Self-paced Ensemble for Highly Imbalanced Massive Data Classification

Zhining Liu[1,2], Wei Cao[3], Zhifeng Gao[3], Jiang Bian[3], Hechang Chen[1,2], Yi Chang[1,2] and Tie-Yan Liu[3]
[1]School of Artificial Intelligence, Jilin University, Changchun, China
[2]Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun, China
[3]Microsoft Research, Beijing, China
znliu19@mails.jlu.edu.cn, {weicao, zhgao, jiang.bian, tyliu}@microsoft.com,
chenhc14@mails.jlu.edu.cn, yichang@jlu.edu.cn

***Abstract*—Many real-world applications reveal difficulties in learning classifiers from imbalanced data. The rising big data era has been witnessing more classification tasks with large-scale but extremely imbalance and low-quality datasets. Most of existing learning methods suffer from poor performance or low computation efficiency under such a scenario. To tackle this problem, we conduct deep investigations into the nature of class imbalance, which reveals that not only the disproportion between classes, but also other difficulties embedded in the nature of data, especially, noises and class overlapping, prevent us from learning effective classifiers. Taking those factors into consideration, we propose a novel framework for imbalance classification that aims to generate a strong ensemble by self-paced harmonizing data hardness via under-sampling. Extensive experiments have shown that this new framework, while being very computationally efficient, can lead to robust performance even under highly overlapping classes and extremely skewed distribution. Note that, our methods can be easily adapted to most of existing learning methods (e.g., C4.5, SVM, GBDT and Neural Network) to boost their performance on imbalanced data.**

***Index Terms*—imbalance learning, imbalance classification, ensemble learning, data re-sampling**

I. Introduction

The development of information technology brings the explosion of massive data in our daily life. However, many real applications usually generate very imbalanced datasets for corresponding key classification tasks. For instance, online advertising services can give rise to a high amount of datasets, consisting of user views or clicks on ads, for the task of click-through rate prediction [1]. Commonly, user clicks only constitute a small rate of user behaviors . For another example, credit fraud detection [2] relies on the dataset containing massive real credit card transactions where only a small proportion are frauds. Similar situations also exist in the tasks of medical diagnosis, record linkage and network intrusion detection etc [3]–[5]. In addition, real-world datasets are likely to contain other difficulty factors, including noises and missing values. Such *highly imbalanced*, *large-scale* and *noisy* data brings serious challenges of downstream classification tasks.

This work was conducted when the first author was an intern at Microsoft Research Asia. This work is partially supported by National Natural Science Foundation of China (No.61976102).

Traditional classification algorithms (e.g., C4.5, SVM or Neural Networks [6]–[8]) demonstrate unsatisfactory performance on imbalanced datasets. The situation can be even worse when the dataset is large-scale and noisy at the same time. Attribute to their inappropriate presuming on relatively balanced distribution between positive and negative samples, the minority class is usually ignored due to the overwhelming number of majority instances. On the other hand, the minority class usually carries the concepts with greater interests than majority class [9], [10].

To overcome such issue, a series of research work has been proposed, which can be classified into three categories:

- *Data-level* methods modify the collection of examples to balanced distributions and / or remove difficult samples. They may be inapplicable on datasets with categorical features or missing values due to their distance-based design (e.g., NearMiss, Tomeklink [11], [12]). Besides, they suffer from large computational cost (e.g., SMOTE, ADASYN [13], [14]) when applying on large-scale data.
- *Algorithm-level* methods directly modify existing learning algorithms to alleviate the bias towards majority objects. However, they require assistance from domain experts before-hand (e.g., setting cost matrix in cost-sensitive learning [15], [16]). They may also fail when cooperating with batch-training classifiers like neural network since they do not balance the class distribution on the training data.
- *Ensemble* methods combine one of the previous approaches with an ensemble learning algorithm to form an ensemble classifier. Some of them suffer from large training cost and poor applicability (e.g., SMOTEBagging [17]) on realistic tasks. The other ones potentially lead to underfitting or overfitting (e.g., EasyEnsemble, BalanceCascade [18]) when the dataset is highly noisy.

For above reasons and more, none of the prevailing methods can well handle the *highly imbalanced*, *large-scale* and *noisy* classification task, while it is a common problem in real-world applications. The main reason behind existing methods' failure on such tasks is that they ignored difficulties embedded in the nature of imbalance learning. Not only the class imbalance it-

2375-026X/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDE48307.2020.00078

self, other factors like presence of noise samples [19] and overlapped underlying distribution between the classes [20], [21] also significantly deteriorate the classification performance. Their influences can be further enlarged by the high imbalance ratio. Besides, different models show various sensitivity to these factors. For above reasons, all these factors need to be considered to achieve more accurate classification.

We introduce the concept of "*classification hardness*" to integrate aforementioned difficulties. Intuitively, hardness represents the difficulty of correctly classifying a sample for a specific classifier. Thus the distribution of classification hardness implicitly contains the information of task difficulties. For example, noises are likely to have large hardness values and the proportion of high-hardness samples reflected the level of class overlapping. Moreover, hardness distribution is naturally adaptive to different models since it was defined with respect to given classifier. Such hardness distribution can be used to guide the re-sampling strategy to achieve better performance.

Based on the classification hardness, we propose a novel learning framework called *Self-paced Ensemble* (abbreviated as `SPE`) in this paper. Instead of simply balancing the positive/negative data or directly assigning instance weights, we consider the distribution of classification hardness over the dataset, and iteratively select the most informative majority data samples according to the hardness distribution. The under-sampling strategy is controlled by a self-paced procedure. Such self-paced procedure enables our framework gradually focuses on the harder data samples, while still keeps the knowledge of easy sample distribution in order to prevent overfitting. Fig. 1 shows the pipeline of self-paced ensemble.

In summary, the contributions of this paper are as follows:

- In this paper we demonstrate the reason of conventional imbalance learning methods failing on the real-world massive imbalanced classification task. We conduct comprehensive experiments with analysis and visualization that can be valuable for other similar classification systems.
- We proposed *Self-paced Ensemble* (`SPE`), a learning framework for massive imbalanced data classification. `SPE` can be used to boost any canonical classifier's performance (e.g., C4.5, SVM, GBDT, and Neural Network) on real-world highly imbalanced tasks while being very computationally efficient. Comparing with the existing methods, `SPE` is accurate, fast, robust, and adaptive.
- We introduce the concept of classification hardness. By considering the distribution of classification hardness over the dataset, the learning procedure of our proposed framework `SPE` is automatically optimized in a model-specific way. Unlike prevailing methods, our learning framework does not require any pre-defined distance metrics which is usually unavailable in real-world scenarios.

II. Problem Definition

In this section, we first describe the class imbalance problem considered in this paper. Then we give some necessary symbol definition and show the evaluation criteria that are commonly used in imbalanced scenarios.

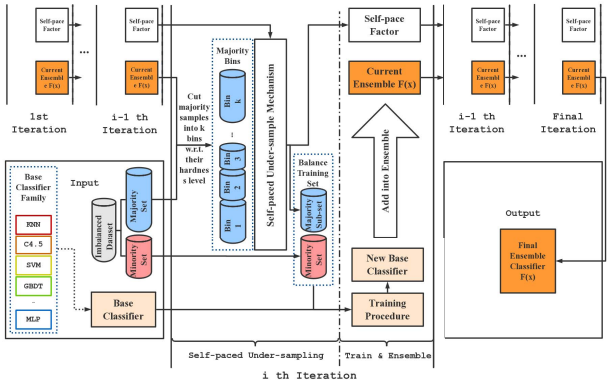


Fig. 1. Self-Paced Ensemble Process.

Class imbalance: A dataset is said to be imbalanced whenever the number of instances from the different classes is not nearly the same. Class imbalance exists in the nature of various real-world applications, like medicine (sick vs. healthy), fraud detection (normal vs. fraud), or click-through-rate prediction (clicked vs. ignored). The uneven distribution poses a difficulty for applying canonical learning algorithms on imbalanced dataset, as they will be biased towards the majority group due to their accuracy-oriented design. Despite such problem has been extensively studied, in real applications, class imbalance often co-exists with other difficulty factors, such as enormous data scale, noises, and missing values. Therefore, the performances of existing methods are still unsatisfactory.

