Self-paced Ensemble for Highly Imbalanced Massive Data Classification

Zhining Liu*†, Wei Cao‡, Zhifeng Gao‡, Jiang Bian‡, Hechang Chen*†, Yi Chang*†, and Tie-Yan Liu‡

* School of Artificial Intelligence, Jilin University
† Key Lab. of Symbolic Computation and Knowledge Engineering of MOE, Jilin University
‡ Microsoft Research

Presenter: Zhining Liu

April 22, 2020

36th IEEE International Conference on Data Engineering (ICDE 2020)

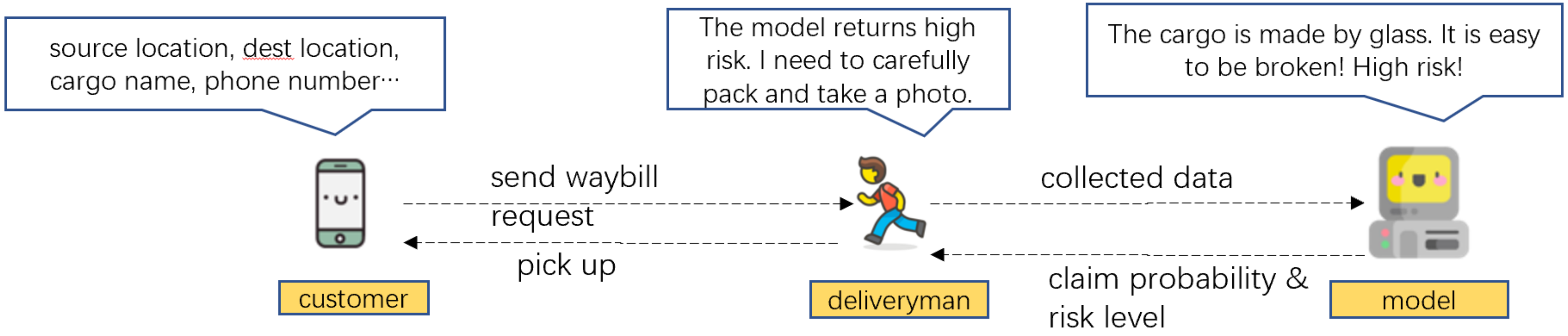
Table of contents

Challenges

Emerging challenges from more ***large-scale, extremely imbalanced*** and ***low-quality*** datasets that come with the development of information systems (e.g., CTR, fraud detection, medical diagnosis).

Challenges

- An example: predict the claim probability of waybill orders
  - more than 3,000,000,000 samples in TB level
  - Imbalance Ratio (IR) = #negative : #positive = 3,000 : 1
  - sparse categorical features (e.g., user IDs)
  - lots of missing values (up to 90% in some columns)
  - noisy data

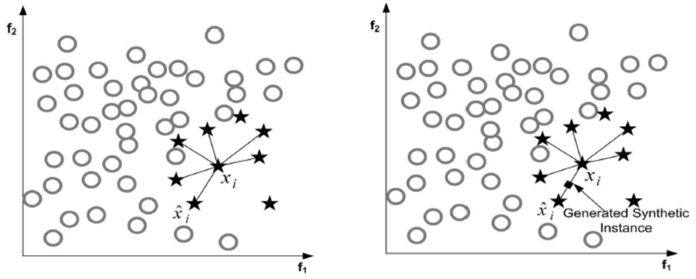


Prior work

- Resampling methods
  - oversample minority cases
  - undersample majority cases
- Reweighting methods
  - cost-sensitive learning
  - hard example mining
- Ensemble methods
  - Integrate resampling/reweighting methods into ensemble learning frameworks

Examples

SMOTE

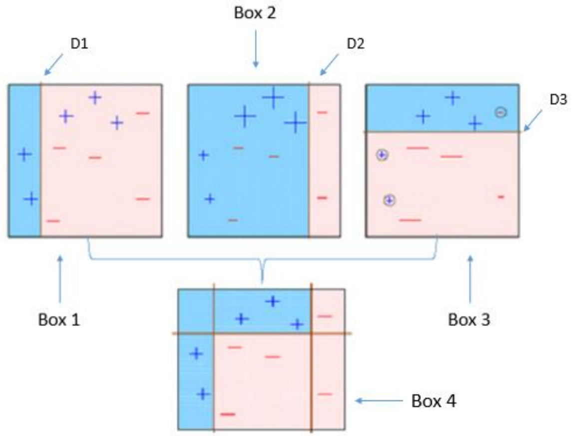


(a) (b)

Cost-sensitive learning

	actual negative	actual positive
predict negative	c_{00}	c_{01}
predict positive	c_{10}	c_{11}

