Towards Automated Imbalanced Learning with Deep Hierarchical Reinforcement Learning

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

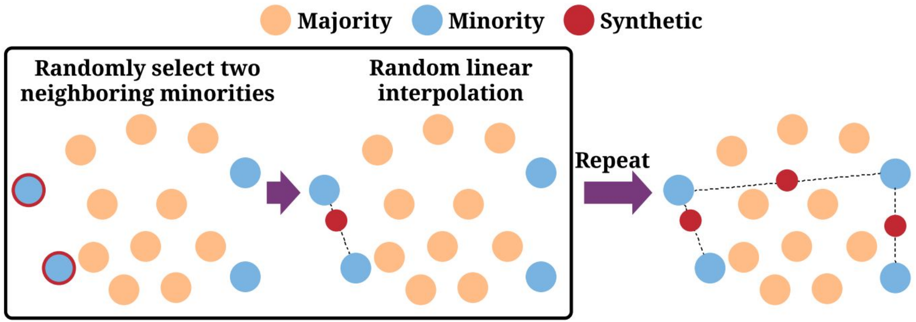
Rice University
Texas A&M University

Background

- **Imbalanced Classification Problem**
  - There is a disproportionate ratio of training samples in each class.
  - # of majority instances >> # of minority instances.
  - Classifiers tend to be dominated by the majority class and perform poorly on the minority class.
- **Over-Sampling**
  - One effective way to tackle data imbalance is over-sampling.
  - It generates new synthetic samples for the minority class.
  - SMOTE is the most popular over-sampling technique.

How does SMOTE Work?

- **SMOTE Iteratively Execute the Following:**
  - Randomly pick a minority instance.
  - Find the nearest minority neighbors of this instance and randomly pick a neighbor.
  - Perform linear interpolation between the selected instance and the neighbor to generate a new sample.



The synthetic instance interleaves with majories!

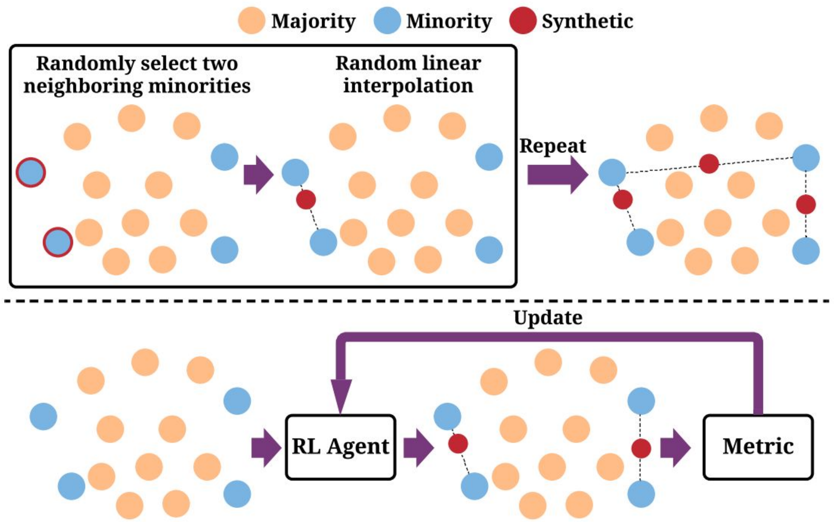
SMOTE Variants

- **Many Ideas Have been Explored**
  - At least 85 SMOTE variants as of the year of 2019 [1].
  - ADASYN [2] generates more synthetic samples for the instances that are harder to learn, which is quantified by the ratio of the majority instances in the nearest neighbors.
  - BorderlineSMOTE [3] and SVMSMOTE [4] only over-sample the minority instances in the borderline.
  - ANS [5] proposes to adapt the number of neighbors needed for each instance.
  - However, the existing SMOTE variants heavily rely on the heuristics to perform over-sampling.

[1] Smote-variants: A python implementation of 85 minority oversampling techniques. Neurocomputing.
[2] ADASYN: Adaptive synthetic sampling approach for imbalanced learning. IJCNN.
[3] Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. ICIC.
[4] Borderline oversampling for imbalanced data classification. International Journal of Knowledge Engineering and Soft Data Paradigms.
[5] Adaptive neighbor synthetic minority oversampling technique under 1NN outcast handling. Songklanakarin J. Sci. Technol.

Learning-based Over-Sampling

- **Research Question**
  - 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?

