

# README for Multi-task Logistic Regression for Survival Prediction Code

Chun-Nam Yu

March 1, 2012

## 1 Compilation

The code should compile on Linux without any modification. To compile, just type 'make'. There should be 2 executables after compilation, `mtlr_train` and `mtlr_test`.

## 2 Training

### 2.1 Input Format

Each line in the input file represents a patient. The attributes of each patient are separated by space ' '. The first value of each line should be the survival time or censoring time of the patient. The second value of each line should be a binary value indicating whether the observation is censored (1 for censored, 0 for uncensored). The rest of the line contains covariates in a sparse vector format, each non-zero is specified by ' $i : v$ ', where  $i$  is the index of the non-zero entry and  $v$  is the value of that entry. For example, the line

20 1 3:20.5 5:1 7:4.3 8:4

represents a patient censored at time 20, with a covariate vector  $\{3 : 20.5, 5 : 1, 7 : 4.3, 8 : 4\}$ .

### 2.2 Changes compared to formulation specified in the paper

Compared to the original formulation specified in [Yu/etal/11],

$$\min_{\Theta} \frac{C_1}{2} \sum_{j=1}^m \|\vec{\theta}_j\|^2 + \frac{C_2}{2} \sum_{j=1}^{m-1} \|\vec{\theta}_{j+1} - \vec{\theta}_j\|^2 - \sum_{i=1}^n \left[ \sum_{j=1}^m y_j(s_i) (\vec{\theta}_j \cdot \vec{x}_i + b_j) - \log \sum_{k=0}^m \exp f_{\Theta}(\vec{x}_i, k) \right] \quad (1)$$

the objective optimized in the training code uses an equivalent but rescaled  $C_1$ :

$$\min_{\Theta} \frac{1}{2} \sum_{j=1}^m \|\vec{\theta}_j\|^2 + \frac{C_2}{2} \sum_{j=1}^{m-1} \|\vec{\theta}_{j+1} - \vec{\theta}_j\|^2 - \frac{C_1}{n} \sum_{i=1}^n \left[ \sum_{j=1}^m y_j(s_i) (\vec{\theta}_j \cdot \vec{x}_i + b_j) - \log \sum_{k=0}^m \exp f_{\Theta}(\vec{x}_i, k) \right]. \quad (2)$$

The definition of  $C_1$  below will follow the rescaled version.

#### 2.2.1 File for specifying time points

The file for specifying the time points  $t_j$  should contain a list of numerical values sorted in ascending order, separated by a newline. Each line contains one time point. For example, to specify the grid  $t_1 = 1$ ,  $t_2 = 2.5$ ,  $t_3 = 3.3$ , we can supply a file of the following format:

1  
2.5  
3.3

## 2.3 Example Usage

To train a model with the sample data with parameter  $C_1 = 10$ ,  $C_2 = 1$  and output the model file to `example.model` you can type

```
./mtlr_train -c 10 -d 1 -m 60 -i example.train -o example.model
```

This uses a regular grid of 60 time points from 1 to 60. To use your own time points file, you can type

```
./mtlr_train -c 10 -d 1 -m 100 -q example.intervals -i example.train -o example.model
```

Note that the value supplied by ‘-m’ has to match the number of time points in the file specified by ‘-q’.

The above commands treat all patients as *uncensored* while training. To make use of the censoring status, specify `-u 1` to trigger EM training for censored data:

```
./mtlr_train -c 10 -d 1 -u 1 -m 100 -q example.intervals -i example.train  
-o example_censored.model
```

While using the EM training for censored data, it’s usually a good idea to initialize the parameters to avoid getting to bad local minima. For example, we could initialize the training with the model file `example.model` we obtained earlier when treating all patients as uncensored with the ‘-w’ option:

```
./mtlr_train -c 10 -d 1 -u 0 -m 100 -q example.intervals -w example.model -i example.train  
-o example_censored.model
```

## 2.4 Command Line Options

- ‘-c’: specify regularization constant  $C_1$  (default: 1)
- ‘-d’: specify regularization constant  $C_2$  (default: 1)
- ‘-i’: specify training input filename
- ‘-o’: specify output model filename
- ‘-m’: specify the number of time points (default: 60)
- ‘-w’: specify the weight (model) filename used to initialize EM training for censored targets
- ‘-u’: specifying value ‘1’ treats all input examples as ‘uncensored’ during training; ‘0’ starts EM training (default: 1)
- ‘-q’: specify the file containing the set of time points for discretization. For input file format see the Section 2.2.1 above. Note that the number of grid points have to match the number specified by ‘-m’. (default: equally spaced grid from 1 up to the value specified by ‘-m’)

## 3 Testing

### 3.1 Input Format

The input file format for testing is the same as the file format for input training data.

## 3.2 Example Usage

### 3.2.1 Point Prediction

To make point predictions optimizing for specific loss functions, say the L1-loss, with the model `example.model` learned earlier on test data `example.test`, you can type:

```
./mtlr_test -l l1 -m 100 -q examples.interval -i example.test -o example.model
```

This will output a list of survival time prediction, one for each patient per line. The option for ‘-m’ and ‘-q’ specified should be the same as the values used during training. At the end of the output there will also be a summary of the average loss of the predictions.

### 3.2.2 Classification

To make binary classification of whether a patient would survive for longer than time  $s$ , you can type

```
./mtlr_test -l class -t s -m 100 -q examples.interval -i example.test -o example.model
```

The option ‘-t’ specifies the threshold for classification. This will output a pair of values ‘ $y : p$ ’ per line for each patient, where  $y$  is the survival status prediction and  $p$  is the predicted probability of survival at time  $t$ . The summary average accuracy reported at the end ignores censored patients with censoring time less than that specified by ‘-t’ (survival status essentially unknown).

### 3.2.3 Printing the survival distribution

You can print the survival distribution for each patient by using the ‘-p’ option:

```
./mtlr_test -p -m 100 -q examples.interval -i example.test -o example.model
```

In the output each line will be a vector of probabilities, representing the survival distribution of one patient. The vector of probabilities will be appended to classification output or survival time prediction above for each line. The  $j$ th entry represent the predicted survival probability for that patient at  $t_j$ .

## 3.3 Command Line Options

- ‘-i’: specify test input filename
- ‘-o’: specify learned model filename
- ‘-m’: specify the number of time points (default: 60)
- ‘-t’: specify threshold for classification of survival status (default: 30)
- ‘-l’: specify the type of loss to optimize when making point predictions for survival time. Available options are ‘l1’, ‘l2’, ‘l1-log’, ‘l2-log’, ‘rae’, ‘class’ (default: ‘l1’)
- ‘-p’: print survival distribution, i.e., a vector of survival probability at each time point (no argument needed)
- ‘-u’: specifying value ‘1’ treats all input examples as ‘uncensored’ during evaluation; specifying ‘0’ truncates the loss at 0 if the predicted survival time is greater than the censoring time for a censored patient (default: 1)
- ‘-q’: specify the file containing the set of time points for discretization. For input file format see the Section 2.2.1 above. Note that the number of grid points have to match the number specified by ‘-m’, and also the same as the model file.

## 4 References

[Yu/etal/11] C.-N. Yu, R. Greiner, H.-C. Lin, V. Baracos, Learning Patient-Specific Cancer Survival Distributions as a Sequence of Dependent Regressors, *NIPS 2011*