AdaBoost



Drawbacks of existing solutions

Family	Branch	Representatives	Drawbacks
Resampling	random resampling	---	**Unsatisfactory performance**
	under-sampling	TomekLink, ENN, NearMiss	**1. High cost for clustering/finding nearest neighbors on a large-scale dataset. 2. fail to work when the dataset is extremely imbalanced or contains many missing values. 3. Over-sampling methods further enlarge the size of the training dataset.**
	over-sampling	SMOTE, ADASYN, Borderline-SMOTE	
	hybrid-sampling	SMOTE-Tomek, SMTOE-ENN	
Reweighting	cost-sensitive learning	Cost-sensitive C4.5	**1. Domain knowledge is required to set an appropriate cost matrix. 2. Sensitive to noises and outliers.**
	hard example mining	AdaBoost, FocalLoss	
Ensemble	resampling/reweighting + Ensemble Learning	RUSBoost, SMOTEBoost, SMOTEBagging	**The aforementioned problems of resampling/reweighting methods still hold.**

Motivation

We need a practical learning framework that can effectively handle ***large-scale data*** with ***extreme class imbalance*** while being robust to ***missing values, noises,*** and ***outliers***.

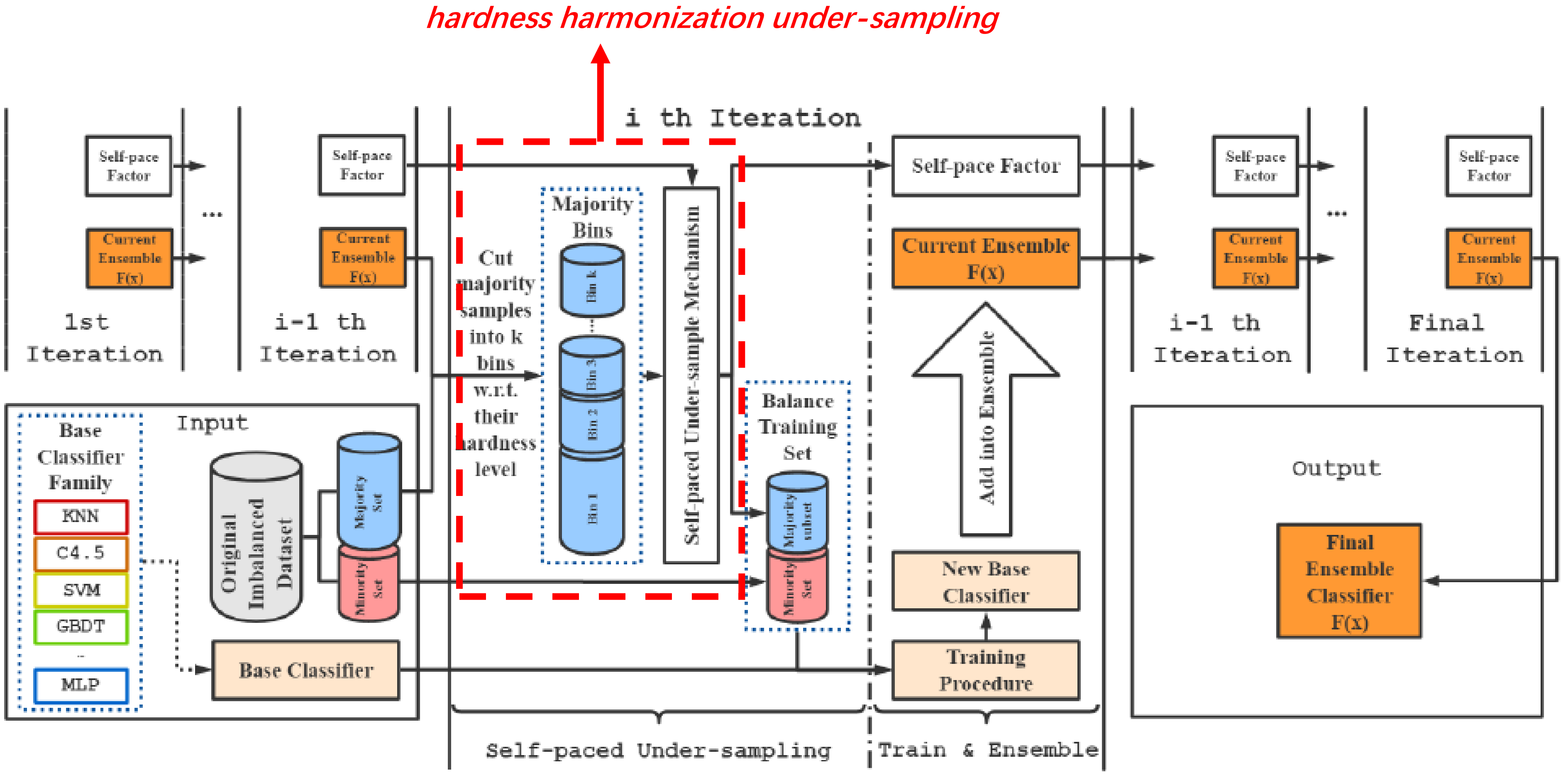
Table of contents

Self-paced Ensemble (SPE)

Challenge
Large-scale data
Extreme class imbalance
Missing values
Sparse categorical features
Noises and outliers

- Efficient ensemble training
  - Only need $O(k \cdot \#pos)$ samples to build k-classifier ensemble
- Introduce the “classification hardness”
  - Adaptive sampling without distance-computing
- Self-paced Learning with hardness harmonization
  - robust to the outliers while ensuring fast convergence

