

Constraint-free Implementation of LRR

Hongning Wang
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana IL, 61801 USA
wang296@illinois.edu

1 Overview

In this constraint-free implementation of latent rating regression (LRR) [Wang et al., 2010], auxiliary variables are introduced to get rid of constraints in the original LRR model, where aspect weights α should be positive and sum up to one. The new implementation performs closely as the original LRR model and can be used as a replacement of it.

2 Latent Rating Regression Model

In the work of Latent Aspect Rating Analysis (LARA) [Wang et al., 2010], we assume in each review: 1) the overall rating is the weighted sum of the individual aspect ratings; 2) the aspect ratings can be predicted by the words associated with each aspect. Intuitively, we can formalize these assumptions as follows:

$$p(t_d|\mu, \Sigma, \delta, \sigma) = \int p(\alpha_d|\mu, \Sigma)p(r_d|\alpha_d^T \mathbf{S}_d, \delta)d\alpha_d$$

where r_d is the overall rating for review d , $\mathbf{S}_d = \{S_{d1}, S_{d2}, \dots, S_{dk}\}$ are the predicted ratings for each aspect where $S_{di} = \beta_i^T \mathbf{w}_i$, and α_d is the corresponding inferred aspect weight; δ is the standard deviation of overall rating prediction. In the original LRR model, we require $\forall i, \alpha_{di} \geq 0$ and $\sum_{i=1}^k \alpha_{di} = 1$.

The constraint on α_d greatly increases the computational complexity. To avoid solving a constraint optimization problem, we introduce a set of auxiliary variables $\{\hat{\alpha}_{d1}, \hat{\alpha}_{d2}, \dots, \hat{\alpha}_{dk}\}$ for each review d , and set

$$\alpha_{di} = \frac{\exp(\hat{\alpha}_{di})}{\sum_{j=1}^k \exp(\hat{\alpha}_{dj})}. \quad (1)$$

Furthermore, we will assume $\hat{\alpha}_{di}$ is drawn from $N(\mu, \Sigma)$ rather than α_{di} as in the original LRR model.

Similar trick can be applied on the aspect rating S_{di} to avoid negative predicted ratings:

$$S_{di} = \exp(\beta_i^T \mathbf{w}_i). \quad (2)$$

3 EM Updating Formulas

The complete-data log-likelihood function for the newly derived problem is very similar as that in the original LRR model:

$$L(r_d, \hat{\alpha}_d, S_d, \mu, \Sigma, \sigma^2, \beta) = -\log \sigma^2 - \frac{(\alpha_d^\top S_d - r_d)^2}{\sigma^2} - \log \Sigma - (\hat{\alpha}_d - \mu)^\top \Sigma^{-1} (\hat{\alpha}_d - \mu) - \lambda \beta^\top \beta \quad (3)$$

where α_d is a function of $\hat{\alpha}_d$ as defined in Eq (1).

Hard-EM is performed to iteratively optimize the complete-data log-likelihood over the training corpus. Most procedures are the same as we have derived in [Wang et al., 2010], and we will only list the updating formulas for $\hat{\alpha}$ and β in this manual.

Besides, in order to ensure numerical stability and provide good initialization of the random variables, we add an additional term in the log-likelihood function for each review document d ,

$$L_{aux}(r_d, \alpha_d, S_d) = \pi \sum_{i=1}^k \alpha_{di} (S_{di} - r_d)^2 \quad (4)$$

where π is a predefined confidence parameter to control the influence of the newly introduced term in the log-likelihood function. Intuitive, $L_{aux}(r_d, \hat{\alpha}_d, S_d)$ guides the optimization procedure to estimate a better starting point of $\hat{\alpha}_d$ and β from the overall rating r_d . Empirically, π should be much smaller than $1/\sigma^2$.

