Carleton University

Department of System and Computer Engineering

SYSC 5045

Ubiquitination Detection Using Logistic Regression

Author:
Tingjun Liu
Ye Tao

Supervisor:
Dr. James Green

April 18, 2017

1 Introduction

The purpose of this project is to classify whether the protein is ubiquitinated or not. 20% of data is held as a blind test set. For the rest of the 80% data, 30% is labeled, 70% is unlabeled. Each group can obtain 1000 label from the unlabeled data-set. The goal is to optimize the precision and recall of the classifier. This project was initially embedded using Weka, then the verification and modifications were implemented using Matlab.

2 Data Reprocess

The labeled data-set has 28759 instances and each instance has 435 features. The data-set is class imbalanced where the ratio between positive and negative is $1 : 6$. Missing data is marked as -9999.

2.1 Feature Selection

CFS subset evaluator in Weka searches the features which are highly correlated to the class. Using this algorithm, the number of feature reduces from 435 to 55.

Within the 55 features, the missing data is handled as follows:

- Calculate the percentage of the missing data in each feature.
- If the percentage of the missing data in a feature in both labeled and unlabeled dataset is bigger than 5%, eliminate this feature.

- If the percentage of the missing data in a feature in both dataset is less than 5%, replace the missing data with the mean value of this feature.

Then the feature reduces from 55 to 40. We used 70% of the labeled data as training-set and the rest as validation-set.

3 Classifier Design

3.1 Logistic Regression

Logistic regression classifier is to assign a data x to a binary class c using:

$$P\big(c|x\big) = \frac{1}{1 + e^{-\omega^T x}}. \tag{1}$$

The decision boundary is defined at $P(c = 1|x) > 0.5$ if the data-set is balanced.

3.2 Handling the Class Imbalance: Bagging Embedded with Under-Sampling

Since Weka cannot adjust the threshold, the first challenge is to handle the class imbalance. There are two ways of handling the class imbalance, either oversample the minority class or down-sample the majority class. However the disadvantage of oversample in this data-set is obvious that two class is highly mixed and adding more artificial points will not help to increase

the performance. We tried to balance the training-set to obtain a optimal solution, but the results reduced significantly when applying the validation-set.

Therefore we used down-sampling technique combined with Bagging as the following procedure:

Mistake 1.

1. Randomly choose 10 subset in the majority class and each set has the same amount number of data with the minority class.

2. Using the 10 balanced data-set to train 10 logistic regression classifiers. Each classifiers' decision boundary is at $\omega^T x = 0$.

3. Rank the performance of each classifier using the validation-set. Choose the one with the highest score.

Mistake 2.

By doing so, the classifier optimally used 100% of both minority class and majority class.

3.3 Active Learning

In order to apply active learning, the decision boundary must be determined. After applying Bagging and under-sampling algorithms, a current optimal solution is obtained and the decision boundary is at

$$f(x) = \omega_0^T x. \tag{2}$$

However we cannot simply set the decision boundary at $f(x) = 0$ because the class is imbalanced and prior probability of the minority class is closed to 1/7. The decision boundary is defined at x_0 so that the predicted class could maintains the prior probability of the training-set. In other words, the decision boundary is defined at

$$\begin{aligned} P(c=0|x) &< \sigma(\omega_0^T x_0) \\ P(c=1|x) &> \sigma(\omega_0^T x_0) \\ \frac{no.\ '0'}{no.\ '0' + no.\ '1'} &= prior_{c=0} \end{aligned} \tag{3}$$

By doing so and proven from the results, it will achieve an optimal precision and recall.

Our strategy of picking up the extra data points is as follows:

- 25% at definite positive region. $(x \in \omega_0^T x \approx 0)$
- 25% at definite negative region. $(x \in \omega_0^T x \approx 1)$
- 50% at decision boundary. $(x \in \|x - x_0\| \approx 0)$

For a better illustration, Fig. 1 shows the data selection assuming $\sigma(\omega_0^T x_0) = 0$.