Symbol definition: In this paper, we only consider the binary situation that exists widely in practical applications [2], [9], [10]. In binary imbalance classification, only two classes were considered: the *minority* class with less samples and the *majority* class with relatively more samples. For simplicity, in this paper we always let the minority class to be positive class and the majority class to be negative. We use $\mathcal{D}$ to denote the collection of all training samples (x, y). The minority class set $\mathcal{P}$ and majority class set $\mathcal{N}$ are then defined as:

$$\mathcal{P} = \{(x, y) \mid y = 1\},\ \mathcal{N} = \{(x, y) \mid y = 0\}$$

Therefore, we have $|\mathcal{N}| \gg |\mathcal{P}|$ for (highly) imbalanced problems. In order to uniformly describe the level of class imbalance in different datasets, we consider the Imbalance Ratio (IR), which is defined as the number of majority class examples divided by the number of minority class examples:

$$\textit{Imbalanced Ratio (IR)} = \frac{n_{majority}}{n_{minority}} = \frac{|\mathcal{N}|}{|\mathcal{P}|}$$

Evaluation criteria: Since the accuracy does not well reflect the model performance, we usually adopt the other evaluation criteria based on the number of true / false positive / negative prediction. Under the binary scenario, the results of the correctly and incorrectly recognized examples of each class can

be recorded in a confusion matrix. Table I shows the confusion matrix for binary classification.

TABLE I
CONFUSION MATRIX FOR BINARY CLASSIFICATION.

Label \ Predict	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

For evaluating the performance on minority class, recall and precision are commonly used. Furthermore, we also consider the F1-score, G-mean (i.e., harmonic / geometric mean of precision and recall) [22], [23], MCC (i.e., Matthews correlation coefficient) [24], and AUCPRC (i.e., the area under the precision-recall curve) [25].

- $Recall = \frac{TP}{TP+FN}$
- $Precision = \frac{TP}{TP+FP}$
- *F1-score* = $2 \cdot \frac{Recall \times Precision}{Recall+Precision}$
- *G-mean* = $\sqrt{Recall \cdot Precision}$
- *MCC* = $\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$
- *AUCPRC = Area Under Precision-Recall Curve*

III. LIMITATIONS OF EXISTING METHODS

In this section, we give a brief introduction to existing imbalance learning solutions, and discuss why they obtain unsatisfactory performance on the real-world industrial tasks. To solve the class imbalance problem, researchers have proposed a variety of methods. This research field is also known as *imbalance learning*. As mentioned in the introduction, we categorize existing imbalance learning methods into three groups: *Data-level*, *Algorithm-level* and *Ensemble*.

Data-level Methods: This group of methods concentrates on modifying the training set to make it suitable for a standard learning algorithm. With respect to balancing distributions, data-level methods can be categorized into three groups:

- Under-sampling approaches that remove samples from the majority class (e.g., [12], [26], [27]).
- Over-sampling approaches that generate new objects for the minority class (e.g., [13], [14], [28]).
- Hybrid-sampling approaches that combine two methods above (e.g., [29], [30]).

Standard random re-sampling methods often lead to removal of important samples or introduction of meaningless new objects. Therefore, more advanced methods were proposed that try to maintain structures of groups and/or generate new data according to underlying distributions. They apply k-Nearest Neighborhood (k-NN) algorithm [31] to extract underlying distribution in the feature space, and use that information to guide their re-sampling.

However, the application of k-NN algorithm requires predefined distance metric, which is usually unavailable in the real-world datasets since they may contain categorical features and/or missing values. k-NN algorithm is also easily disturbed by noises thus unable to reveal the underlying distribution for re-sampling methods when the dataset is noisy. Moreover, the computational cost of applying k-NN grows quadratically with the size of the dataset. Thus running such distance-based re-sampling methods on large-scale datasets can be extremely slow.

Algorithm-level Methods: This group of methods concentrates on modifying existing learners to alleviate their bias towards majority groups. It requires good insight into the modified learning algorithm and precise identification of reasons for its failure in mining skewed distributions. The most popular algorithm-level method is cost-sensitive learning [15], [16]. By assigning large cost to minority instances and small cost to majority instances, it boosts minority importance during the learning process to alleviate the bias towards majority class.

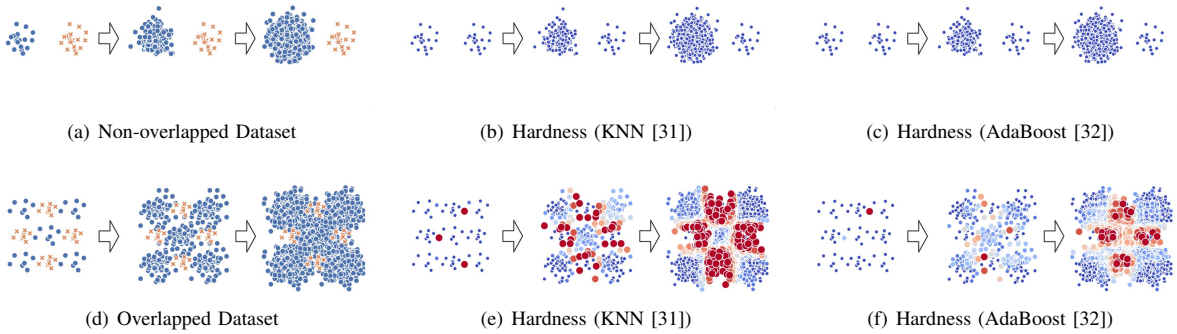
It must be noted that the cost matrix on a specific task is given by domain expert before-hand, which is usually unavailable in many real-world problems. Even if one has the domain knowledge required for setting the cost, such cost matrix is usally designed for specific tasks and do not generalize across different classification tasks. On the other hand, for the batch training models such as neural networks, the positive (minority) samples are only contained in a few batches. Even if we apply cost-sensitive into the training process, the model still soon stuck into local minima.

Ensemble Methods: This group of methods concentrates on merging one of the data-level or algorithm-level solutions with an ensemble learning method to get a robust and strong classifier. Ensemble approaches are gaining more popularity in real-world applications for their good performance on imbalanced tasks. Most of them are based on a canonical ensemble learning algorithm with an imbalance learning algorithm embedded in the pipeline, e.g., SMOTE [13] + Adaptive Boosting [32] = SMOTEBoost [33]. Some other ensemble methods introduce another ensemble classifier as their base learner, e.g., EasyEnsemble [18] trains multiple AdaBoost classifier to form its final ensemble.

However, those ensemble-based methods suffer from low efficiency, poor applicability and high sensitivity to noise when applying on realistic imbalanced tasks, since they still have those data-level/algorithm-level imbalance learning methods in their pipeline. There are few methods carried out preliminary exploration of using training feedback information to perform dynamic re-sampling on imbalance datasets. However, such methods do not take full account of the data distribution. For instance, BalanceCascade iteratively discards majority samples that were well-classified by the current classifier. It may result in overweighting outliers in late iterations and finally deteriorate the ensemble.

IV. CLASSIFICATION HARDNESS DISTRIBUTION

Before we describe our algorithm, we introduce the concept of the "classification hardness" in this section. We explain the benefits of considering hardness distribution into imbalance learning framework. We also present an intuitive visualization in Fig. 2 to help understand the relationship between hardness, imbalance ratio, class overlapping and model capacity.



(a) Non-overlapped Dataset (b) Hardness (KNN [31]) (c) Hardness (AdaBoost [32])

(d) Overlapped Dataset (e) Hardness (KNN [31]) (f) Hardness (AdaBoost [32])

Fig. 2. Comparison of overlapped / non-overlapped dataset under different level of class imbalance. (a)(c) shows the original datasets, (b)(e) are the hardness w.r.t. KNN classifier, and (e)(f) are the hardness w.r.t. AdaBoost classifier.

Definition: We use the symbol $\mathcal{H}$ to denote the classification hardness function, where $\mathcal{H}$ can be any "decomposable" error function, i.e., the overall error is calculated by the summation of individual sample errors. Examples include Absolute Error, Squared Error (Brier-score) and Cross Entropy. Suppose F is a trained classifier, we use $F(x)$ to denote the classifier's output probability of x being a positive instance. Then the classification hardness of sample (x, y) with respect to F is given by the function $\mathcal{H}(x, y, F)$.