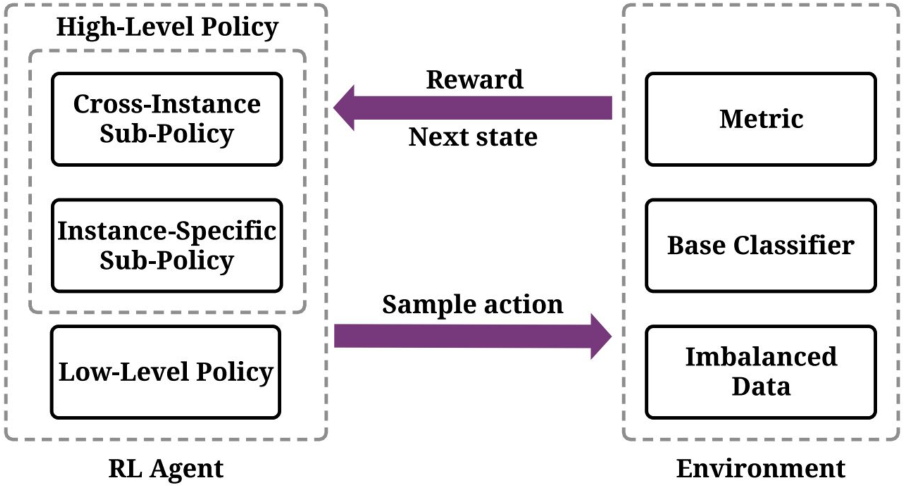


Challenges

- **How to Optimize?**
  - The sampling is independent of the classifier so that it can only indirectly impact the performance. We need an effective mechanism to fill this gap so that the sampling strategy can be learned.
- **How to Deal with the Huge Decision Space?**
  - the number of generated samples can be arbitrarily large, and each synthetic sample can be anywhere in the feature space.
- **How to Perform Hierarchical Reasoning?**
  - At the high level, we decide the over-sampling ratio, i.e., how many synthetic samples should be generated.
  - At the low level, we decide where the synthetic samples should be located.
  - The low-level decision depends on the high-level decision in that the optimal locations of the samples may differ for different numbers of samples

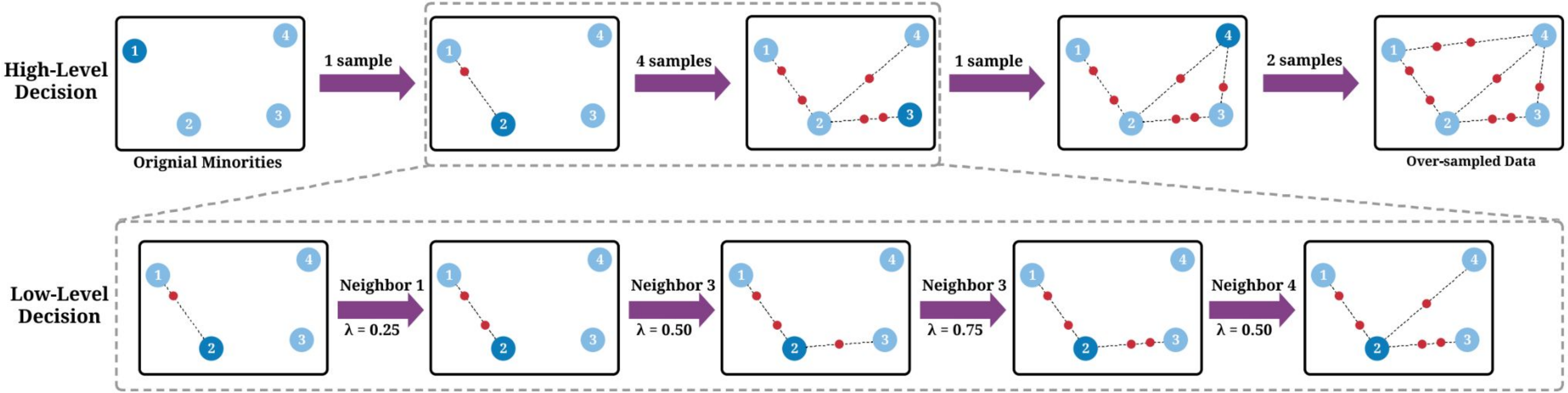
AutoSMOTE Framework

- **Hierarchical Reinforcement Learning**
  - High-level policy: it decides how many synthetic instances will be generated for each instance.
  - Low-level policy: it decides how the interpolation is performed.



