Cost Learning Network for Imbalanced Classification

Chun-Yi Tu
University of California, San Diego
San Diego, CA, U.S.A.
eric.cytu@gmail.com

Hsuan-Tien Lin
National Taiwan University
Taipei, Taiwan
htlin@csie.ntu.edu.tw

***Abstract*—This paper proposes a method to improve the performance of imbalanced classification via reinforcement learning and cost-sensitive learning. Since the cost information is usually unavailable for cost-sensitive learning, we incorporate reinforcement learning to optimize the specified metric by adjusting the cost-matrix for the underlying cost-sensitive classifiers. Our experiment results show that, with the learned cost-matrix, the cost-sensitive classifiers can achieve better performance on several benchmark imbalanced data sets.**

I. Introduction

Class Imbalance is a common problem in practice, including fraud detection and medical diagnosis. However, when the number of examples in one class far exceeds the other, regular classifiers would tend to classify all minority class instances into the majority class because of the underrepresentation in the minority class.

Furthermore, the metric for imbalanced classification should be carefully chosen. While the measure for most classification algorithms is accuracy, it is not a proper criterion for imbalanced classification. For example, in a fraud detection system where the ratio of fraud is often quite low $(1 : 1000)$, a classifier which always predicts a transaction as legitimate one can still achieve 99.9% accuracy. Hence, we often use G-mean [1] or average Recall to measure the performance of imbalanced classification.

In this work, we use cost-sensitive classification to deal with imbalanced classification, and try to improve its performance with respect to G-mean. We find the critical problem of cost-sensitive classification is that the real cost information is usually unavailable, which results in the sub-optimal performance of this method. The existing method only uses some plausible approaches to obtain the artificial cost, and then apply cost-sensitive classification. For example, a naive method for getting the cost is based on the data distribution, as follows.

$$\begin{cases} C_{i,j} = \frac{\text{\# of examples in class } j}{\text{\# of examples in class } i}, & \text{if } i \neq j. \\ C_{i,j} = 0, & \text{if } i = j. \end{cases} \tag{1}$$

With this design, when the number of examples in class j is larger than that in class i, the cost of assigning a class i instance to class j would also be larger. However, this method still oversimplifies the problem. Instead, we use reinforcement learning to adjust the cost-matrix, and directly optimize the specified metric for imbalanced classification.

II. Related Work

A. Cost-sensitive Classification

To deal with imbalanced classification, generally, there are two common approaches [2].

1) Data-level approaches: pre-process the original data to make data distribution balanced
2) Algorithm-level approaches: modify the learning algorithm to improve the performance on minority class.

The first approach often pre-processes the data by under-sampling the majority or over-sampling the minority class. The most popular one is SMOTE (Synthetic Minority Over-sampling TEchnique) [3] , which over-sampled the minority class. The synthetic examples are created by taking interpolation between each minority class example and its nearest neighbors. However, we believe under-sampling could loss some important information, and over-sampling does not provide new information.

Instead, we focus on an algorithm-level approach: cost sensitive classification. Regular classification assumes all errors have the same cost, while cost-sensitive classification assigns different costs for different kinds of mis-classification error, and the goal of cost-sensitive classification is to minimize the total cost. As a result, with higher cost on the minority class and smaller cost on the majority class, the performance of imbalanced classification can be improved by avoiding predicting all examples as majority class.

For instance, Domingos [4] proposed a method, MetaCost, which makes an arbitrary classifier cost-sensitive. The MetaCost procedure first estimates the probability $P(j|x)$ of the given example x and the class j, and the optimal prediction for x is the class i that minimizes the conditional risk:

$$R(i|x) = \sum_j P(j|x)C(j,i), \tag{2}$$

where $R(i|x)$ is the expected cost of assigning x to class i, and C is the cost matrix. Then, it relabels the training data with the estimated optimal class, and reapplies the regular classifier to the relabeled training data.