Advantages: The concept of the classification hardness has two advantages under the imbalance classification scenario:

- First, it fills the gap between the imbalance ratio and the task difficulty. As mentioned in the introduction, even with the same imbalance ratio, different tasks could demonstrate extremely different difficulties. We show a detailed example in Fig. 2. In Fig. 2(a), the dataset is generated with two disjoint Gaussian components. The growth of the imbalance ratio does not affect much of the task hardness. While in Fig. 2(d) the dataset is generated by several overlapped Gaussian components. As the imbalance ratio grows, it varies from an easy classification task to an extremely hard task. However, the imbalance ratio could not well reflect such task hardness. Instead, we show the classification hardness of those two datasets based on different classifiers. As the imbalance ratio grows, the quantity of the hard samples increases sharply in Fig. 2(e) and Fig. 2(f), while stays constant in Fig. 2(b) and Fig. 2(c). Thus, the classification hardness carries more information about the underlying data structure and better indicates the current task hardness.
- Second, the classification hardness also fills the gap between data sampling strategy and the classifiers' capacity. Most of the existing sampling method totally ignores the capacity of the base classifier. However, different classifiers usually demonstrate very different performances on the imbalanced data classification. For example, in Fig.2, KNN and Adaboost show very different hardness distribution for the same dataset. It is beneficial to consider the model capacity when performing under-sampling. Using the classification hardness, our framework is able to optimize any kind of classifier's final performance in a model-specific way.

Types of Data Samples: Intuitively, we distinguish three kinds of data samples, i.e., *trivial*, *noise* and *borderline* samples according to their corresponding hardness values:

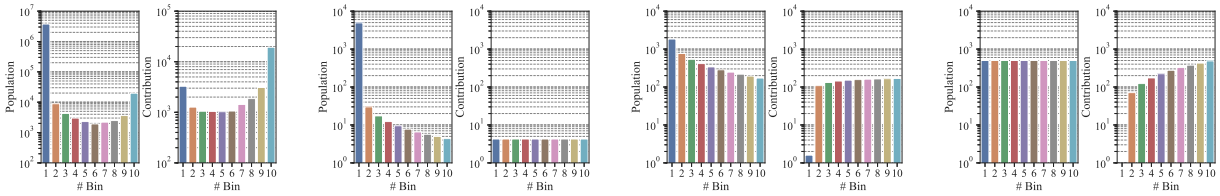
- Most of the data samples are *trivial* samples and can be well-classified by the current model, e.g., the blue samples in Fig. 2(e) and Fig. 2(f). Each of the trivial samples only contributes tiny hardness. However, the overall contribution is non-negligible due to its large population. For such kind of samples, we only need to keep a small proportion of them to represent the "skeleton" of their corresponding distribution in order to prevent overfitting, then drop most of them since they have already been well-learned.
- On the contrary, there are also several *noise* samples, e.g., the dark red samples in Fig. 2. Despite their small population, each of them contributes a large hardness value. Thus, the total contribution can be very huge. We stress that these noise samples are usually caused by the indistinguishable overlapping or outliers since they exist stably even when the model is converged. Enforcing model to learn such samples could lead to serious overfitting.
- For the rest samples, here we simply classify them as the *borderline* samples. The borderline samples are the most informative data samples during the training. For example, as we can see, in Fig. 2, the light red points are very close to the decision boundary of the current model. Enlarging the weights of those borderline samples is usually helpful to further improve the model performance.

The above discussion provides us with an intuition to distinguish different data samples. However, since it is hard to make such an explicit distinction in practice, we alternatively categorize the data samples in a "soft" way, as described in the next section.

V. Self-paced Ensemble

We now describe *Self-paced Ensemble*[1] (`SPE`), our framework for massive imbalance classification. Firstly, we demonstrate the ideas of hardness harmonize and self-paced factor. After that, we summarize the `SPE` procedure in Algorithm 1.

[1] `github.com/ZhiningLiu1998/self-paced-ensemble`



(a) Original majority set $\mathcal{N}$ (b) $\alpha = 0$ (c) $\alpha = 0.1$ (d) $\alpha \to \infty$

Fig. 3. An example to visualize how self-paced factor α controls the self-paced under-sampling. The left part of each subfigure shows the number of samples in each bin, the right part shows the hardness contribution from each bin. Subfigure (a) is the distribution over all majority instances. (b)(c)(d) are the distribution over subsets under-sampled by our mechanism when $\alpha = 0$, $\alpha = 0.1$, and $\alpha \to \infty$, respectively. Note that the y-axis uses log scale since the number of samples within different hardness bins can differ by orders of magnitude.

A. Self-paced Under-sampling

Motivated by previous observations, we aim to design an under-sampling mechanism that reduces the effect of trivial and noise samples, while enlarges the importance of the borderline samples as we expected. Therefore, we introduce the concept of "hardness harmonize" and a self-paced training procedure, to achieve such goal.

1) Hardness Harmonize: We split all the majority samples into k bins according to their hardness values, where k is a hyper-parameter. Each bin indicates a specific hardness level. We then under-sample the majority instances into a balanced dataset by keeping the total hardness contribution in each bin as the same. Such method is so-called "harmonize" in the gradient-based optimization literature [34], where they harmonize the gradient contribution in batch training of neural networks. In our case, we adopt a similar idea to harmonize the hardness in the first iteration.

However, we do not simply use the hardness harmonize in all the iterations. The main reason is that the population of trivial samples grows during the training process since the ensemble classifier will gradually fit the training set. Hence, simply harmonizing the hardness contribution still leaves a lot of trivial samples (Fig. 3(b)). Those samples greatly slow down the learning procedure in the later iterations since they are less informative. Instead, we introduce the "self-pace factor" to perform self-paced harmonize under-sampling.

2) Self-paced Factor: Specifically, start from harmonizing the hardness contribution of each bin, we gradually decrease the sample probability of those bins with a large population. The decreasing level is controlled by a self-paced factor α. When α goes large, we focus more on the harder samples instead of the simple hardness contribution harmonize. In the first few iterations, our framework mainly focuses on those informative borderline samples, thus the outliers and noises do not affect much of the generalization ability of our model. In the later iterations where α is very large, our framework still keeps a reasonable fraction of trivial (high confidence) samples as the "skeleton", which effectively prevents our framework from overfitting. Fig. 3 shows the self-paced under-sampling process of a real-world large-scale dataset[2].

[2]Payment Simulation dataset, statistics can be found in Table III.

B. Algorithm Formalization

Finally, in this subsection, we describe our algorithm formally. Recall that in Section 2, we use $\mathcal{D}$ to denote the collection of all training samples (x, y). $\mathcal{N}$ / $\mathcal{P}$ is the majority / minority set in $\mathcal{D}$. We use $\mathcal{D}_{dev}$ to denote the validation set, which is used to measure the generalization ability of the ensemble model. Note that $\mathcal{D}_{dev}$ keeps the original imbalanced distribution with no re-sampling. Moreover, we use B_ℓ to denote the ℓ-th bin, where B_ℓ is defined as

$$B_\ell = \{(x, y) \mid \frac{\ell - 1}{k} \leq \mathcal{H}(x, y, F) < \frac{\ell}{k}\} \; w.l.o.g. \; \mathcal{H} \in [0, 1]$$

The details are shown in Algorithm 1. Notice that we update hardness value in each iteration (line 4-5) in order to select data samples that were most beneficial for the current ensemble. We use the $\tan$ function (line 7) to control the growth of self-paced factor α. Thus we have $\alpha = 0$ in the first iteration and $\alpha \to \infty$ in the last iteration.

Algorithm 1: Self-paced Ensemble

Input: Training set $\mathcal{D}$, hardness function $\mathcal{H}$, base classifier f, number of base classifiers n, number of bins k,

1 **Initialize:** $\mathcal{P} \Leftarrow$ minority in $\mathcal{D}$, $\mathcal{N} \Leftarrow$ majority in $\mathcal{D}$
2 Train classifier f_0 using random under-sample majority subsets $\mathcal{N}'_0$ and $\mathcal{P}$, where $|\mathcal{N}'_0| = |\mathcal{P}|$.
3 **for** *i=1 to* n **do**
4 Ensemble $F_i(x) = \frac{1}{i} \sum_{j=0}^{i-1} f_j(x)$
5 Cut majority set into k bins w.r.t. $\mathcal{H}(x, y, F_i)$: $B_1, B_2, \cdots, B_k$
6 Average hardness contribution in ℓ-th bin: $h_\ell = \sum_{s \in B_\ell} \mathcal{H}(x_s, y_s, F_i)/|B_\ell|, \; \forall \ell = 1, \ldots, k$
7 Update self-paced factor $\alpha = tan(\frac{i\pi}{2n})$
8 Unnormalized sampling weight of ℓ-th bin: $p_\ell = \frac{1}{h_\ell + \alpha}, \forall \; \ell = 1, \ldots, k$
9 Under-sample from ℓ-th bin with $\frac{p_\ell}{\sum_m p_m} \cdot |\mathcal{P}|$ samples
10 Train f_i using newly under-sampled subset
11 **end**
12 **return** *final ensemble* $F(x) = \frac{1}{n} \sum_{m=1}^{n} f_m(x)$

TABLE II
GENERALIZED PERFORMANCE (AUCPRC) ON CHECKER BOARD DATASET.