AutoSMOTE Framework

- **Sampling Process**



AutoSMOTE Framework

- **Training**

Algorithm 2 Training of AutoSMOTE

Input: $\pi_h^{(1)}, \pi_h^{(2)}, \pi_l, \mathbf{X}^{\min}, G_1, G_2, K$, total number of iterations I, three buffer sizes $B_h^{(1)}$, $B_h^{(2)}$, and B_l

Initialize three queue buffers $\mathcal{B}_h^{(1)}, \mathcal{B}_h^{(2)}, \mathcal{B}_l$

for iteration = 1, 2, ..., I **do**

  Generate samples following Algorithm 1 and store the generated episodes to $\mathcal{B}_h^{(1)}, \mathcal{B}_h^{(2)}$ and $\mathcal{B}_l$

  Train on the augmented training data, get reward on validation data, and set the final steps of all the episodes to be the obtained reward with all the intermediate rewards as 0

  if size($\mathcal{B}_h^{(1)}$) $\geq B_h^{(1)}$ **then**

    Pop out $B_h^{(1)}$ steps of data and update $\pi_h^{(1)}$ with Eq. 2 — Cross-Instance Sub-Policy

  end if

  if size($\mathcal{B}_h^{(2)}$) $\geq B_h^{(2)}$ **then**

    Pop out $B_h^{(2)}$ steps of data and update $\pi_h^{(2)}$ with Eq. 2 — Instance-Specific Sub-Policy

  end if

  if size($\mathcal{B}_l$) $\geq B_l$ **then**

    Pop out B_l steps of data and update π_l with Eq. 2 — Low-level Policy

  end if

end for

Experiments

- **Datasets**

Dataset statistics with imbalanced ratios of 20/50/100

	# Majorities	# Minorities	# Features	Domain
Phoneme	3818	190/76/38	5	Audio
PhishingWebsites	6157	307/123/61	68	Security
EEGEyeState	8257	412/165/82	14	EEG
Mozilla4	10437	521/208/104	5	Product defect
MagicTelescope	12332	616/246/123	10	Telescope
Electricity	26075	1303/521/260	14	Electricity

Experiments

- **Base Classifiers**
  - SVM, KNN, DecisionTree, AdaBoost
- **Metrics**
  - Macro-F1, MCC
  - Average rank across the 12 settings (4 classifier x 3 imbalanced ratios)

Daochen Zha (daochen.zha@rice.edu)

