's dimension, respectively.
And let $\{(X^{(i)}, y^{(i)})\}_{i=1,2,\cdots,N}$ be the training data sets. At the binary logistic regression, we train the parameter θ where the model was

$$\mathbb{P}(y=1|X=(X_1,X_2,\cdots X_d)) = \frac{1}{1+exp(<\theta, concatenate(1,X)>} = \frac{1}{1+exp(\sum_{i=1}^{d}\theta_i X_i + \theta_0)}. \tag{2.1}$$

Here, $concatenate(1,X) := (1, X_1, X_2, \cdots X_d)$ and $< \cdot, \cdot >$ indicates the inner product.

Let's review the SVM. To escape the non-linear separable situation, the mapping function ϕ was defined which increase the dimension of the data's input space. For the hyper-plane $\beta_0 + <\vec{\beta}, \phi(X)>$, the optimal $\vec{\beta}^* = \sum_{j=1}^{N}\alpha_j y^{(j)}\phi(X^{(j)})$.
From that $\vec{\beta}^*$, we obtain the optimal dual variable $\{\alpha_j{}^*\}_{j=1,2,\cdots,N}$ by solving quadratic programming.
And from that mapping function ϕ, the kernel function K was defined by

$$K(X^{(i)}, X^{(j)}) =< \phi(X^{(i)}), \phi(X^{(j)}) > . \tag{2.2}$$

Now, let's deal with the kernel logistic regression from these backgrounds. For binary kernel logistic regression, the model can be constructed by

$$\begin{aligned}
\mathbb{P}(y=1|X=(X_1,X_2,\cdots X_d)) &= \frac{1}{1+exp(<(\beta_0,\vec{\beta}), concatenate(1,\phi(X))>} && (2.3)\\
&= \frac{1}{1+exp(<(\beta_0,\sum_{j=1}^{N}\alpha_j y^{(j)}\phi(X^{(j)})), concatenate(1,\phi(X))>} && (2.4)\\
&= \frac{1}{1+exp(\sum_{j=1}^{N}\alpha_j y^{(j)} < \phi(X^{(j)}), \phi(X) > + \alpha_0)} && (2.5)\\
&= \frac{1}{1+exp(\sum_{j=1}^{N}\alpha_j y^{(j)} K(\phi(X^{(j)}), \phi(X)) + \alpha_0)}. && (2.6)
\end{aligned}$$

(2.4) was derived by using $\vec{\beta}^* = \sum_{j=1}^{N} \alpha_j y^{(j)} \phi(X^{(j)})$. And at the (2.5), we redefine the β_0 to α_0.
In short, you need to implement the binary kernel logistic regression model

$$\mathbb{P}(y = 1|X = (X_1, X_2, \cdots X_d)) = \frac{1}{1 + exp(\sum_{j=1}^{N} \alpha_j y^{(j)} K(\phi(X^{(j)}), \phi(X)) + \alpha_0)} \tag{2.7}$$

by training parameter α.

3 TODO for Code Implementation

There are 2 tasks for the code implementation. '`task 1`' and '`task2`' are for binary logistic regression and binary kernel logistic regression, respectively. '`task1`' and '`task2`' files are attached and they are almost same from midterm project's '`mid_task2_titanic`' folder. The different things is that I provide additional files '`./App/Pre_processing/kernel.py`' and '`./App/rbf.py`'. These are from the assignment 4 and will be helpful for the final project. Unlike midterm project, I already filled the '`getData()`' method in the '`.\main.py`'. And you need to complete followings for the code implementation.

(1) (task1)
It is similar to the midterm project. You need to fill out the '`./App/logistic_regressor.py`'. The code should be implemented in the minus-log-likelihood loss function, `epoch_num=5000` and `lr=0.00005` settings. And '`main.py`' needs to print the 50 train accuracy results(print the test accuracy for every 100 epoch) and one test accuracy. Lastly, you need to use the bias term θ_0 of (2.1).

(2) (task2)
You need to fill out the '`./App/logistic_regressor.py`' and '`./main.py`' . The code should be implemented in the minus-log-likelihood loss function, RBF kernels, `epoch_num=5000` and `lr=0.005` settings. RBF kernels should use the hyperparameter (1,1,1,1,...1). And '`main.py`' needs to print the 50 train accuracy results(print the test accuracy for every 100 epoch) and one test accuracy. Lastly, you need to use the bias term α_0 of (2.7).

4 TODO for Report

(1) Write the kernel logistic regression model and logistic regression model for the binary labels data. For those models, which parameter we have to train ? State the parameter's dimension precisely. **(5 pts)**

(2) What's the needs of the kernel logistic regression ? i.e. from kernel logistic regression, what we can recover the issue that original logistic regression can't solve and state the reason precisely. **(10 pts)**

(3) State the advantages and disadvantages of kernel logistic regression and original logistic regression. **(5 pts)**

(4) State how the weight update should happen for the original logistic regression and kernel logistic regression at the binary version. You need to write precisely how the gradient is induced also. **(5 pts)**

(5) Construct the model for the original logistic regression and kernel logistic regression at the multiple labels(K : number of labels > 2). You need to clarify the parameters. **(10 pts)**

(6) For the model you construct at the (5) and minus-log-likelihood loss function, write the pseudo code about each weight update situation. You also need to state the reason and input value in the method function precisely.**(10 pts)**

(7) Describe the method about your logistic regression model with the code at the task1 line by line.**(10 pts)**

(8) Describe the method about your kernel logistic regression model and `main.py` with the code at the task2 line by line.**(15 pts)**

5 Grading Criteria

You have to submit your entire completed python files and a report with containing the answers to (1)-(8) of the section 4.

Code implementation about task 1 will be 10 points.
Code implementation about task 2 will be 20 points.

Thus, total grade will be 100 points.

Deadline : 11 :59(PM), Tue, July 7, 2020

You can write your report within 5 pages.