Model	Hyper	RandUnder	Clean	SMOTE	$\texttt{Easy}_{10}$	$\texttt{Cascade}_{10}$	$\texttt{SPE}_{10}$
KNN	k_neighbors=5	0.281±0.003	0.382±0.000	0.271±0.003	0.411±0.003	0.409±0.005	**0.498**±0.004
DT	max_depth=10	0.236±0.010	0.365±0.001	0.299±0.007	0.463±0.009	0.376±0.052	**0.566**±0.011
MLP	hidden_unit=128	0.562±0.017	0.138±0.035	0.615±0.009	0.610±0.004	0.582±0.005	**0.656**±0.005
SVM	C=1000	0.306±0.003	0.405±0.000	0.324±0.002	0.386±0.001	0.456±0.010	**0.518**±0.004
AdaBoost_{10}	n_estimator=10	0.226±0.019	0.362±0.000	0.297±0.004	0.487±0.017	0.391±0.013	**0.570**±0.008
Bagging_{10}	n_estimator=10	0.273±0.002	0.401±0.000	0.316±0.003	0.436±0.004	0.389±0.007	**0.568**±0.005
RandForest_{10}	n_estimator=10	0.260±0.004	0.229±0.000	0.306±0.011	0.454±0.005	0.402±0.012	**0.572**±0.003
GBDT_{10}	boost_rounds=10	0.553±0.015	0.602±0.000	0.591±0.008	0.645±0.006	0.648±0.009	**0.680**±0.003

VI. EXPERIMENTS & ANALYSIS

In this section, we present the results of our experimental study on one synthetic and five real-world extremely imbalanced datasets. We tested the applicability of our proposed algorithm to incorporate with different kinds of base classifiers. We also show some visualizations to help understand the difference between our proposed method and the other imbalance learning methods. We evaluated the experiment results with multiple criteria, and demonstrate the strength of our proposed framework.

A. Synthetic Dataset

To provide more insights of our framework, we first show the experimental results on the synthetic dataset. We create a 4×4 *checkerboard* dataset to validate our method. The dataset contains 16 Gaussian components. All Gaussian components share the same covariance matrix of $0.1 \times \mathbf{I}_2$. We set the number of minority samples $|\mathcal{P}|$ as $1,000$, and the number of majority $|\mathcal{N}|$ as $10,000$. The training set $\mathcal{D}$, validation set $\mathcal{D}_{dev}$ and test set $\mathcal{D}_{test}$ were independently sampled from same original distribution. See Fig. 4 for an example.

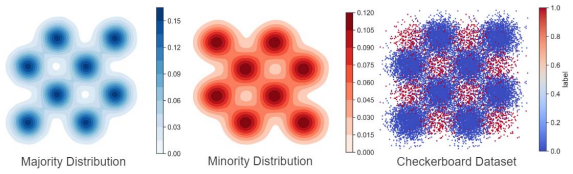


Fig. 4. An example of *checkerboard* dataset. Blue dots represent the majority class samples, red ones represent the minority class samples.

1) Setup Details: We compared our proposed method SPE[3] with following imbalance learning approaches:

- RandUnder (*Random Under-sampling*) randomly under-sample the majority class to get a subset $\mathcal{N}'$ such that $|\mathcal{N}'| = |\mathcal{P}|$. The set $\mathcal{N}' \cup \mathcal{P}$ was then used for training.
- Clean (*Neighbourhood Cleaning Rule based under-sampling*) [27] removes a majority instance if most of its neighbors come from another class.
- SMOTE (*Synthetic Minority Over-sampling TechniquE*) [13] generates synthetic minority instances between existing minority samples until the dataset is balanced.

[3]In our implementation of SPE, we set the number of bins $k = 20$, and use absolute error as the classification hardness, i.e., $\mathcal{H}(x, y, F) = |F(x) - y|$, unless otherwise stated.

- Easy (*EasyEnsemble*) [18] utilizes RandUnder to train multiple AdaBoost [32] models and combine their outputs.
- Cascade (*BalanceCascade*) [18] extends Easy by iteratively drop majority examples that were already well classified by current base classifier.

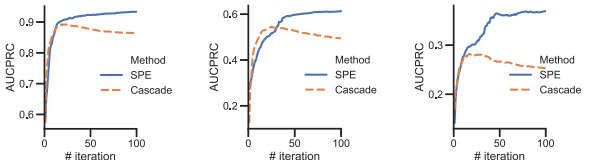
In addition, according to our aforementioned discussion in the Classification Hardness section, by considering the hardness distribution our proposed framework SPE is able to work with any kind of classifiers and optimize the final performance in a model-specific way. Hence, we introduce 8 canonical classifiers in order to test the effectiveness and applicability of different imbalance learning methods:

- *K-Nearest Neighbors* (KNN) [31]
- *Decision Tree* (DT) [6]
- *Support Vector Machine* (SVM) [7]
- *Multi-Layer Perceptron* (MLP) [8]
- *Adaptive Boosting* (AdaBoost) [32]
- *Bootstrap aggregating* (Bagging) [35]
- *Random Forest* (RandForest) [36]
- *Gradient Boosting Decision Tree* (GBDT) [37]

We apply imbalanced-learn [38] package to implement aforementioned imbalance learning methods, and scikit-learn [39], LightGBM [40], Pytorch [41] packages to implement the canonical classifiers. We use subscripts to denote the number of base models in an ensemble classifier, e.g., $\texttt{Easy}_{10}$ indicates Easy with 10 base models. Due to space limitation, we only present the experimental results of AUCPRC in this experiment, other metrics will be used in following extensive experiments on real-world datasets.

2) Results on synthetic dataset: Table II lists the results on checkerboard task. Note that to reduce randomness, we show the mean and standard deviation of 10 independent runs. We also list the hyper-parameters we used for each base classifier. From the Table II we can observe that:

- SPE consistently outperform other methods on the checkerboard dataset using 8 different classifiers.
- Distance-based re-sampling lead to poor results when cooperating with specific classifiers, e.g., SMOTE+KNN, Clean+RandForest. We argue that the ignorance of difference in model capacity is the main reason that causes invalidity to those re-sample methods.
- Comparing with other methods, ensemble methods Easy and Cascade obtain better and more robust performance but still worse than our proposed ensemble framework SPE.

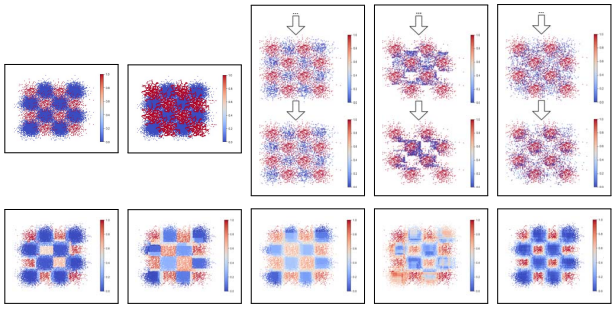


(a) $cov = 0.05$ (b) $cov = 0.10$ (c) $cov = 0.15$

Fig. 5. Training curve under different level of overlap.

3) Robustness under Class Overlapping: Furthermore, we test the robustness of SPE, when the Gaussian components have different levels of overlapping. We control the components overlapping by replacing the original covariance matrix from $0.1 \times \mathbf{I}_2$ to $0.05 \times \mathbf{I}_2$ and $0.15 \times \mathbf{I}_2$. The distribution is less overlapped when the covariance factor in covariance matrix is smaller, and more overlapped when it is bigger. We keep the size and imbalance ratio to be the same, and sample three different checkerboard datasets respectively. Fig. 5 shows how the AUCPRC (on test set) changes within training process:

- The level of distribution overlapping significantly influences the classification performance, even though the size and imbalance ratio of all datasets are totally the same.
- As the overlapping aggravates, the performance of Cascade shows more obvious downward trend in later iterations. The reason behind is that Cascade inclines to overfit the noise samples, while SPE can alleviate this issue by keeping a reasonable proportion of trivial and borderline samples.