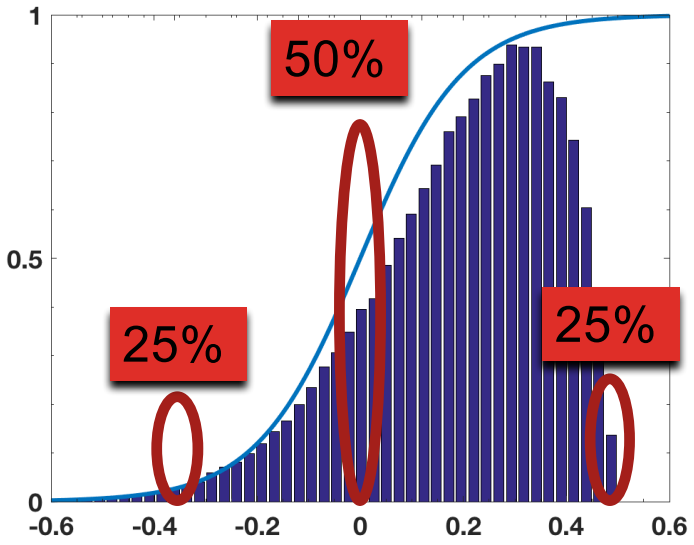

Figure 1: Data Selection for Active Learning

After obtaining the extra data, the extra data is added to the original training-set. Then repeat the procedure in Sect.3.2 to obtain the final classifier. The overall procedure of the training process is shown in Fig. 2.

Mistake 3.

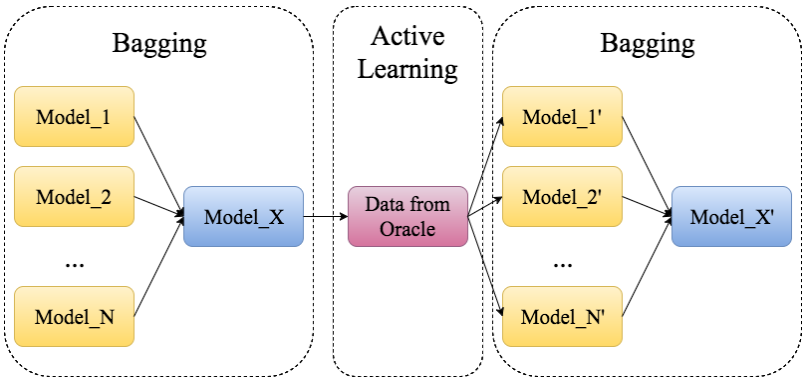

Figure 2: Data Selection for Active Learning

4 Performance Prediction

Once the final classifier is obtained, bootstrap method is applied to observe the distribution of the targets performance metrics: precision and recall. This algorithm is to randomly select a percentage of the data with replacement, then measure the precision and recall of this subset. Repeat this procedure until the mean and standard deviation of precision and recall are converged. For each step in Sec.3.2 and Sec.3.3, the precision is shown in Tab. 1 and the recall is shown in Tab. 2.

	mean	standard deviation
Under-sample	0.2956	0.0197
Under-sample with Bagging	0.2936	0.0192
Under-sample, Bagging and active	0.3001	0.0201
Blind-test	0.267	-

Table 1: Precision

	mean	standard deviation
Under-sample	0.2934	0.0182
Under-sample with Bagging	0.2903	0.0173
Under-sample, Bagging and active	0.3025	0.0179
Blind-test	0.280	-

Table 2: Recall

Showing the precision-recall curve of all the features is difficult. Fig. 3 shows a precision-recall curve of one random feature.