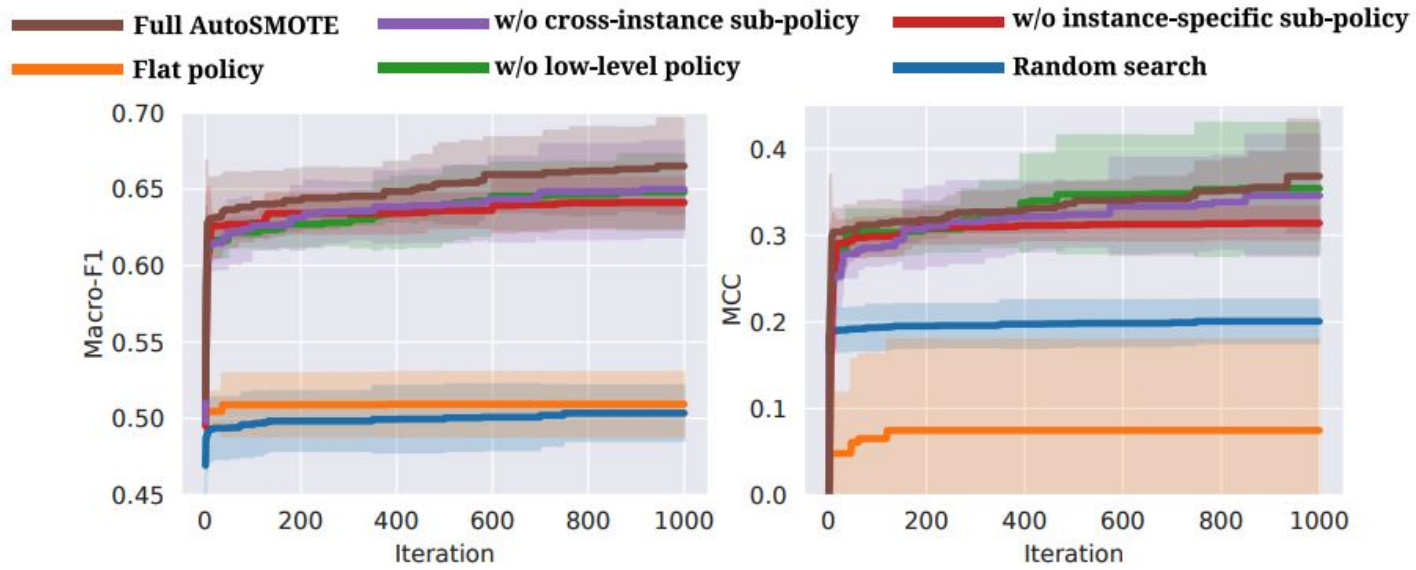
Experiments

- **Comparison with the State-of-the-Art Samplers**

Category	Method	Dataset						Overall
		Phoneme	PhishingWebsites	EEGEyeState	Mozilla4	MagicTelescope	Electricity	
No-resampling	-	16.50▲/16.75▲	9.75 /9.42	16.08▲/17.08▲	12.75▲/12.83▲	15.92▲/13.00▲	18.17▲/15.67▲	14.86▲/14.12▲
Under-sampling	ClusterCentroids	13.25▲/14.58▲	19.50▲/19.58▲	13.67▲/14.33▲	15.67▲/15.42▲	16.42▲/19.58▲	15.25▲/18.25▲	15.62▲/14.58▲
	CondensedNearestNeighbour	16.62▲/17.46▲	16.92▲/16.96▲	17.75▲/19.17▲	19.88▲/20.46▲	14.33▲/14.92▲	13.58▲/14.58▲	16.51▲/17.19▲
	EditedNearestNeighbours	14.17▲/15.42▲	11.83 /12.42	15.71▲/16.88▲	11.96▲/12.46▲	13.04▲/10.71	15.25▲/15.50▲	13.66▲/13.90▲
	RepeatedEditedNearestNeighbours	14.71▲/17.04▲	15.29▲/15.96▲	15.33▲/17.17▲	13.88▲/14.12▲	10.38▲/8.88	14.62▲/14.88▲	14.03▲/14.67▲
	AllKNN	14.04▲/15.46▲	13.46▲/13.46	15.50▲/16.75▲	13.88▲/14.29▲	10.71▲/8.38	16.08▲/17.17▲	13.94▲/14.25▲
	InstanceHardnessThreshold	14.21▲/13.38▲	20.67▲/20.67▲	14.29▲/14.46▲	17.79▲/18.04▲	10.25 /10.25	12.58▲/12.83▲	14.97▲/14.94▲
	NearMiss	24.42▲/24.75▲	24.17▲/24.42▲	22.58▲/20.92▲	23.25▲/23.33▲	25.00▲/25.00▲	19.58▲/21.08▲	23.17▲/23.25▲
	NeighbourhoodCleaningRule	16.33▲/17.83▲	13.08 /13.25▲	15.12▲/15.88	12.83▲/13.25▲	10.79 /9.12 ▲	12.54▲/11.71▲	13.45▲/13.51▲
	OneSidedSelection	17.21▲/18.21▲	11.08 /10.67▲	15.92▲/16.17▲	13.83▲/14.17▲	15.62▲/13.21▲	17.12▲/15.71▲	15.13▲/14.69▲
	RandomUnderSampler	12.50▲/10.00▲	17.42▲/17.58▲	11.00▲/9.25 ▲	12.83▲/12.92▲	10.17 /11.67	10.83▲/12.25▲	12.46▲/12.28▲
	TomekLinks	16.88▲/17.46▲	10.04 /9.38	14.12▲/14.04▲	14.12▲/14.29▲	15.62▲/12.96▲	17.38▲/16.21▲	14.69▲/14.06▲
Over-sampling	RandomOverSampler	6.75 /8.17	12.33▲/12.75▲	5.00 /5.58	8.00 ▲/8.42 ▲	9.92 ▲/13.33	8.58 ▲/10.83▲	8.43 ▲/9.85 ▲
	SMOTE	7.25 /8.67 ▲	10.42▲/10.67▲	7.00 ▲/6.67	12.00▲/12.00▲	11.42▲/14.17▲	7.17 ▲/7.42 ▲	9.21 ▲/9.93 ▲
	SMOTEN	16.83▲/18.25▲	10.71 /10.54	12.42▲/15.58▲	9.58 ▲/10.08	18.17▲/17.33▲	17.83▲/18.67▲	14.26▲/15.08▲
	ADASYN	7.33 /8.00	9.75 /9.58	7.50 ▲/8.17 ▲	12.58▲/12.25▲	10.17 /12.08▲	8.00 ▲/8.50 ▲	9.22 ▲/9.76 ▲
	BorderlineSMOTE	6.92 /8.67	9.42 ▲/9.25	9.67 ▲/10.92▲	9.17 ▲/9.33 ▲	7.50 /9.75	4.67 /5.08	7.89 ▲/8.83 ▲
	KMeansSMOTE	15.92▲/16.67▲	10.00 /9.83	16.08▲/16.92▲	12.83▲/12.79▲	14.92▲/12.17	17.92▲/15.42▲	14.61▲/13.97▲
	SVMSMOTE	6.25 /9.08 ▲	10.17▲/10.00	7.25 /7.75	8.25 ▲/9.33 ▲	6.67 /8.58	4.50 /4.83	7.18 ▲/8.26 ▲
Combined over- and under-sampling	SMOTEENN	6.25 /6.50	14.67 /14.50	7.17 /6.92	10.75▲/10.08	8.00 /7.42	9.75 ▲/9.67 ▲	9.43 ▲/9.18 ▲
	SMOTETomek	8.67 /9.25 ▲	9.58 /9.75	6.67 ▲/6.42	11.42▲/11.08▲	8.58 /10.25	7.75 ▲/7.92 ▲	8.78 ▲/9.11 ▲
Generative models	CTGAN	12.08 /9.33 ▲	11.75▲/11.42▲	11.50▲/12.58	10.42▲/9.25 ▲	15.17 /15.42▲	12.75 /11.83▲	12.28▲/11.64▲
	TVAE	14.25▲/12.17▲	9.42 /9.17	23.50▲/18.50▲	16.17▲/16.58▲	20.92▲/20.92▲	19.58▲/17.25▲	17.31▲/15.76▲
Meta-learning	MESA	19.92▲/7.33	17.08▲/16.50▲	20.17▲/12.17▲	17.50▲/13.25▲	19.58▲/18.92▲	20.83▲/18.50▲	19.18▲/14.44▲
Auto-sampling	AutoSMOTE	**5.75 /4.58**	**6.50 /7.67**	**4.00 /4.75**	**3.67 /4.96**	**5.75 /7.00**	**2.67 /3.25**	**4.72 /5.37**