(a) Clean (b) SMOTE (c) Easy (d) Cascade (e) SPE

Fig. 6. Visualization of training set (upper, blue/red dot indicates a sample from majority/minority class) and predict probability (lower, blue/red dot indicates the classifier tend to classify a sample as majority/minority class) on *checkerboard* dataset. Note that ensemble methods Easy, Cascade and SPE train multiple base models in each iteration with different training sets, so we show training sets of 5_{th} and 10_{th} model in their pipeline.

4) Intuitive Visualization: We give a visualization in Fig. 6 to show how the aforementioned imbalance learning approaches train / predict on checkerboard dataset.

As we can see, Clean tries to clean up the majority outliers who were surrounded by minority data points, however, it retains all the trivial samples so that the learning model cannot focus on more informative data. SMOTE over-generalizes minority class due to indistinguishable overlapping. Easy performs simple random under-sampling and thus part of majority samples are dropped which causes the information loss. Cascade keeps many outliers in late iterations. Those outliers finally lead to bad generalization. By contrast, SPE gets a much more accurate and robust results by considering the classification hardness distribution over the dataset.

B. Real-world Datasets

We choose several real-life datasets with highly skewed class distribution to assess the effectiveness of our proposed learning framework on realistic tasks.

Credit Fraud contains transactions made by credit cards in September 2013 by European card-holders [2]. The task is to detect frauds from credit card transaction records. It is a highly imbalanced dataset with only 492 frauds out of 284,807 transactions, which brings a high imbalance ratio of 578.88:1. Payment Simulation is a large-scale dataset with 6.35 million instances. It simulates mobile money transactions based on a sample of real transactions extracted from one month of financial logs from a mobile money service implemented in an African country. Similarly, it has 8,213 frauds out of 6,362,620 transactions and a high imbalance ratio 773.70:1. Record Linkage is a dataset of element-wise comparison of records with personal data from a record linkage setting. The task requires us to decide whether the underlying records belong to one person. The underlying records stem from the epidemiological cancer registry of the German state of North Rhine-Westphalia, which has 5,749,132 record pairs, and 20,931 of them are matches. KDDCUP-99 contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between "bad" connections, called intrusions or attacks, and "good" normal connections. It is a multi-class task with 4 main categories of attacks: DOS, R2L, U2R and probing (PRB). We formed 2 two-class imbalanced problems by taking the majority class (i.e., DOS) and a minority class (i.e., PRB and R2L), namely, KDDCUP (DOS vs. PRB) and KDDCUP (DOS vs. R2L).

Table III lists the statistics of each dataset.

1) Setup Details: For each real-world task, we use 60% of the full data as the training set $\mathcal{D}$ and 20% as the validation set $\mathcal{D}_{dev}$ (some classifiers like GBDT need validation set for early stopping), the rest 20% is then used as the test set $\mathcal{D}_{test}$. All results in this section were evaluated on the test set in order to test the classifier's generalized performance.

2) Results on Real-world Datasets: We first extend the previous experiment on synthetic data to realistic datasets that we mentioned above. Table IV lists the experimental results of applying 6 different imbalance learning approaches (i.e., RandUnder, Clean, SMOTE, Easy, Cascade, and SPE) combine with 5 different canonical classification algorithms (i.e., KNN, DT, MLP, AdaBoost$_{10}$, and GBDT$_{10}$) on 5 real-

TABLE III
STATISTICS OF THE REAL-WORLD DATASETS

Dataset	#Attribute	#Sample	Feature Format	Imbalance Ratio	Model
Credit Fraud	31	284,807	Numerical	578.88:1	KNN, DT, MLP
KDDCUP (DOS vs. PRB)	42	3,924,472	Integer & Categorical	94.48:1	AdaBoost_{10}
KDDCUP (DOS vs. R2L)	42	3,884,496	Integer & Categorical	3448.82:1	AdaBoost_{10}
Record Linkage	12	5,749,132	Numerical & Categorical	273.67:1	GBDT_{10}
Payment Simulation	11	6,362,620	Numerical & Categorical	773.70:1	GBDT_{10}

TABLE IV
GENERALIZED PERFORMANCE ON 5 REAL-WORLD DATASETS.

Dataset	Model	Metric	RandUnder	Clean	SMOTE	Easy_{10}	Cascade_{10}	SPE_{10}
Credit Fraud	KNN	AUCPRC	0.052±0.002	0.677±0.000	0.352±0.000	0.162±0.012	0.676±0.015	**0.752**±0.018
		F1	0.112±0.007	0.821±0.000	0.559±0.001	0.250±0.020	0.792±0.023	**0.843**±0.016
		GM	0.228±0.009	0.822±0.000	0.593±0.001	0.399±0.025	0.810±0.001	**0.852**±0.002
		MCC	0.222±0.014	0.822±0.000	0.592±0.001	0.650±0.004	0.815±0.006	**0.855**±0.006
	DT	AUCPRC	0.014±0.001	0.598±0.013	0.088±0.011	0.339±0.039	0.592±0.029	**0.783**±0.015
		F1	0.032±0.002	0.767±0.004	0.177±0.006	0.478±0.021	0.737±0.023	**0.838**±0.021
		GM	0.119±0.003	0.778±0.006	0.303±0.017	0.548±0.048	0.749±0.011	**0.843**±0.007
		MCC	0.124±0.001	0.780±0.008	0.310±0.003	0.409±0.015	0.778±0.049	**0.831**±0.008
	MLP	AUCPRC	0.225±0.050	0.001±0.000	0.527±0.017	0.605±0.016	0.738±0.009	**0.747**±0.011
		F1	0.388±0.047	0.003±0.000	0.725±0.013	0.762±0.023	0.803±0.004	**0.811**±0.010
		GM	0.494±0.040	0.040±0.000	0.665±0.060	0.748±0.019	0.806±0.007	**0.828**±0.003
		MCC	0.178±0.008	0.000±0.000	0.718±0.006	0.705±0.004	0.744±0.046	**0.826**±0.008
KDDCUP (DOS vs. PRB)	AdaBoost_{10}	AUCPRC	0.930±0.012	- - -	- - -	0.995±0.002	**1.000**±0.000	**1.000**±0.000
		F1	0.962±0.001	- - -	- - -	0.997±0.000	**0.999**±0.000	**0.999**±0.000
		GM	0.964±0.001	- - -	- - -	0.997±0.001	0.998±0.000	**0.999**±0.000
		MCC	0.956±0.004	- - -	- - -	0.992±0.001	0.993±0.003	**0.999**±0.000
KDDCUP (DOS vs. R2L)	AdaBoost_{10}	AUCPRC	0.034±0.005	- - -	- - -	0.108±0.011	0.945±0.005	**0.999**±0.001
		F1	0.050±0.005	- - -	- - -	0.259±0.058	0.965±0.005	**0.991**±0.003
		GM	0.164±0.011	- - -	- - -	0.329±0.015	0.967±0.008	**0.988**±0.004
		MCC	0.175±0.016	- - -	- - -	0.214±0.004	0.905±0.056	**0.986**±0.004
Record Linkage	GBDT_{10}	AUCPRC	0.988±0.011	- - -	- - -	0.999±0.000	**1.000**±0.000	**1.000**±0.000
		F1	0.995±0.000	- - -	- - -	0.996±0.000	**0.998**±0.000	**0.998**±0.000
		GM	0.994±0.002	- - -	- - -	0.996±0.000	**0.998**±0.000	**0.998**±0.000
		MCC	0.780±0.000	- - -	- - -	0.884±0.000	0.940±0.000	**0.998**±0.000
Payment Simulation	GBDT_{10}	AUCPRC	0.278±0.030	- - -	- - -	0.676±0.058	0.776±0.004	**0.944**±0.001
		F1	0.446±0.030	- - -	- - -	0.709±0.021	0.851±0.003	**0.885**±0.001
		GM	0.530±0.020	- - -	- - -	0.735±0.011	0.851±0.001	**0.885**±0.001
		MCC	0.290±0.023	- - -	- - -	0.722±0.015	0.856±0.002	**0.876**±0.001

world classification tasks[4]. The performance was evaluated by 4 criterions (i.e., AUCPRC, F1-score, G-mean, and MCC) on the test set. For reduce the effect of randomness, we show the mean and standard deviation of 10 independent runs:

- `SPE` demonstrates the best performance on all tested real-world tasks using 5 classifiers over 4 evaluation criteria.
- `Clean` + MLP performs poorly on Credit Fraud task since `Clean` only cleans up noises and does not guarantee a balanced dataset. As described above, the batch training method will fail when the class distribution is skewed.
- `RandUnder` and $\texttt{Easy}_{10}$ use randomized under-sampling to get a small majority subset for training. They suffer from severe information loss and high potential variance when applying on highly imbalanced dataset.