3.1 Updating $\hat{\alpha}$

In E-step, $\hat{\alpha}_d$ is inferred for every review document d by maximizing the following objective function:

$$L(\hat{\alpha}_d) = \frac{(\alpha_d^\top S_d - r_d)^2}{\sigma^2} + \pi \sum_{i=1}^k \alpha_{di} (S_{di} - r_d)^2 + (\hat{\alpha}_d - \mu)^\top \Sigma^{-1} (\hat{\alpha}_d - \mu) \quad (5)$$

The gradient with respect to $\hat{\alpha}_{di}$ is,

$$\frac{\partial L(\hat{\alpha}_d)}{\partial \hat{\alpha}_{di}} = \frac{2(\alpha_d^\top S_d - r_d)}{\sigma^2} \frac{\partial \alpha_d^\top S_d}{\partial \hat{\alpha}_i} + \pi \frac{\partial \sum_{j=1}^k \alpha_{dj} (S_{dj} - r_d)^2}{\partial \hat{\alpha}_i} + 2 \sum_{j=1}^k \Sigma_{ij}^{-1} (\hat{\alpha}_{dj} - \mu_j) \quad (6)$$

where

$$\frac{\partial \alpha_d^\top S_d}{\partial \hat{\alpha}_i} = \alpha_{di} \sum_{j=1}^k \left[\delta(i=j) S_{di} (1 - \alpha_{di}) - \delta(i \neq j) S_{dj} \alpha_{dj} \right] \quad (7)$$

and

$$\frac{\partial \sum_{j=1}^k \alpha_{dj} (S_{dj} - r_d)^2}{\partial \hat{\alpha}_i} = \alpha_{di} \sum_{j=1}^k \left[\delta(i=j) (S_{di} - r_d)^2 (1 - \alpha_{di}) - \delta(i \neq j) (S_{dj} - r_d)^2 \alpha_{dj} \right] \quad (8)$$

3.2 Updating β

In M-step, the optimal β is estimated over the whole corpus by maximizing the following objective function:

$$L(\beta) = \sum_d^D \left[\frac{(\alpha_d^T S_d - r_d)^2}{\sigma^2} + \pi \sum_{i=1}^k \alpha_{di} (S_{di} - r_d)^2 \right] + \lambda \beta^T \beta \quad (9)$$

Taking derivative with respect to β_i , we get,

$$\frac{\partial L(\beta)}{\partial \beta_i} = 2 \sum_d^D \alpha_{di} \left[\frac{(\alpha_d^T S_d - r_d)}{\sigma^2} + \pi (S_{di} - r_d) \right] \frac{\partial S_{di}}{\partial \beta_i} + 2\lambda \beta_i \quad (10)$$

where $\frac{\partial S_{di}}{\partial \beta_i} = S_{di} \mathbf{w}_{di}$ according to Eq (2).

4 Package Details

The code implements the keyword-based bootstrapping aspect segmentation and latent rating regression (LRR) model originally published in [Wang et al., 2010]. In particular, the LRR model is slightly changed to avoid solving constrained optimization as described above.

4.1 Dependency

There are two external libraries used in the implementation:

1. OpenNLP: it is used to extract and tokenize the sentences in the review content. Both the lib and trained model files are needed. Latest update of this toolkit can be found in <http://opennlp.apache.org/>.
2. Colt: it is used for the matrix operations in LRR, e.g., matrix inverse. Latest update for this toolkit can be found in <http://acs.lbl.gov/software/colt/>.

In this package, the required files are located at `./libs` and `./Data/Model/NLP`.

4.2 Aspect Segmentation

The keyword-based bootstrapping aspect segmentation is implemented in the same way as in [Wang et al., 2010].

To apply bootstrapping for aspect keyword extraction, call the following methods:

```
Analyzer analyzer = new Analyzer("Data/Seeds/hotel_bootstrapping.dat",
    "Data/Seeds/stopwords.dat", "Data/Model/NLP/en-sent.zip",
    "Data/Model/NLP/en-token.zip", "Data/Model/NLP/en-pos-maxent.bin");
analyzer.LoadDirectory("Data/Reviews/", ".dat");
analyzer.BootStrapping("Data/Seeds/hotel_bootstrapping_sel.dat");
```

where the file “hotel_bootstrapping.dat” is your initial aspect keyword list, one aspect per row and keywords are separated by “0x09.” “Data/Reviews” is the folder contains your target review files with fixed format. And “Data/Seeds/hotel_bootstrapping_sel.dat” is the result aspect keyword list expanded by bootstrapping.