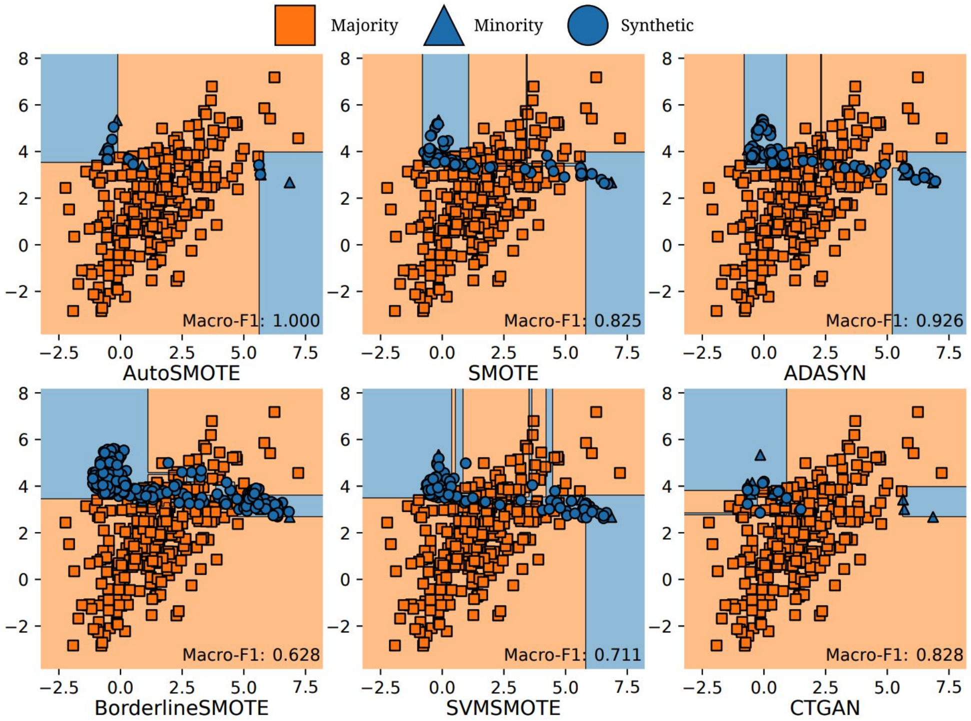
Experiments

- **Ablation Study**



Experiments

- **Visualization**



Summary and Takeaways

- **Contributions**
  - We investigated AutoML for over-sampling for imbalanced classification.
  - We proposed AutoSMOTE, which samples synthetic instances with deep hierarchical reinforcement learning.
  - Extensive experiments demonstrated that AutoSMOTE outperforms the state-of-the-art over-sampling algorithms.

Paper

Code

Daochen Zha (daochen.zha@rice.edu)