# 統計學習初論（105-2）

# 作業三 (Part B)

## 作業設計: 盧信銘

## 國立台灣大學資管系

## 截止時間: 2017年4月25日上午9點

第二題請至RSAND上批改，第(a)小題範例命令為: sl\_check\_hw3q2a ./your\_program，第(b)小題範例命令為: sl\_check\_hw3q2b ./your\_program。作業自己做。嚴禁抄襲。不接受紙本繳交，不接受遲交。請以英文或中文作答。

### 第二題

(40 points) The goal of sentence categorization is to assign a sentence to one of two or more predefined categories. We are going to develop a sentence classification model using regularized logistic regression. The underlying problem is to detect sentences that mentioned about cost reduction efforts during the current fiscal year. We denoted this category as “O.COST.” A sentence can belong to O.COST, or not belongs to this category.

As an illustrative example, the following sentences belong to O.COST:

* The Company expects the operational restructuring plan to be fully implemented by June 30, 2006, and the implementation of the operational restructuring plan will result in severance related and other charges of approximately $14.6 million.
* Operational Restructuring On August 19, 2005, the Company announced that it had taken the initial step in implementing an expense restructuring plan, necessitated by the Company’s declining revenue trend over the previous two and one-half years.

The following sentences are examples not belonging to O.COST:

* Such capital expenditures and employee compensation costs will continue to be incurred in advance of the first revenues to be earned from the contract, expected later in 2006.
* The Company expects many clients to continue to use their own internal recovery audit functions as a substitute for its recovery audit services.

The o\_cost\_train.rdata file contains a data frame named ds4a\_train. Each row represents a sentence. To save your time, all sentences have been process via natural language processing approaches and have been converted to appropriate representations. The first column (named o.cost) is the label of the sentence. The value is either “pos” or “neg.” The “pos” string indicates that this sentence belong to the O.COST category. There are three types of feature representations in this dataset. We summarize the information below:

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Column Position | Description | Value |
| f\_past | 2 | Sentence timing | 0 or 1 |
| g1 to g100 | 3-102 | Latent topics | 0 to 1 |
| w\* | 103-3102 | Unigram (word feature) | 0 or 1 |

The specific feature representations used in this dataset is beyond the scope of this course. You only need to know the column positions of the three types of features in order to complete the following two tasks.

1. (30%) Write a function named logicreg\_l2\_train to perform model training. This function should take the following input parameters (in this order):
   1. y: the (one-column) matrix of outcome value. The value should be either 1 or 0, representing positive and negative cases.
   2. xmat: the matrix containing feature values. Should be compatible with y.
   3. lambda\_rate: the coefficient to compute initial lambda value. The default value is 0.0005.
   4. param\_tol: the tolerance of error. The default value is .
   5. granditertol: the tolerance of error for overall convergence, measured by number of inner iteration. The default value is 2.
   6. outitermax: the maximum number of outer iteration. The default value is 50.
   7. inneritermax: the maximum number of inner iteration. The default value is 20.
   8. debuglevel: whether to output debug information. The default value is 0. You should not output any debug information when the value is 0. Set this value to a positive integer if you want to output debugging information when developing the function.

The function should start by computing the initial value for subsequent numerical optimization procedure. Set initial , where is the number of training instance, and we can compute the initial by:

We are ready to iterate to update and . The outer iteration can loop for a maximum of *outitermax* times. In each outer iteration, we update using Newton’s Method, and then update through the algorithm presented in class. We called the updating of and the “inner iteration.”

To update , compute the gradient and hessian matrix according to our discussion in class (cf. slides 25, 05c linear model for classification.pdf), and then compute the new via . Declare convergence if the mean absolute difference between the new and old is less than param\_tol. This inner iteration should loop for a maximum of inneritermax times.

To update , follow our discussion in class to compute the new given the new (cf. slide 30, 05c linear model for classification.pdf). Declare convergence if the mean absolute difference between the new and old is less than param\_tol. This inner iteration should loop for a maximum of inneritermax times.

After updating and , continue the outer iteration if the inner iteration for takes more than 2 iterations to converge. Otherwise, break the outer iteration and return a list that contain the parameters of the trained model.

The returned list should include the following components (in this order):

* w: the estimated coefficients
* w\_sd: the standard deviation of the estimated coefficients. Recall that the posterior of is approximately normal with a covariance , and . This component (w\_sd) is simply the square root of the diagonal elements of .
* lambda: The lambda coefficient.
* M: the dimension of input features (i.e., the length of w).
* N: number of training instances.