To segment the aspects in review text document and generate the corresponding vector representation of each hotel (i.e., hReviews as defined in [Wang et al., 2010]), call the following methods:

```
Analyzer analyzer = new Analyzer("Data/Seeds/hotel_bootstrapping_sel.dat",
    "Data/Seeds/stopwords.dat", "Data/Model/NLP/en-sent.zip",
    "Data/Model/NLP/en-token.zip", "Data/Model/NLP/en-pos-maxent.bin");
analyzer.LoadDirectory("Data/Reviews/", ".dat");
analyzer.Save2Vectors("Data/Vectors/vector_CHI_4000.dat");
```

Sample codes can be found in “src/aspectSegmenter/Analyzer.java”.

4.3 Latent Rating Regression Model

The LRR model can be simply executed by the following lines:

```
LRR model = new LRR(500, 1e-2, 5000, 1e-2, 2.0);
model.EM_est("Data/Vectors/Vector_CHI_4000.dat", 10, 1e-4);
model.SavePrediction("Data/Results/prediction.dat");
model.SaveModel("Data/Model/model_hotel.dat");
```

In the constructor of LRR model, you need to specify the following parameters: max iteration of α update, α ’s convergence criterion, max iteration of β update, β ’s convergence criterion, and β ’s regularization parameter λ .

Besides, the LRR model can also be initialized previously trained model:

```
LRR model = new LRR(500, 1e-2, 5000, 1e-2, 2.0, "Data/Model/model_hotel.dat");
```

Sample codes can be found in “src/lara/LRR.java”.

4.4 Baselines

In this package, there are also implementations of the baseline methods, SVR-O and SVR-A, as described and compared in [Wang et al., 2010]. The only difference is we used logistic regression rather than SVR in this implementation. These baselines can be executed in a similar way as LRR model:

```
RatingRegression model = new RatingRegression(500, 5e-2, 5000, 1e-4, 1.0);
model.EstimateAspectModel("Data/Vectors/Vector_CHI_4000.dat");
model.SavePrediction("Data/Results/prediction_baseline.dat");
model.SaveModel("Data/Model/model_base_hotel.dat");
```

The parameter “RatingRegression.BY_OVERALL” specifies the choice of SVR-O (true) or SVR-A (false). And the parameters in the constructor of RatingRegression model are the same as in LRR model.

Sample codes can be found in “src/lara/RatingRegression.java”.

4.5 Miscellaneous

1. The trade-off parameter π for controlling the importance of the newly introduced term $L_{aux}(r_d, \alpha_d, S_d)$ (as defined in Eq (4)) can be set in “LRR.PI”. Default value is 0.5.
2. The predicted score mapping can also be achieved by square function, i.e., $S_{di} = (\beta_i^T \mathbf{w}_{di})^2$. This variation is also implemented in the package and can be enabled by the setting of “RatingRegression.SCORE_SQUARE”: *true* for square mapping and *false* for exponential mapping.
3. I have included a parsed vector file for 4000 hotels (i.e., hReviews) for testing purpose. It can be found in the folder of “Data/Vectors.”
4. The corresponding 4000 selected words for building the hReview vectors are stored in the file “Data/Seeds/hotel_vocabulary_CHI.dat,” scripts are provided to list the learned top ranked opinion words under each aspect (“Data/topics.py” and “Data/run.sh”).
5. 21 sample hotel review text documents are included in the folder of “Data/Reviews” for illustrating the required format in the aspect segmentation module.
6. **NOTE:** In the current implementation, the loaded corpus is split into train/test with ratio 0.75, and the performance printed during model training is computed based on the testing corpus. This setting is different from our original setting in [Wang et al., 2010], where we train and test on the same data set for LRR evaluation.

References

- [Wang et al., 2010] Wang, H., Lu, Y., and Zhai, C. (2010). Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 783–792. ACM.