Overview

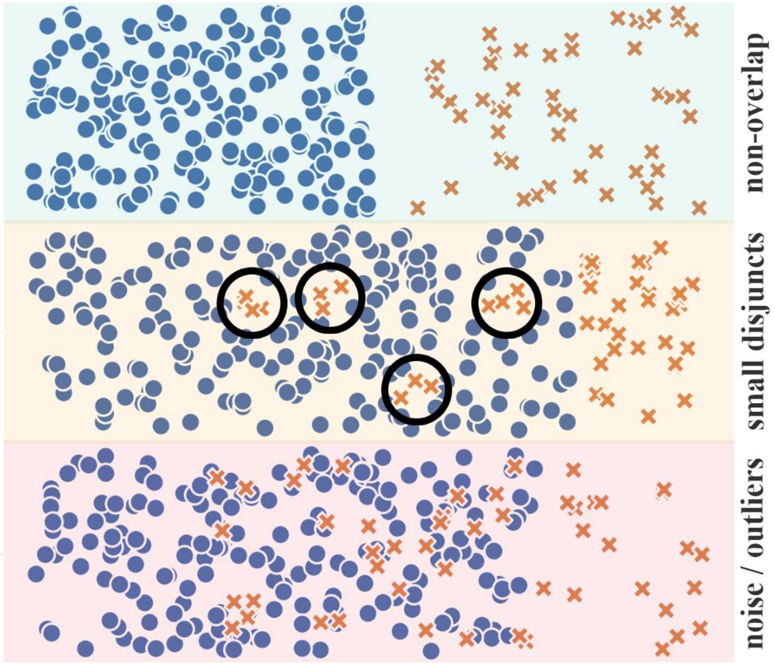


hardness harmonization under-sampling
i th Iteration
Self-pace Factor
Current Ensemble F(x)
1st Iteration
i-1 th Iteration
Cut majority samples into k bins w.r.t. their hardness level
Majority Bins
Bin k
Bin 3
Bin 2
Bin 1
Self-paced Under-sample Mechanism
Balance Training Set
Majority subset
Minority Set
Add into Ensemble
New Base Classifier
Training Procedure
Final Iteration
Output
Final Ensemble Classifier F(x)
Input
Base Classifier Family
KNN
C4.5
SVM
GBDT
MLP
Original Imbalanced Dataset
Majority Set
Minority Set
Base Classifier
Self-paced Under-sampling
Train & Ensemble

Table of contents

Classification hardness

- Class imbalance is **NOT** the sole source of learning difficulties.
  - Other factors:
    - Small disjuncts problem
    - Presence of noises and outliers
    - Overlapped underlying class distribution



- We introduce "classification hardness distribution" to integrate all these factors into our learning framework.

Classification Hardness

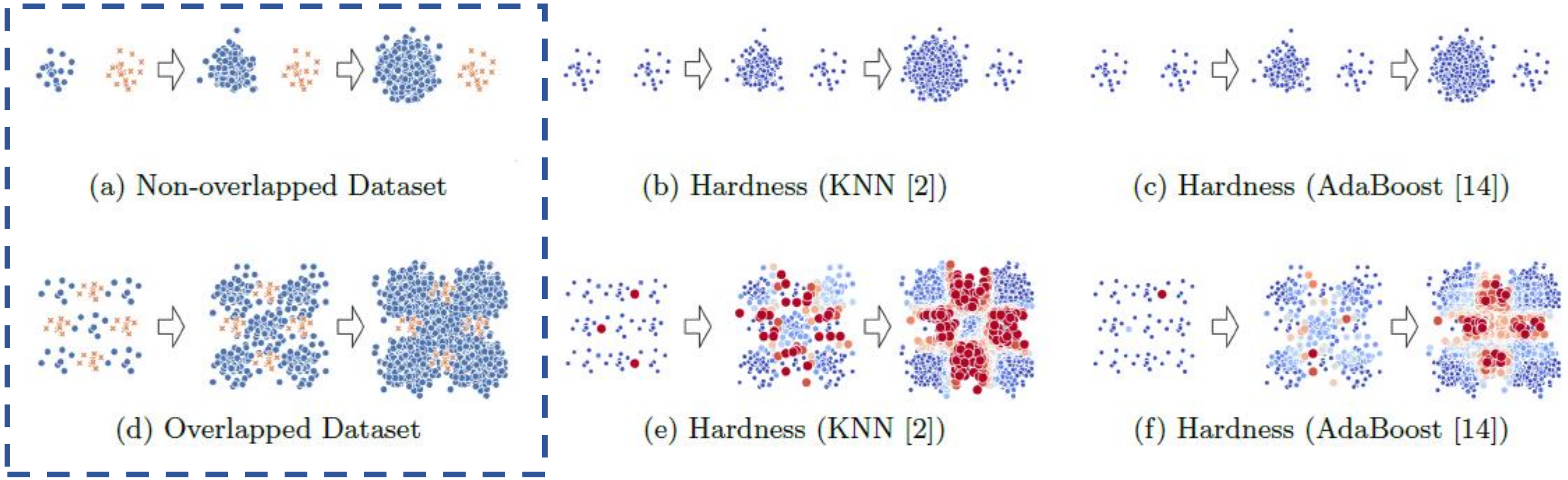
- What is “classification hardness”?
  - *How hard is a sample to be correctly classified by the current model.*
  - *Can be given any “decomposable” error function*
  - *(e.g., absolute error, squared error, cross entropy)*