Some results of `Clean` and `SMOTE` are missing in Table IV due to lack of appropriate distance metric and unacceptable computational cost. Take the KDDCUP (DOS vs. PRB) dataset as an example, from our experiment, `Clean` needs more than 8 hours to calculate the distance between each data sample. Similarly, `SMOTE` generates millions of synthetic samples that further enlarge the scale of the training set.

[4]Due to space limitation, Table 5 only lists some most representative results. See `github.com/ZhiningLiu1998/self-paced-ensemble` for additional experimental results on more datasets and classifiers.

C. Extensive Experiments on Real-world Datasets

We further introduce some other widely used re-sampling and ensemble-based imbalance learning methods for a more comprehensive comparison. By showing supplementary information, e.g., the number of samples used for training and the processing time, we demonstrate the efficiency of different methods on real-world highly imbalanced tasks.

1) Comparison with Re-sampling Methods: 9 more re-sampling based imbalance learning methods were introduced, including 5 under-sampling methods, 3 over-sampling methods and 2 hybrid-sampling methods (see Table V):

- `NearMiss` [11] selects $|\mathcal{P}|$ samples from the majority class for which the average distance of the k nearest samples of the minority class is the smallest.
- `ENN` (*Edited Nearest Neighbor*) [42] aims to remove noisy samples from the majority class for which their class differs from the one of their nearest-neighbors.
- `TomekLink` [12] removes majority samples by detecting "TomekLinks". A TomekLink exists if two samples of different class are the nearest neighbors of each other.
- `AllKNN` [43] extends `ENN` by repeating the algorithm multiple times, the number of neighbors of the internal nearest neighbors algorithm is increased at each iteration.
- `OSS` (*One Side Sampling*) [26] makes use of `TomekLink`

TABLE V
GENERALIZED PERFORMANCE (AUCPRC) OF 12 DIFFERENT RE-SAMPLING METHODS.

Category	Method	LR	KNN	DT	AdaBoost$_{10}$	GBDT$_{10}$	#Sample	Re-sampling Time(s)
No re-sampling	ORG	0.587±0.001	0.721±0.000	0.632±0.011	0.663±0.026	0.803±0.001	170885	- - -
Under-sampling	RandUnder	0.022±0.008	0.068±0.000	0.011±0.001	0.013±0.000	0.511±0.096	632	0.07
	NearMiss	0.003±0.003	0.010±0.009	0.002±0.000	0.002±0.001	0.050±0.000	632	2.06
	Clean	0.741±0.018	0.697±0.010	0.727±0.029	0.698±0.032	0.810±0.003	170,680	428.88
	ENN	0.692±0.036	0.668±0.013	0.637±0.021	0.644±0.026	0.799±0.004	170,779	423.86
	TomekLink	0.699±0.050	0.650±0.031	0.671±0.018	0.659±0.027	0.814±0.007	170,865	270.09
	AllKNN	0.692±0.041	0.668±0.012	0.652±0.023	0.652±0.015	0.808±0.002	170,765	1066.48
	OSS	0.711±0.056	0.650±0.029	0.671±0.025	0.666±0.009	0.825±0.016	163,863	240.95
Over-sampling	RandOver	0.052±0.000	0.532±0.000	0.051±0.001	0.561±0.010	0.706±0.013	341,138	0.14
	SMOTE	0.046±0.001	0.362±0.005	0.093±0.002	0.087±0.005	0.672±0.026	341,138	1.23
	ADASYN	0.017±0.001	0.360±0.004	0.031±0.001	0.035±0.007	0.496±0.081	341,141	1.87
	BorderSMOTE	0.067±0.006	0.579±0.010	0.145±0.003	0.126±0.011	0.242±0.020	341,138	1.89
Hybrid-sampling	SMOTEENN	0.045±0.001	0.329±0.006	0.084±0.004	0.074±0.012	0.665±0.017	340,831	478.36
	SMOTETomek	0.046±0.001	0.362±0.004	0.094±0.004	0.094±0.004	0.682±0.033	341,138	293.75
Under-sampling + Ensemble	SPE$_{10}$	**0.755**±0.003	**0.767**±0.001	**0.783**±0.015	**0.808**±0.004	**0.849**±0.002	632×10	0.116×10

by running it multiple times to iteratively decide if a sample should be kept in a dataset or not.

- RandOver (Random Over-sampling) randomly repeats some minority samples to balance the class distribution.
- ADASYN (*ADAptive SYNthetic over-sampling*) [14] focuses on generating samples next to the original samples which are wrongly classified using a k-nearest neighbors classifier.
- BorderSMOTE (*Borderline Synthetic Minority Over-sampling TechniquE*) [28] offers a variant of the SMOTE algorithm, where only the borderline examples could be seeds for over-sampling.
- SMOTEENN (*SMOTE with Edited Nearest Neighbours cleaning*) [30] utilizes ENN as the cleaning method after applying SMOTE over-sampling to obtain a cleaner space.
- SMOTETomek (*SMOTE with Tomek links cleaning*) [29] uses TomekLink instead of ENN as the cleaning method.

As mentioned before, running some of these re-sampling methods on large-scale datasets can be extremely slow. It is also hard to define an appropriate distance metric on a dataset with non-numerical features. With these considerations, we apply all methods on the Credit Fraud dataset. This dataset has 284,807 samples, and only contains normalized numerical features, which enables all distance-based re-sampling methods to achieve their maximum potential. Thus we can fairly compare the pros and cons of different methods.

We use 5 different classifiers, i.e., Logistic Regression (LR), KNN, DT, AdaBoost$_{10}$, and GBDT$_{10}$, to collaborate with: ORG which refers to train classifier over the original training set with no re-sampling, 12 re-sampling methods which refer to train classifier on the re-sampled training set, and SPE which refers to use our proposed method to get an ensemble of the given classifier. We also list the number of examples that used for training and the time it takes to perform re-sampling for each method. All aforementioned re-sampling methods were implemented using imbalanced-learn Python package 0.4.3 [38] with Python 3.7.1, and run on an Intel Core i7-7700K CPU with 16 GB RAM. Experimental results were shown in Table V:

- SPE significantly boosts the performance of various canonical classification algorithms on highly imbalanced dataset. Comparing with other re-sampling methods, it only requires very little training data and short processing time to achieve such effects.
- Most methods can only obtain reasonable results (better than ORG) when cooperating with specific classifiers. For instance, TomekLink works well with LR, DT, and GBDT but fails to boost the performance of KNN and AdaBoost. The reason behind is that they perform model-agnostic re-sampling without considering classifier's capacity.
- On a dataset with such high imbalance ratio (IR=578.88:1), the minority class is often poorly represented and lacks a clear structure. Therefore, straightforward application of re-sampling, especially over-sampling that rely on relations between minority objects can actually deteriorate the classification performance, e.g., advanced over-sampling method SMOTE perform even worse than RandOver and ORG.

2) Comparison with Ensemble Methods: In this experiment, we introduce four other ensemble based imbalance learning approaches for comparison:

- RUSBoost [44], which applies RandUnder within each iteration of Adaptive Boosting (AdaBoost) pipeline.
- SMOTEBoost [33], which applies SMOTE to generate $|\mathcal{P}|$ new synthetic minority samples within each iteration of AdaBoost pipeline.
- UnderBagging [45] which applies RandUnder to get each bag for Bagging [35]. Note that the only difference between UnderBagging and Easy is that Easy use AdaBoost as its base classifier.
- SMOTEBagging [17], which applies SMOTE to get each bag for Bagging [35], where each bag's sample quantity varies.

