Towards Automated Imbalanced Learning with Deep Hierarchical Reinforcement Learning

Daochen Zha†, Kwei-Herng Lai†, Qiaoyu Tan‡, Sirui Ding‡, Na Zou‡, Xia Hu†

†*Department of Computer Science, Rice University*

‡*Department of Computer Science and Engineering, Texas A&M University*

Imbalanced Classification Problem

- **What is the problem?** Imbalanced learning is a fundamental challenge in data mining, where there is a disproportionate ratio of training samples in each class.
- **How to tackle imbalanced learning?** Data-level methods aim to balance the data distributions. It includes over-sampling, under-sampling, and hybrid methods. We focus on over-sampling.
- **What is SMOTE?** SMOTE is one of the most popular over-sampling techniques with many extensions built upon it.

High-Level Idea

- **Motivation:** The existing algorithms often rely on the heuristics and are designed based on some assumptions, such as samples in the borderline are more important, which could be sub-optimal.
- **Automating over-sampling:** Given a dataset and a base classifier, how can we optimize the over-sampling strategy such that the trained classifier can achieve the best generalization performance?

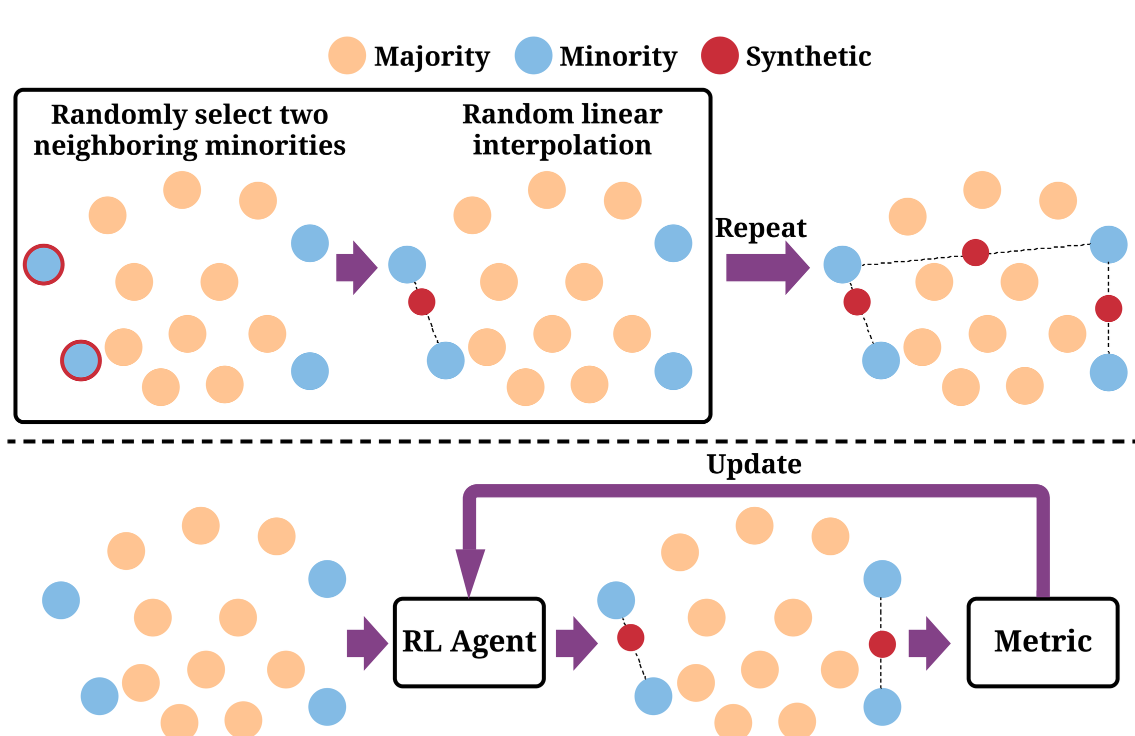

Figure: The decisions are randomly made in SMOTE (top) while the decisions in AutoSMOTE are made with an RL agent to optimize the performance on the validation set (bottom).

Challenges

- **How to optimize?** It is hard to directly optimize the performance metric. The sampling is independent of the classifier so that it can only indirectly impact the performance.
- **How to be efficient?** The problem has a huge decision space since the number of generated samples can be arbitrarily large, and each synthetic sample can be anywhere in the feature space.
- **How to deal with hierarchical reasoning?** At the high level, we need to decide the over-sampling ratio. At the low level, we need to decide where the synthetic samples should be located, which depends on the high-level decision.

AutoSMOTE: Optimizing Over-Sampling with Hierarchical Reinforcement Learning

- **High-level policy:** We decide how many samples will be generated around the current instance.
- **Low-level policy:** We decide which neighbors to perform linear interpolation and the interpolation weight to generate a new sample.
- **Reward:** Classification performance on the validation set.