Example (absolute error):

$$hardness_{x,y,F} = |F(x) - y|$$

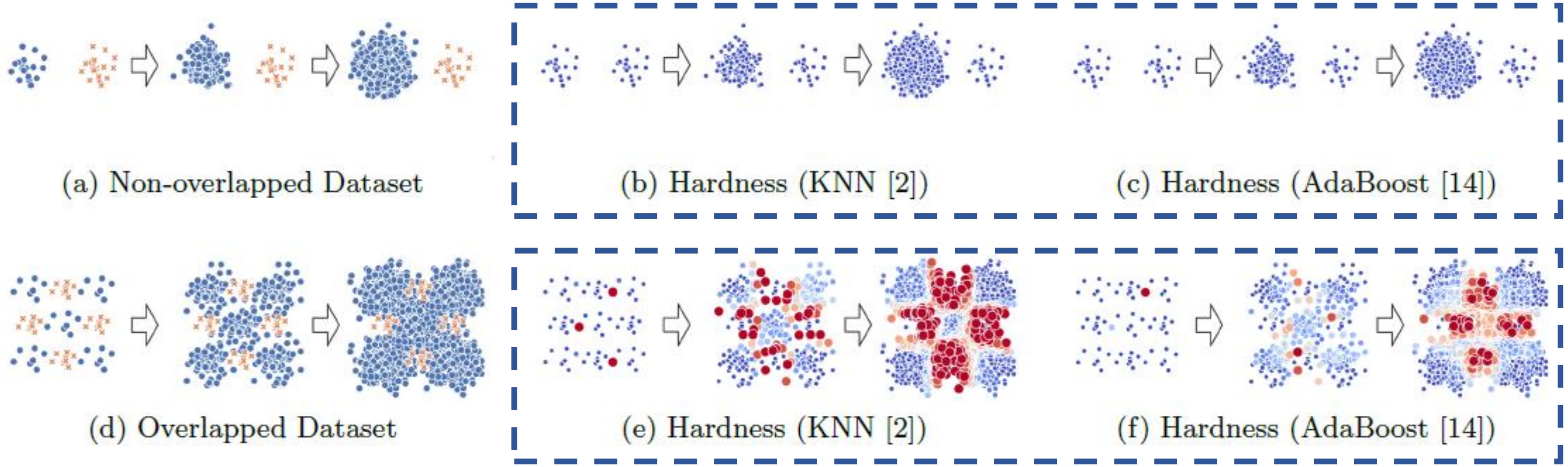
Hardness Distribution

- Fills the gap between **imbalance ratio** and **task difficulty**
  - Fig.(a) & Fig.(d) have the same imbalance ratio (IR) (1:1 to 1:100)
  - As the IR grows, Fig.(d) becomes a much harder classification task