Our proposed method was then compared with 4 above methods and the Cascade that we used before. They were considered as the most effective and widely-used imbalance learning methods in very recent reviews [5], [46]. Considering most of the previous approaches were proposed in combination with C4.5 [17], [44], [45], for a fair comparison, we also use the C4.5 classifier as the base model in this experiment. Easy was not included here since it is equivalent to UnderBagging when cooperating with C4.5 classifier.

TABLE VI
GENERALIZED PERFORMANCE OF 6 ENSEMBLE METHODS WITH DIFFERENT AMOUNT OF BASE CLASSIFIERS.

# Base Classifiers	Metric	RUSBoost$_n$	SMOTEBoost$_n$	UnderBagging$_n$	SMOTEBagging$_n$	Cascade$_n$	SPE$_n$
$n = 10$	AUCPRC	0.424±0.061	0.762±0.011	0.355±0.049	0.782±0.007	0.610±0.051	**0.783**±0.015
	F1	0.622±0.055	**0.842**±0.012	0.555±0.053	0.818±0.002	0.757±0.031	0.832±0.018
	GM	0.637±0.045	**0.847**±0.014	0.577±0.044	0.819±0.002	0.760±0.031	0.835±0.018
	MCC	0.189±0.016	0.822±0.018	0.576±0.044	0.819±0.002	0.759±0.031	**0.835**±0.018
	# Sample	6,320	1,723,295	6,320	1,876,204	6,320	6,320
$n = 20$	AUCPRC	0.550±0.032	0.783±0.005	0.519±0.125	0.804±0.013	0.673±0.008	**0.811**±0.005
	F1	0.722±0.021	0.840±0.009	0.678±0.088	0.833±0.005	0.779±0.012	**0.856**±0.008
	GM	0.725±0.019	0.844±0.009	0.685±0.078	0.837±0.005	0.785±0.010	**0.858**±0.010
	MCC	0.202±0.006	0.833±0.005	0.685±0.078	0.837±0.005	0.784±0.010	**0.857**±0.010
	# Sample	12,640	3,478,690	12,640	3,752,408	12,640	12,640
$n = 50$	AUCPRC	0.714±0.025	0.786±0.009	0.676±0.022	0.818±0.004	0.696±0.028	**0.822**±0.006
	F1	0.784±0.010	0.825±0.010	0.773±0.006	0.839±0.009	0.776±0.009	**0.865**±0.012
	GM	0.784±0.010	0.830±0.010	0.774±0.006	0.843±0.008	0.785±0.011	**0.868**±0.012
	MCC	0.297±0.015	0.794±0.007	0.774±0.006	0.842±0.008	0.784±0.011	**0.868**±0.012
	# Sample	31,600	8,937,475	31,600	9,381,020	31,600	31,600

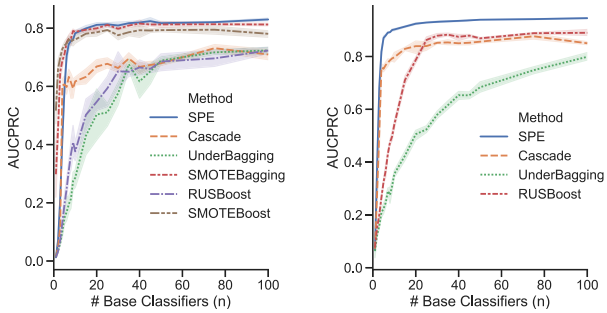
TABLE VII
PERFORMANCE (AUCPRC) OF 6 ENSEMBLE METHODS WITH MISSING VALUES.

Missing Ratio	RUSBoost$_{10}$	SMOTEBoost$_{10}$	UnderBagging$_{10}$	SMOTEBagging$_{10}$	Cascade$_{10}$	SPE$_{10}$
0%	0.424±0.061	0.762±0.011	0.355±0.049	0.782±0.007	0.610±0.051	**0.783**±0.015
25%	0.277±0.043	0.652±0.042	0.258±0.053	0.684±0.019	0.513±0.043	**0.699**±0.026
50%	0.206±0.025	0.529±0.015	0.161±0.013	0.503±0.020	0.442±0.035	**0.577**±0.016
75%	0.084±0.015	0.267±0.019	0.046±0.006	0.185±0.028	0.234±0.023	**0.374**±0.028

Because the number of base models significantly influences the performance of ensemble methods, we test each method with 10, 20 and 50 base models in its ensemble. We must note that such comparison is not totally fair since over-sampling methods need far more data and resources to train each base model. In consideration of computational cost (SMOTEBoost and SMOTEBagging generate a huge amount of synthetic samples on large-scale highly-imbalanced dataset, see Table VI), all ensemble methods were applied on the Credit Fraud dataset with AUCPRC, F1-score, G-mean, MCC for evaluation. For each method, we also list the total number of data samples (# Samples.) that used for training all base models in the ensemble. Table VI shows the experimental results of 5 ensemble methods and our proposed method:

- Comparing with other 3 under-sampling based ensemble methods, SPE uses the same amount of training data but significantly outperforms them over 4 evaluation criteria.
- Comparing with 2 over-sampling based ensemble methods, SPE demonstrates competitive performance using far less (around 1/300) training data.
- Over-sampling based methods are woefully sample-inefficient. They generate a substantial number of synthetic samples under high imbalance ratio. As a result, they further enlarge the scale of training set thus need far more computing resources to train each base model. Higher imbalance ratio and larger dataset can make this situation even worse.

We conduct more detailed experiments on Credit Fraud and Payment Simulation datasets, as shown in Fig. 7. We can see that although SPE uses little data for training, it can still obtain a desirable result which is even better than over-sampling based methods. Moreover, on both tasks SPE shows consistent performance in multiple independent runs. Compared to SPE, other methods are less stable and have greater randomness.



(a) Credit Fraud (b) Payment Simulation

Fig. 7. Generalized performance of ensemble methods on two real-world tasks with the number of base classifiers (n) ranging from 1 to 100. Each curve shows the results of 10 independent runs. Notice that the results of SMOTEBoost and SMOTEBagging are missing on Payment Simulation task due to lack of appropriate distance metric and large computational cost.

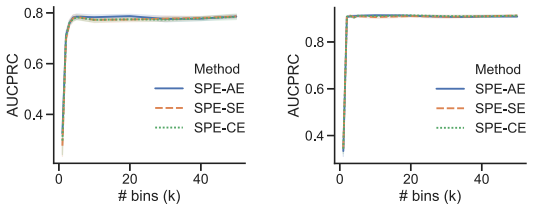
3) Robustness under Missing Values: Finally, we test the robustness of different ensemble methods when there are missing values in the dataset. It is also a common problem that widely existing in real-world applications. To simulate the situation of missing values, we randomly select values from all features in both training and test datasets, then replace them with meaningless 0. We tested all methods on the Credit Fraud dataset, where 0% / 25% / 50% / 75% values are missing.

Results were reported in Table VII. We can observe that SPE demonstrates robust performance under different level of missing, while other methods performing poorly when the missing ratio is high. We also notice that tested methods show different sensitivity to missing values. For an example, SMOTEBagging obtains results better than SMOTEBoost on the original dataset, but this situation is reversed when the missing ratio is greater than 50%.

4) Sensitivity to Hyper-parameters: SPE has 3 key hyper-parameters: number of base classifiers n, number of bins k and hardness function $\mathcal{H}$. In previous discussion we demonstrate the influence of the number of base classifiers (n). Now we conduct experiment to verify the impact of the number of bins (k) and different choices of hardness function ($\mathcal{H}$). Specifically, we test SPE_{10} on two real-world tasks with k ranging from 1 to 50, in cooperation with 3 different hardness functions. They are Absolute Error (AE), Squared Error (SE) and Cross Entropy (CE), where:

1) $\mathcal{H}_{AE}(x, y, F) = |F(x) - y|$
2) $\mathcal{H}_{SE}(x, y, F) = (F(x) - y)^2$
3) $\mathcal{H}_{CE}(x, y, F) = -ylog(F(x)) - (1 - y)log(1 - F(x))$

The results in Fig. 8 show that our method is robust to different selection of k and $\mathcal{H}$. Note that k determines how detailed our hardness distribution approximation is, thus setting a small k, e.g., $k < 10$, may lead to poor performance.



(a) Credit Fraud (b) Payment Simulation

Fig. 8. Performance (mean of 10 independent runs) of SPE_{10} on two real-world tasks using different number of bins (k) and hardness function ($\mathcal{H}$).