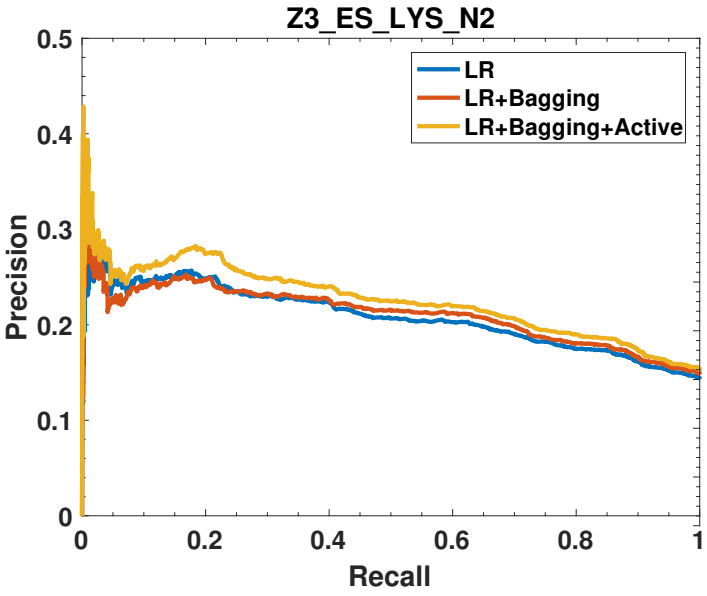

Figure 3: Precision-Recall Curve

5 Discussion

Unfortunately, neither performance metrics achieved to the expectation. Precision was out of one σ range. There are several mistakes we made during the experiment shown in the comments in Sec. 3.2 and 3.3.

- Mistake 1: As beginners of Weka users, we could not adjust the threshold of logistic regression classifier. Therefore at the beginning of the project, only a balanced data set can give a reasonable results. When research goes deeper, we implemented the classifier in Matlab and the threshold of the classifier is no longer bounded at 0.5. Thus the balance of data-set is no longer needed.

- Mistake 2: After obtaining the 10 classifier, we used the one with the highest score out of the 10. According to the results in Tab. 1 and 2, this step improved very little to the overall performance.

- Mistake 3: After adding the 1000 extra data to the original training-set, we re-created 10 subset using under-sampling. Then we trained the classifier using each of the subset and then chose the one with the highest score. The mistake is that once doing the under-sampling, the negative data around the decision boundary are probably never used.

- The data in each feature is highly mixing. When adjusting the decision boundary, it might add some points to the correct region, but lose some points that were correct before. Thus both meta-learning scheme did improve the performance significantly.

Based on the above observation, we set up another experiment in the following procedures:

1. Set 80% of labeled data as training-set, 20% as validation set.

2. Re-sample 80% of the training-set as a subset, train a classifier using this subset.

3. Repeat step 2 10 times, then average the decision boundaries. ($\bar{\omega} = \frac{\sum_1^{10} \omega_i}{10}$)

4. Ask 1000 data around the decision boundaries.

5. Adding the 1000 data to each one of the subset, train 10 classifier again, then average the decision boundaries. Obtain the final classifier.

After 10 cross-validation and applying the above algorithm in each cross-validation, the performance metric is shown in Tab. 3 and 4. As can be seen, both meta-learning techniques failed to improve the overall performance. But by avoiding the possible mistakes shown above, the overall performance was improved and the results was stable using the validation-set. However the blind-test dropped to 26.7% again, which is very closed to the first results. There are two possible reasons:

- The performance of the blind-test depends on the prior probability of the training-set. One observation from the results of the blind-test is that the precision is not as closed to the recall as the validation-test.

- Due to the lack experience of programming on the dataset, we did not find a very efficient method to obtain the cross-validation data-set. We manually obtained 10 sets and it turned out that it is not enough. T-test on the performance metrics did not give a persuasive confidence interval as well. Thus our expectation is higher than the actual results.

During this project we explored every corner of logistic regression that we could think of. For the next time, we will put more effort on cross-validation and predict a more reliable performance metrics.

	mean	standard deviation
Direct Logistic Regression	0.3034	0.0245
Bagging	0.3059	0.0246
Bagging and Active	0.3059	0.0246
Blind-test	0.2668	-

Table 3: Precision in re-experiment

	mean	standard deviation
Direct Logistic Regression	0.3016	0.0213
Bagging	0.3045	0.0210
Bagging and Active	0.3045	0.0210
Blind-test	0.2841	-

Table 4: Recall in Re-experiment