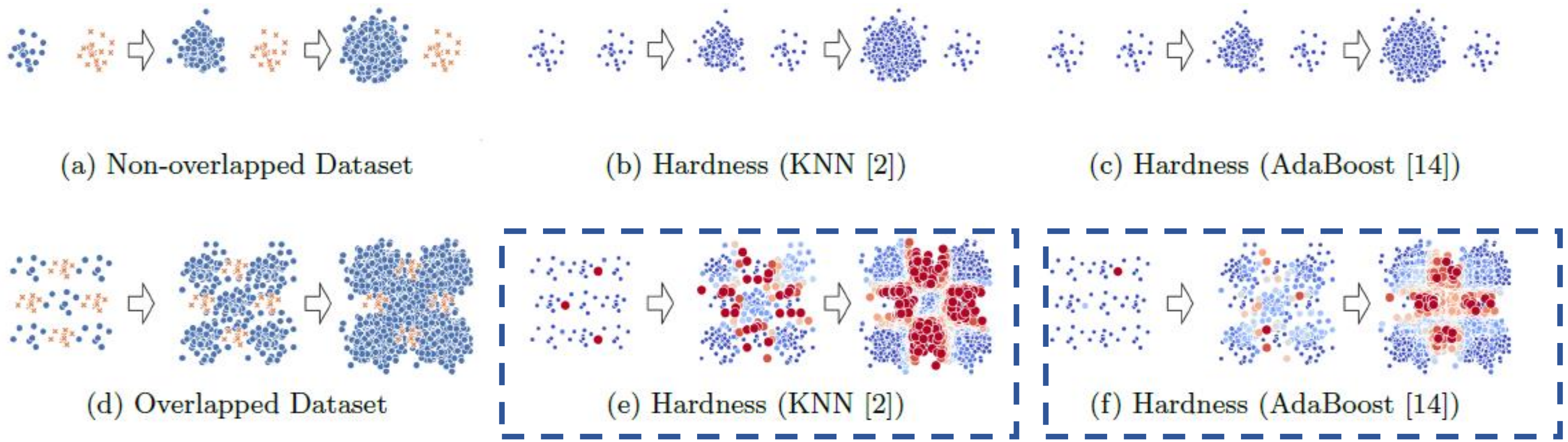
Hardness Distribution

- Fills the gap between **imbalance ratio** and **task difficulty**
  - Fig.(a) & Fig.(d) have the same imbalance ratio (IR) (1:1 to 1:100)
  - Fig.(a) & Fig.(d) have the very different "classification hardness"



Hardness Distribution

- Fills the gap between **model capacity** and **sampling strategy**
  - Different learning models have very different capacities
  - Hardness distribution can capture such difference that is ignored by other preprocessing methods such as SMOTE.

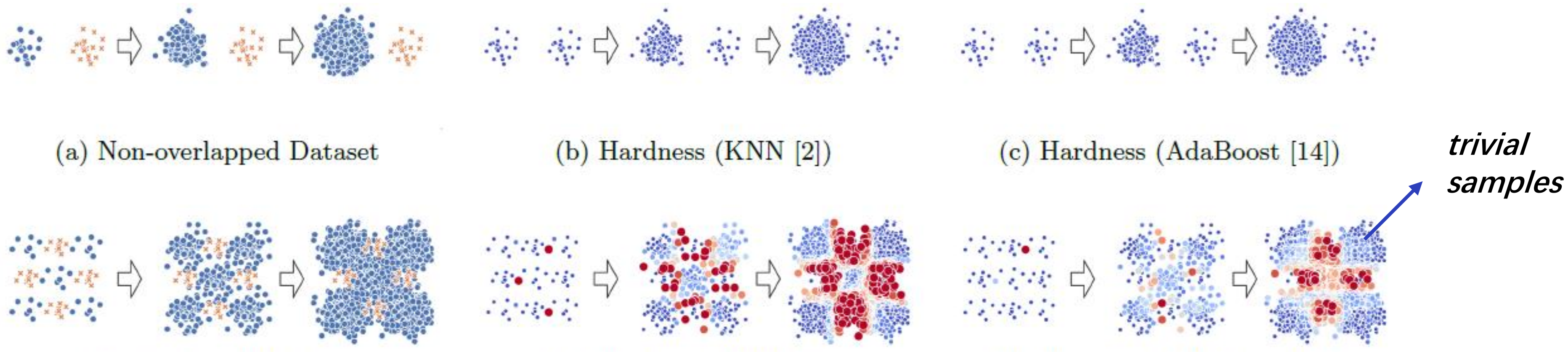


(a) Non-overlapped Dataset (b) Hardness (KNN [2]) (c) Hardness (AdaBoost [14])

(d) Overlapped Dataset (e) Hardness (KNN [2]) (f) Hardness (AdaBoost [14])

General types of data samples

- **trivial samples (in blue)**
  - only contribute tiny hardness
  - large population, total hardness is non-negligible
  - *trivial samples do not provide new information for training:*
    - *reduce population, only keep the "skeleton"*

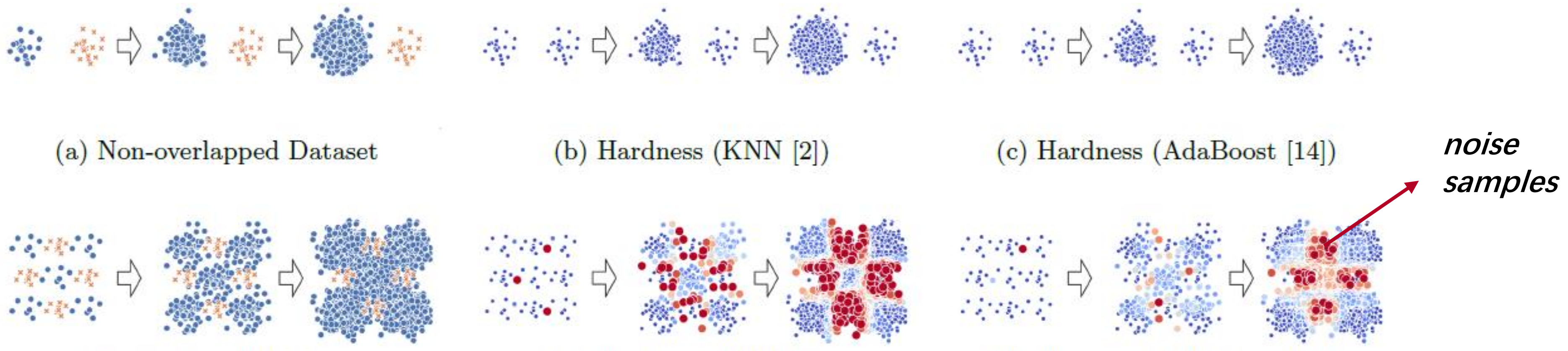


(a) Non-overlapped Dataset (b) Hardness (KNN [2]) (c) Hardness (AdaBoost [14])