B. Reinforcement Learning

Reinforcement learning addresses the problem of maximizing cumulative reward through interactions between agents and environment. Generally, the learner samples data from the environment and optimizes an objective function alternately. To update the cost-matrix, we use Policy gradient in this work, which are the most popular reinforcement learning algorithms that can output continuous actions. The basic policy gradient tries to perform gradient ascent on the following loss function

$$L^{PG}(\theta) = \mathbb{E}_t[\log \pi_\theta(a_t|s_t)A_t],$$

where π_θ is the policy network and A_t is an advantage function at time t. In this work, we adopt the state-of-the-art policy gradient method, Proximal Policy Optimization (PPO) [5]. PPO uses the clipped surrogate objective to achieve better performance.

$$L^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)],$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, and ϵ is a hyperparameter.

III. Methods

We model the process of learning a good cost-matrix for imbalanced-classification as a reinforcement learning problem, and use PPO as the RL learner to adjust the cost-matrix. The framework can be separated to two parts: 1. RL learner. 2. Environment. The RL implementation is modified from OpenAI's baseline package "PPO2" [6], and our work is mainly in the design of the environment and the interaction between them.

A. RL learner

First, the observation s_t for RL is defined as the confusion-matrix of the previous classification result because the learner can try to adjust the cost-matrix to maximize the total rewards based on the information from the previous confusion-matrix. For example, when an off-diagonal entry in the confusion-matrix is far larger than the others, increasing the corresponding value in the cost-matrix may decrease that value, and thus improve the performance. The action a_t is an update direction that is added to the cost-matrix (off-diagnoal value of the previous cost-matrix). The reward A_t is a classification metric (we use average g-mean) from the underlying cost-sensitive classification trained with the cost-matrix after the update.

The RL learner (PPO) tries to update the cost-matrix from the regular one in 20 steps, 1 episode, and then the cost-matrix (environment) will be reset. We use LSTM [7] as the base model for PPO, since the previous observations can be helpful to update the cost-matrix. The concept is that, when the RL learner see a similar confusion-matrix it has seen before, it may output a better action based on the history information.

B. Environment

The classification is performed in the environment. Since training a new classifier for every new cost-matrix is very expensive, in this work, we use direct cost-sensitive decision making [8] as a cost-sensitive classifier.

1) Direct Cost-sensitive Decision Making: Direct cost-sensitive decision making is similar to MetaCost. However, after training a probability estimator, it just uses the estimated probabilities and the given cost matrix to make the optimal prediction without relabeling. That is, it simply predicts the example as the class i that minimizes the conditional risk, according to the Eq.(2).

2) Expected Cost Regularization for Probability Estimator: To improve the probability estimator's performance in imbalanced classification, we propose to minimize both cross entropy and expected cost, which could be seen as a regularizer in Cost-sensitive learning. Specifically, as the probability estimator minimizes cross entropy, it also tries to penalize high expected cost to achieve good probability estimates and low expected cost. As a result, our probability estimator optimizes the special loss

$$Loss = CrossEntropy + ExpectedCost \tag{3}$$

$$= min\frac{1}{N}\sum_i^N y_i \log p(y_i|x_i,\theta) + \frac{\lambda}{N}\sum_i^N p(y_i|x_i,\theta)^T C_i, \tag{4}$$

where C_i is the row i of the cost matrix.

C. Learning Approach

We first train an Neural Network probability estimator on different data sets to derive the estimated probability. Then when resetting the environment, we initialize the cost matrix. For every step the RL learner takes, we update the cost-matrix according to the given action, and use direct cost-sensitive decision making to perform classification. The specified metric of the classification result is the reward, and the confusion matrix is the next observation for the RL learner. That is, we can derive a new classification result for every new cost matrix. With this design, the RL learner could try to optimize the cost-matrix and maximize the reward. We present our model in Algorithm 1.

IV. Experiments

In the experiments, we first show the effect of cost regularization. Then, we compare our learning approach with the existing methods on KEEL imbalanced data sets [9] which contains multi-class data with varying imbalance ratio.