VII. Related work

Imbalanced data classification has been a fundamental problem in machine learning [9], [10]. Many research works have been proposed to solve such problem. This research field is also known as *Imbalance Learning*. Recently, Guo *et al.* provided a systematic review of existing methods and real-world applications in the field of imbalance learning [5].

Most of proposed works employed distance-based methods to obtain re-sampled data for training canonical classifiers [12]–[14], [27]. Based on them, many works combine re-sampling with ensemble learning [17], [33], [44], [45]. Such strategies have proven to be very effective [46]. Distance-based methods have several deficiencies. First, it is hard to define distance on a real-world dataset, especially when it contains categorical features or missing values. Second, the cost of computing distances between each samples can be huge when applying on large-scale datasets. Even though the distance-based methods have been successfully used for re-sampling, they do not guarantee desirable performance for different classifiers due to their model-agnostic designs.

Some other methods try to assigning different weights to samples rather than re-sampling the whole dataset [15], [16]. They require assistance from domain experts and may fail when cooperating with batch training methods (e.g. neural network). We prefer not to include such methods in this paper because previous experiments [16] have shown that setting arbitrary costs without domain knowledge do not allow them to achieve their maximum potential.

There are some works in other domains (e.g. Active Learning [47], Self-paced Learning [48]) that adopt the idea of selecting "informative" samples but focus on completely different problems. Specifically, an active learner interactively queries the user to obtain the labels of new data points, while a self-paced learning algorithm tries to present the training data in a meaningful order that facilitates learning. However, they perform the sampling without considering the overall data distribution, thus their fine-tuning process can be easily disturbed when the training set is imbalanced. In comparison, SPE applies under-sampling + ensemble strategy to balance the dataset, making it applicable to any canonical classifier. By considering the dynamic hardness distribution over the whole dataset, SPE performs adaptive and robust under-sampling rather than blindly selecting "informative" data samples.

To summarize, traditional distance-based re-sampling methods ignore the difference of model capacity, thus may lead to poor performance when cooperating with specific classifiers. They also require additional computation to calculate distances between samples, making them computationally inefficient, especially on large datasets. Moreover, it is often difficult to determine a clear distance metric in practice, as real-world datasets may contain categorical features and missing values. Most ensemble-based methods integrate such distance-based re-sampling into their pipelines, thus are still negatively affected by the above factors. Comparing with existing works, SPE doesn't require any pre-defined distance metric or computation, making it easier to apply and more computationally efficient. By self-paced harmonizing the hardness distribution w.r.t the given classifier, SPE is adaptive to different models and robust to noises and missing values.

VIII. Conclusions

In this paper we have described the problem of *highly imbalanced*, *large-scale* and *noisy* data classification that widely exists in real-world applications. Under such a scenario, we have demonstrate that canonical machine learning / imbalance learning approaches suffer from unsatisfactory results and low computational efficiency.

Self-paced Ensemble, a novel learning framework for massive imbalance classification has been proposed in this paper. We argue that all of the difficulties - high imbalance ratio, overlapping between classes, presence of noises - are critical for massive imbalance classification. Hence, we have introduced the concept of classification hardness distribution to integrate the information of these difficulties into our learning framework. We conducted extensive experiments on a variety of challenging real-world tasks. Comparing with other methods, our framework has better performance, wider applicability, and higher computational efficiency. Overall, we believe that we have provided a new paradigm of integrating task difficulties into the imbalance classification system. Various real-world applications can benefit from our framework.

REFERENCES

[1] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich, "Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine." Omnipress, 2010.

[2] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit card fraud detection: a realistic modeling and a novel learning strategy," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 8, pp. 3784–3797, 2018.

[3] D. Gamberger, N. Lavrac, and C. Groselj, "Experiments with noise filtering in a medical domain," in *ICML*, 1999, pp. 143–151.

[4] M. Sariyar, A. Borg, and K. Pommerening, "Controlling false match rates in record linkage using extreme value theory," *Journal of Biomedical Informatics*, vol. 44, no. 4, pp. 648–654, 2011.

[5] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220–239, 2017.

[6] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[7] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[8] S. S. Haykin, S. S. Haykin, S. S. Haykin, K. Elektroingenieur, and S. S. Haykin, *Neural networks and learning machines*. Pearson education Upper Saddle River, 2009, vol. 3.

[9] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge & Data Engineering*, no. 9, pp. 1263–1284, 2008.

[10] H. He and Y. Ma, *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons, 2013.

[11] I. Mani and I. Zhang, "knn approach to unbalanced data distributions: a case study involving information extraction," in *Proceedings of workshop on learning from imbalanced datasets*, vol. 126, 2003.

[12] I. Tomek, "Two modifications of cnn," *IEEE Trans. Systems, Man and Cybernetics*, vol. 6, pp. 769–772, 1976.

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[14] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE, 2008, pp. 1322–1328.

[15] C. Elkan, "The foundations of cost-sensitive learning," in *International joint conference on artificial intelligence*, vol. 17, no. 1. Lawrence Erlbaum Associates Ltd, 2001, pp. 973–978.

[16] X.-Y. Liu and Z.-H. Zhou, "The influence of class imbalance on cost-sensitive learning: An empirical study," in *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 2006, pp. 970–974.

[17] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," in *2009 IEEE Symposium on Computational Intelligence and Data Mining*. IEEE, 2009, pp. 324–331.

[18] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 539–550, 2009.

[19] K. Napierała, J. Stefanowski, and S. Wilk, "Learning from imbalanced data in presence of noisy and borderline examples," in *International Conference on Rough Sets and Current Trends in Computing*. Springer, 2010, pp. 158–167.

[20] V. García, J. Sánchez, and R. Mollineda, "An empirical study of the behavior of classifiers on imbalanced and overlapped data sets," in *Iberoamerican Congress on Pattern Recognition*. Springer, 2007, pp. 397–406.

[21] R. C. Prati, G. E. Batista, and M. C. Monard, "Learning with class skews and small disjuncts," in *Brazilian Symposium on Artificial Intelligence*. Springer, 2004, pp. 296–306.

[22] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

[23] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation," in *Australasian joint conference on artificial intelligence*. Springer, 2006, pp. 1015–1021.

[24] S. Boughorbel, F. Jarray, and M. El-Anbari, "Optimal classifier for imbalanced data using matthews correlation coefficient metric," *PloS one*, vol. 12, no. 6, p. e0177678, 2017.

[25] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 233–240.

[26] M. Kubat, S. Matwin *et al.*, "Addressing the curse of imbalanced training sets: one-sided selection," in *Icml*, vol. 97. Nashville, USA, 1997, pp. 179–186.

[27] J. Laurikkala, "Improving identification of difficult small classes by balancing class distribution," in *Conference on Artificial Intelligence in Medicine in Europe*. Springer, 2001, pp. 63–66.

[28] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *International conference on intelligent computing*. Springer, 2005, pp. 878–887.

[29] G. E. Batista, A. L. Bazzan, and M. C. Monard, "Balancing training data for automated annotation of keywords: a case study." in *WOB*, 2003, pp. 10–18.

[30] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[31] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[32] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[33] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *European conference on principles of data mining and knowledge discovery*. Springer, 2003, pp. 107–119.

[34] B. Li, Y. Liu, and X. Wang, "Gradient harmonized single-stage detector," *arXiv preprint arXiv:1811.05181*, 2018.

[35] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[36] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[37] J. H. Friedman, "Stochastic gradient boosting," *Computational statistics & data analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[38] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365.html

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[40] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, 2017, pp. 3146–3154.

[41] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[42] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 3, pp. 408–421, 1972.

[43] I. Tomek, "An experiment with the edited nearest-neighbor rule," *IEEE Transactions on systems, Man, and Cybernetics*, no. 6, pp. 448–452, 1976.

[44] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "Rusboost: A hybrid approach to alleviating class imbalance," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 40, no. 1, pp. 185–197, 2010.

[45] R. Barandela, R. M. Valdovinos, and J. S. Sánchez, "New applications of ensembles of classifiers," *Pattern Analysis & Applications*, vol. 6, no. 3, pp. 245–256, 2003.

[46] F. Alberto, G. Salvador, G. Mikel, P. Ronaldo C., and K. Bartosz, *Learning from Imbalanced Data Sets*. Springer, 2018.

[47] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[48] M. P. Kumar, B. Packer, and D. Koller, "Self-paced learning for latent variable models," in *Advances in Neural Information Processing Systems*, 2010, pp. 1189–1197.