(d) Overlapped Dataset (e) Hardness (KNN [2]) (f) Hardness (AdaBoost [14])

General types of data samples

- **noise samples (in dark red)**
  - small population
  - each contributes a large hardness value
  - *noise samples can disturb the training process:*
    - *eliminate the noise, while keep useful information*

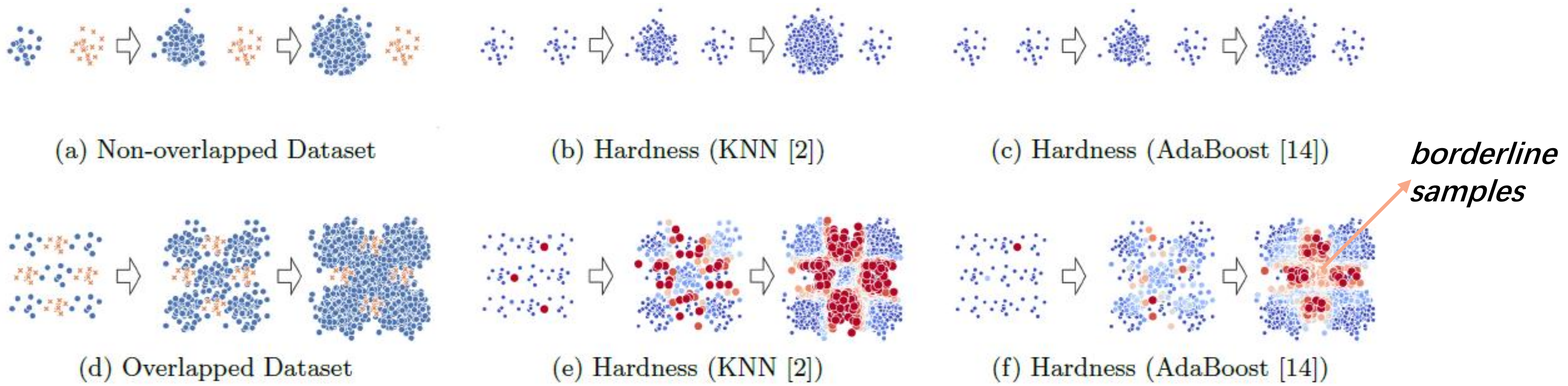


(a) Non-overlapped Dataset (b) Hardness (KNN [2]) (c) Hardness (AdaBoost [14])

(d) Overlapped Dataset (e) Hardness (KNN [2]) (f) Hardness (AdaBoost [14])

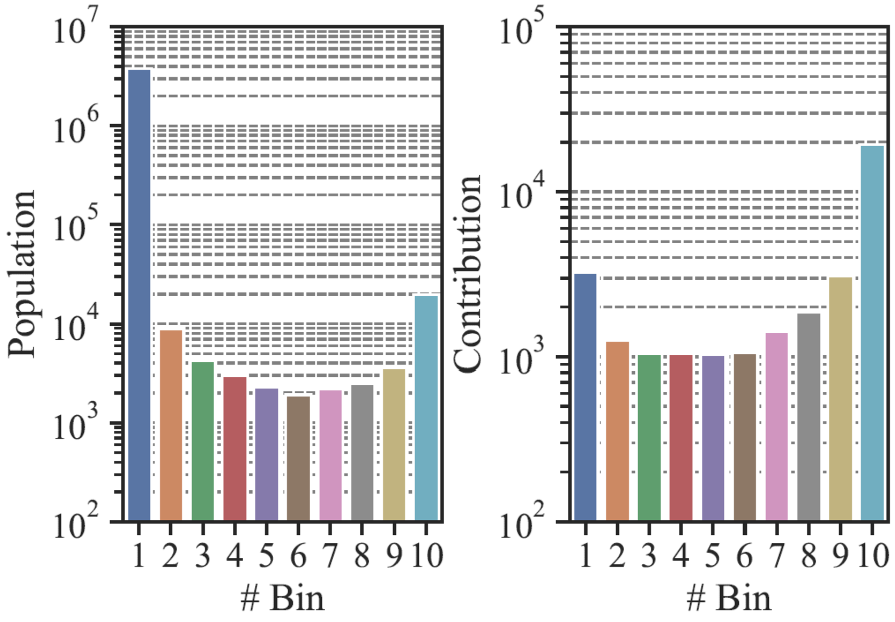
General types of data samples

- **borderline samples (in light red)**
  - close to the decision boundary
  - *borderline samples are the most informative*
    - *enlarge the corresponding weights usually can be helpful*



Hardness Histogram

- Identify samples under finer granularity:
  - use the current model to calculate the hardness
  - cut data into bins and form a histogram