A. Cost Regularization

In this section, we show the classification result of different λ in Eq.(4). Our model is trained with Adam optimizer. The model is a NN with 2 hidden layers, and the number of neurons for both hidden layers are 64. Instead of optimizing regular cross entropy, we optimize our special loss. The learning rate is set to 0.001, and the batch size is 32. We use *contraceptive* and *thyroid* from KEEL data sets to show the classifier's performance, and follow the traditional cost matrix setting Eq.(1). Figure 1 and 2 show that when $\lambda = 1$, the expected cost is the lowest, and the cross entropy becomes slightly higher than the others, which corresponds to our intuition that the

Algorithm 1 Cost Learning Network

```
Require: Training set $D = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$
 1: Train a probability estimator on $D$ with Eq.(4), and store
    all estimated probabilities
 2: for t = 1 to T do
 3:     Reset the cost-matrix
 4:     for step = 1 to 20 do
 5:         Update the cost-matrix with the action output from
            the RL learner
 6:         Derive the classification result with the new cost-
            matrix according to Eq.(2)
 7:         Compute the observations and the rewards for the RL
            learner
 8:     end for
 9:     Optimize the RL learner
10: end for
Ensure: The best classification result according to the given
    metric
```

cross entropy is compromised to achieve low expected cost. Figure 3 and 4 show that although the accuracy of $\lambda = 1$ is lower than the regular one ($\lambda = 0$), cost regularization can achieve a better g-mean.

B. Cost Learning

We compare our method described in Algorithm 1 on multiclass imbalanced classification with Datta's boosted ensemble: Dual-LexiBoost [10]. Note that the C4.5 decision tree [11] and the k-Nearest Neighbor (kNN) classifier are used as base learners for Dual-LexiBoost. Table 1 summarizes the performance of different methods on KEEL data sets. Note that the three NN methods use the same model described in section 4.1. NN is the regular neural-network classification, which can be seen as using regular cost-matrix. NN w/ cost-matrix uses the cost-sensitive classifier specified in section 3.2.1 with the fixed cost-matrix according to Eq.(1). The proposed NN w/ CLN is described in Algorithm 1, which starts from a regular cost-matrix, and uses RL to learn an optimal cost-matrix for the same cost-sensitive classifier.

The experiment result shows that the performance of using RL to tune the cost-matrix for cost-sensitive NN is superior to using regular NN or cost-sensitive NN with the cost-matrix based on data distribution. However, if the performance of the initial NN is bad or even zero, there is little improvement of using RL-NN since it is hard to train RL with only zero reward. Furthermore, most of our method is better than Dual-LexiBoost. The zero g-mean result of some dataset is because the data distribution of that is too extreme (only several examples in some classes) to train a classifier.

V. Conclusion

We introduced a framework to directly learn the cost-matrix for imbalanced classification via reinforcement learning. The experiments shows the good performance on some benchmark imbalanced data sets. To reduce the high computational complexity, we use direct cost-sensitive decision making to avoid re-training the base classifier. Besides, the framework is applicable in more general settings, and can be used with any probability estimators.

Acknowledgement

We thank members of the NTU Computational Learning Laboratory and the anonymous reviewers for valuable comments. The work was partially supported by the Ministry of Science and Technology of Taiwan via MOST 107-2628-E-002-008-MY3 and 108-2119-M-007-010.

References

[1] Y. Sun, M. S. Kamel, and Y. Wang, "Boosting for learning multiple classes with imbalanced class distribution," in *Sixth International Conference on Data Mining (ICDM'06)*, Dec 2006, pp. 592–602.

[2] A. Ali, S. M. Shamsuddin, and A. Ralescu, "Classification with class imbalance problem: A review," vol. 7, pp. 176–204, 01 2015.

[3] K. W. Bowyer, N. V. Chawla, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *CoRR*, vol. abs/1106.1813, 2011. [Online]. Available: http://arxiv.org/abs/1106.1813

[4] P. Domingos, "Metacost: A general method for making classifiers cost-sensitive," in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '99. New York, NY, USA: ACM, 1999, pp. 155–164. [Online]. Available: http://doi.acm.org/10.1145/312129.312220

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[6] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "Openai baselines," https://github.com/openai/baselines, 2017.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[8] B. Zadrozny and C. Elkan, "Learning and making decisions when costs and probabilities are both unknown," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '01. New York, NY, USA: ACM, 2001, pp. 204–213. [Online]. Available: http://doi.acm.org/10.1145/502512.502540

[9] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, and S. García, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework." *Multiple-Valued Logic and Soft Computing*, vol. 17, no. 2-3, pp. 255–287, 2011.