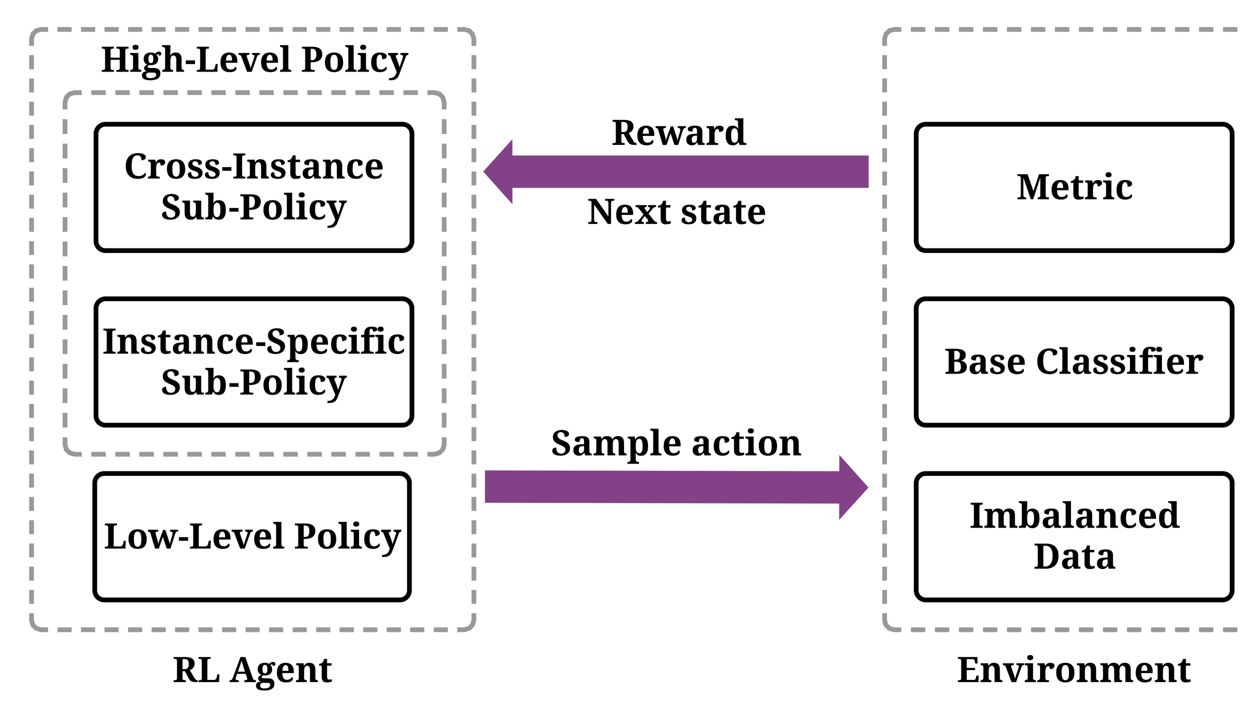

Figure: An overview of AutoSMOTE. The RL agent generates synthetic samples (actions) based on the current data distribution (state) with a high-level policy for deciding sampling ratios, and a low-level policy for performing actual sampling, where the high-level policy consists of two sub-policies that collaboratively make decisions. The environment takes as input the action and transits to the next state. The performance metric of the base classifier on the validation data serves as the reward to update the RL agent.

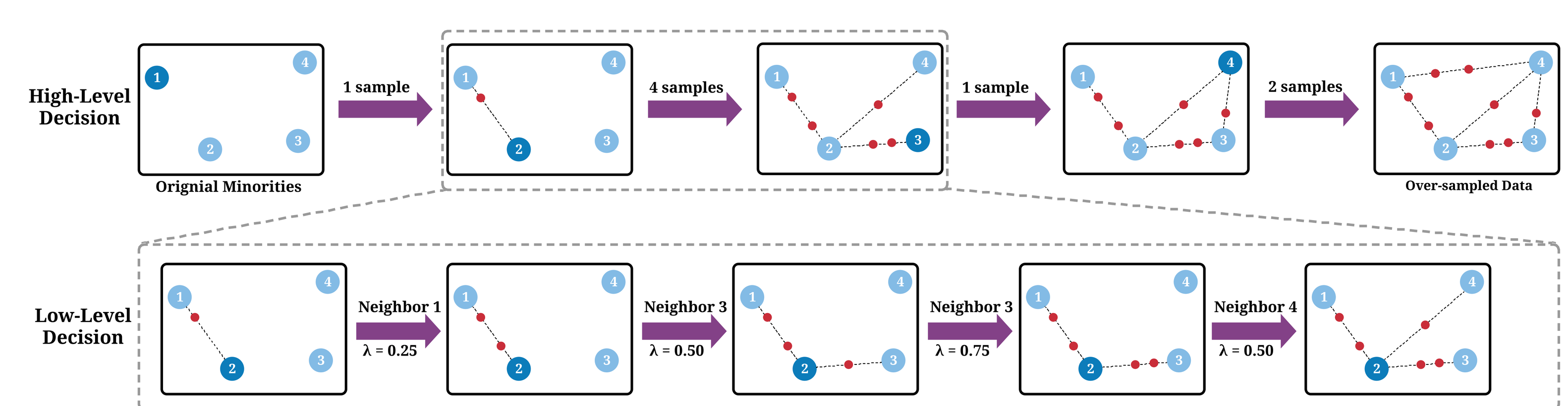

Figure: An illustration of the hierarchical decision process. We go through the minority instances one by one, where the darker blue instance is the current instance to be augmented. At the high level, we decide how many synthetic instances will be sampled around the current instance. At the low level, we decide which neighboring instances to perform linear interpolation and the interpolation weights λ. The low-level decision depends on the high-level decision since the length of the low-level sampling is determined by the decisions made in the high-level.

Results

AutoSMOTE outperforms the state-of-the-art over-sampling algorithms. Read more with the following QR codes.

Paper

Code