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

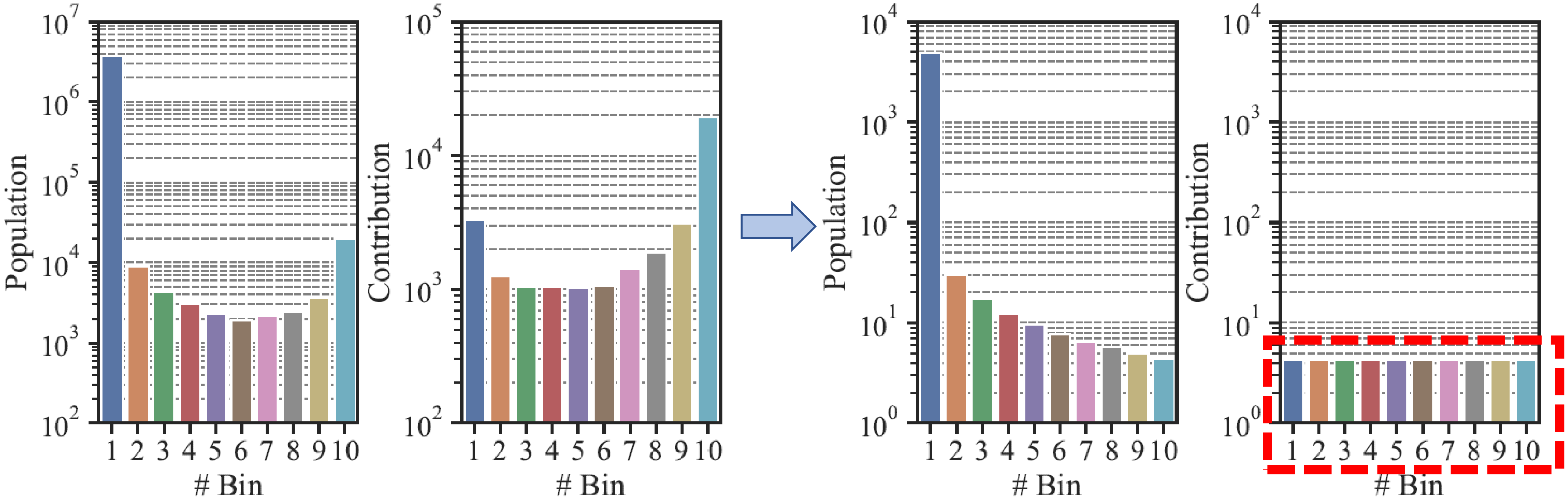
Hardness histogram approximates the distribution of hardness values and implicitly reflects the task difficulty.

Table of contents

Our First Algorithm

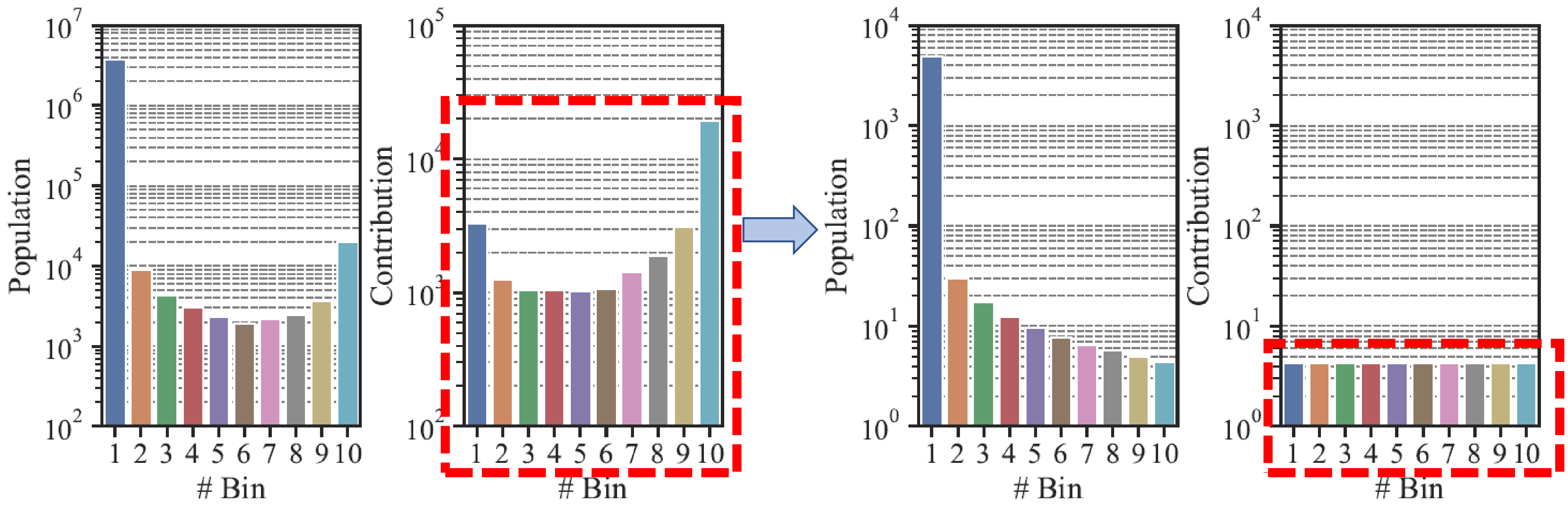
Hardness Harmonization

- Resampling to equalize the *hardness contribution* from each bin in the histogram.
- Resulting in a "harmonized" subset.