[10] S. Datta, S. Nag, and S. Das, "On boosting, tug of war, and lexicographic programming," *CoRR*,

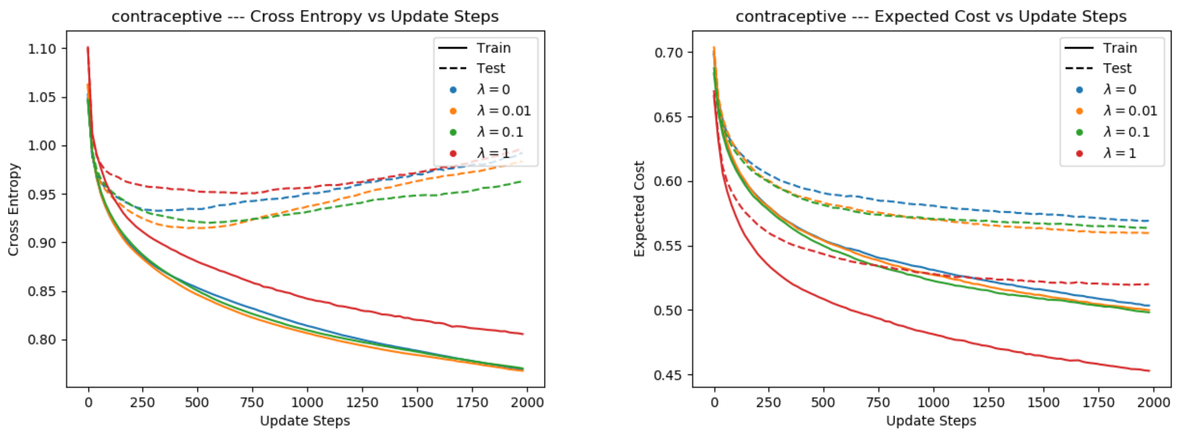

Fig. 1. The learning curve of Cross entropy(L) and expected cost(R) for different λ on *contraceptive*

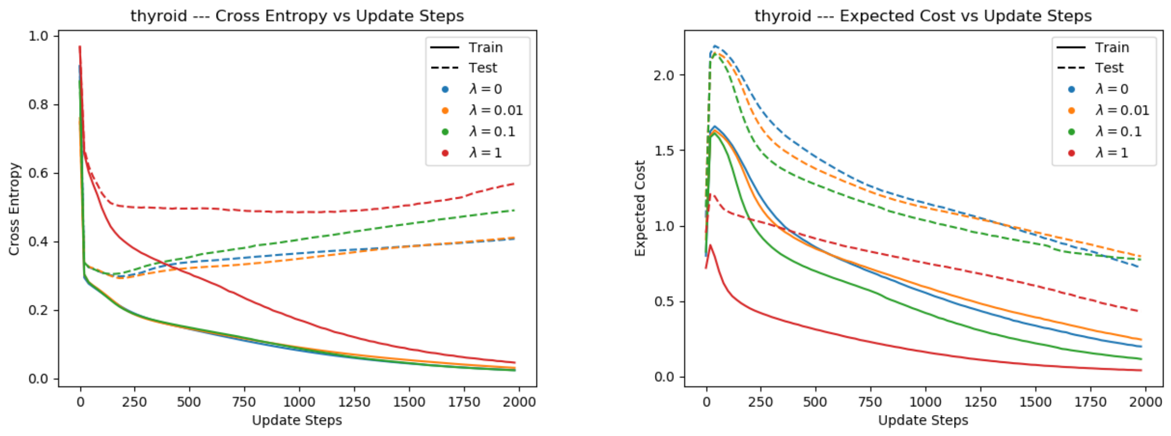

Fig. 2. The learning curve of Cross entropy(L) and expected cost(R) for different λ on *thyroid*