Sample input and output

|  |
| --- |
| **>** load**(**file**=**"o\_cost\_train.rdata"**)**  #Sample 1  **>** dm\_train\_t **=** as.numeric**(**ds4a\_train**[**,1**]** **==** "pos"**)**  **>** tall**=**as.matrix**(**dm\_train\_t**)**  **>** xmat **=** model.matrix**(~**f\_past**+**g1**+**g2**+**g3**+**g4**+**g5**+**g6**+**g7**+**g8**+**g9**+**g10, data**=**ds4a\_train**[**,**-**1**])**  **>** model1 **=** logicreg\_l2\_train**(**tall, xmat, debuglevel**=**0**)**  **>** model1  **$**w  **[**,1**]**  **(**Intercept**)** **-**1.4053791  f\_past **-**2.1991748  g1 0.8093526  g2 **-**8.5513319  g3 **-**6.9447606  g4 4.5614423  g5 0.3788263  g6 2.4862296  g7 **-**4.8258687  g8 **-**4.5343458  g9 **-**9.7827331  g10 **-**1.2796956  **$**w\_sd  **(**Intercept**)** f\_past g1 g2 g3 g4  0.04366356 0.23233368 0.65868466 2.45398092 2.02103128 0.49332275  g5 g6 g7 g8 g9 g10  0.44533757 0.72333805 1.60224100 1.41466936 2.48016361 1.25627118  **$**lambda  **[**1**]** 0.03738331  **$**M  **[**1**]** 12  **$**N  **[**1**]** 6106  #Sample 2  **>** dm\_train\_t **=** as.numeric**(**ds4a\_train**[**,1**]** **==** "pos"**)**  **>** tall**=**as.matrix**(**dm\_train\_t**)**  **>** xmat2 **=** model.matrix**(~**., data**=**ds4a\_train**[**,c**(**2, 103**:**204**)])**  **>** model2 **=** logicreg\_l2\_train**(**tall, xmat2, debuglevel**=**0**)**  **>** print**(**head**(**model2**$**w, n**=**10**))**  **[**,1**]**  **(**Intercept**)** **-**3.609966  f\_past **-**3.251449  w4035\_reduction 11.494339  w4037\_restructure 9.834049  w4078\_cost 6.479697  w3998\_employee 8.264094  w4111\_costs 2.953915  w3492\_savings 11.699770  w3755\_severance 10.435952  w4028\_reduce 10.774457  **>** print**(**head**(**model2**$**w\_sd, n**=**10**))**  **(**Intercept**)** f\_past w4035\_reduction w4037\_restructure  0.1133109 0.4242634 1.0368524 0.8800527  w4078\_cost w3998\_employee w4111\_costs w3492\_savings  1.1815602 1.0729767 1.0726329 1.2645677  w3755\_severance w4028\_reduce  1.1358556 1.0819511  **>** model2**$**lambda  **[**1**]** 0.03569337  **>** model2**$**M  **[**1**]** 104  **>** model2**$**N  **[**1**]** 6106 |

Evaluation: All credits will be given based on the correctness of 10 testing cases. Correct output in a case is worth 3 points.

1. (10%) Write a function named logicreg\_l2\_predict to perform prediction on a trained model. Use the learned parameter to perform prediction. This function takes the following parameters (in this order):
   1. model1: The learned model.
   2. xmat: the feature matrix to be used for prediction.

This function should return a list that contains the following component (in this order):

* prob: a vector of predicted probability for each sentence.
* class: a vector of predicted class (0 or 1) for each sentence.

Sample input and output:

|  |
| --- |
| #Sample 1  **>** load**(**file**=**"o\_cost\_train.rdata"**)**  **>** load**(**file**=**"o\_cost\_test.rdata"**)**  **>**  **>** dm\_train\_t **=** as.numeric**(**ds4a\_train**[**,1**]** **==** "pos"**)**  **>** tall**=**as.matrix**(**dm\_train\_t**)**  **>** xmat **=** model.matrix**(~**f\_past**+**g1**+**g2**+**g3**+**g4**+**g5**+**g6**+**g7**+**g8**+**g9**+**g10, data**=**ds4a\_train**[**,**-**1**])**  **>** model1 **=** logicreg\_l2\_train**(**tall, xmat, debuglevel**=**0**)**  **>** #perform prediction  **>** xmattest1 **=** model.matrix**(~**f\_past**+**g1**+**g2**+**g3**+**g4**+**g5**+**g6**+**g7**+**g8**+**g9**+**g10, data**=**ds4a\_train**[**,**-**1**])**  **>** logicpred1 **=** logicreg\_l2\_predict**(**model1, xmattest1**)**  **>** head**(**logicpred1**$**class,n**=**25**)**  **[**1**]** 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  **>** head**(**logicpred1**$**prob,n**=**25**)**  **[**,1**]**  126 0.2448322  234 0.2116120  235 0.2054513  236 0.1935176  256 0.2094563  522 0.1918210  643 0.1883066  645 0.1889833  662 0.1975938  712 0.1893408  739 0.2117428  776 0.1858589  894 0.1851180  897 0.1813298  898 0.1780135  899 0.1673133  2360 0.1702156  2361 0.2426719  2397 0.1872902  2442 0.2024119  2457 0.1956855  2458 0.1745073  2510 0.2607178  2513 0.1967121  2514 0.1066816  #Sample 2  **>** dm\_train\_t **=** as.numeric**(**ds4a\_train**[**,1**]** **==** "pos"**)**  **>** tall**=**as.matrix**(**dm\_train\_t**)**  **>** xmat2 **=** model.matrix**(~**., data**=**ds4a\_train**[**,c**(**2, 103**:**204**)])**  **>** model2 **=** logicreg\_l2\_train**(**tall, xmat2, debuglevel**=**0**)**  **>** #perform prediction  **>** xmattest2 **=** model.matrix**(~**., data**=**ds4a\_test**[**,c**(**2, 103**:**204**)])**  **>** logicpred2 **=** logicreg\_l2\_predict**(**model2, xmattest2**)**  **>** head**(**logicpred2**$**class,n**=**30**)**  **[**1**]** 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0  **>** head**(**logicpred2**$**prob,n**=**30**)**  **[**,1**]**  266 0.99441301  644 0.99508324  815 0.04801758  2509 0.98630634  2511 0.96278682  2523 0.99776405  3168 0.99999183  3715 0.99990698  3716 0.36628387  3765 0.60906632  4286 0.97563708  4709 0.63226300  4895 0.91296762  4937 0.97449792  5989 0.99996757  6091 0.99982256  6154 0.96100325  6165 0.97107921  6174 0.96185452  6293 0.07284124  6384 0.99954049  6410 0.99994020  6529 0.95066773  6546 0.99994284  6551 0.99418181  6560 0.98777522  6661 0.99868930  6662 0.99916273  6665 0.99481762  7911 0.34222445 |

Evaluation: All credits will be given based on the correctness of 10 testing cases. Correct output in a case is worth 1 points.