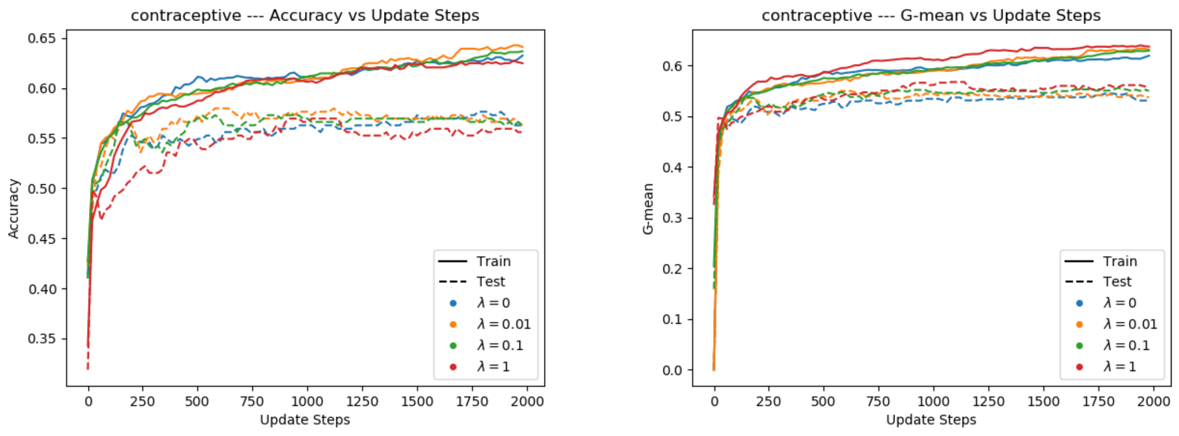

Fig. 3. The learning curve of Accuracy(L) and G-mean(R) for different λ on *contraceptive*

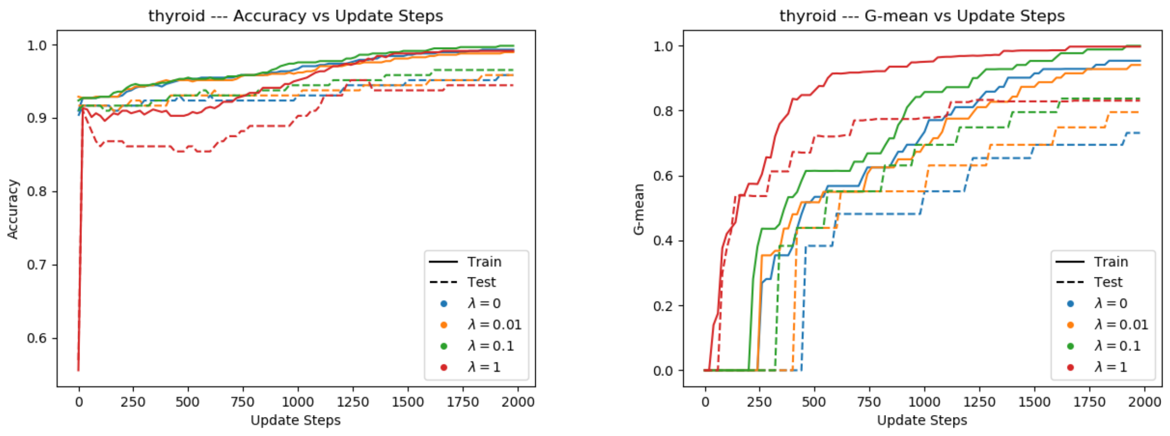

Fig. 4. The learning curve of Accuracy(L) and G-mean(R) for different λ on *thyroid*

Dataset/G-mean	Dual-LexiBoost w/ C4.5	Dual-LexiBoost w/ kNN	NN	NN w/ cost-matrix	NN w/ CLN
wine	0.9091	0.6959	**0.9458**	**0.9458**	**0.9458**
hayes-roth	0.7318	0.6767	0.6745	0.7100	**0.7688**
contraceptive	0.4135	0.5037	0.5605	0.4874	**0.5911**
new-thyroid	0.8716	0.9452	0.9086	**0.9880**	0.9086
thyroid	**0.9091**	0.4406	0.7482	0.6546	0.8348
pageblocks	0.5479	0.5272	**0.9677**	0.5788	0.9657
shuttle	0.6652	0.6617	0.0000	0.0000	**0.8324**
ecoli	0.0000	**0.5622**	0.0000	0.0000	0.0000
yeast	0.0000	**0.5081**	0.0000	0.0000	0.0000

TABLE I
COMPARISON OF OUR APPROACH WITH DUAL-LEXIBOOST

vol. abs/1708.09684, 2017. [Online]. Available: http://arxiv.org/abs/1708.09684

[11] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.