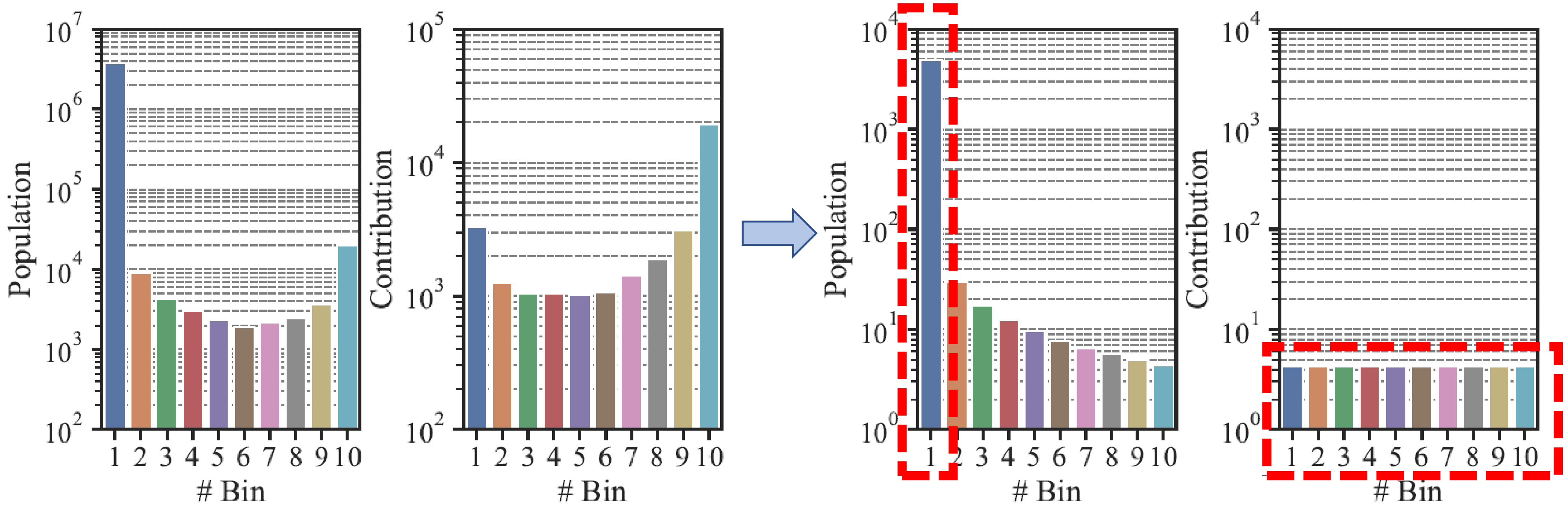
Hardness Harmonization

- Advantage:
  - weights of borderline samples are enhanced
  - effect of trivial/noise samples are reduced



Hardness Harmonization

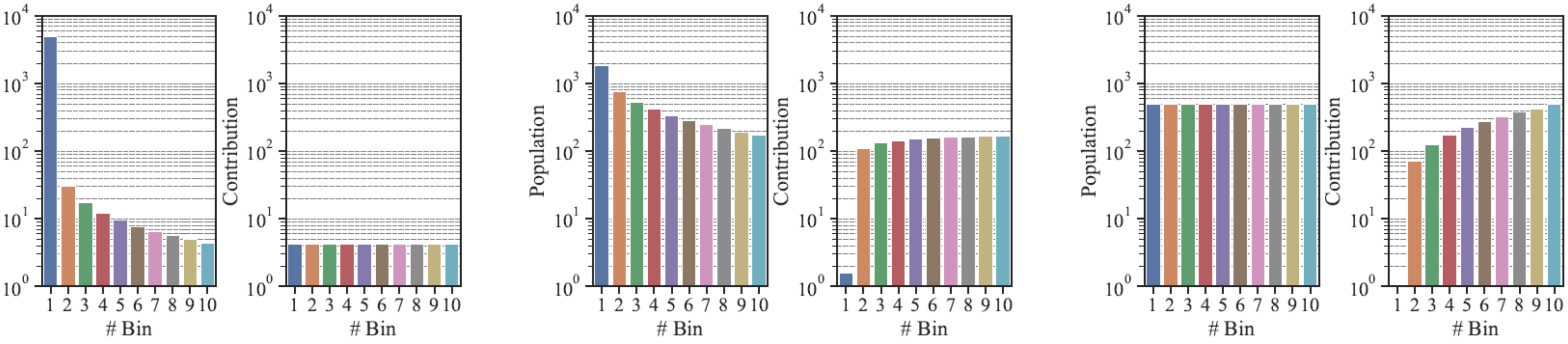
- **Problem?**
  - population of trivial samples grow rapidly during the training
  - simple harmonization leaves lots of trivial samples and slows down the training



Our Second Algorithm

Self-paced Under-sample

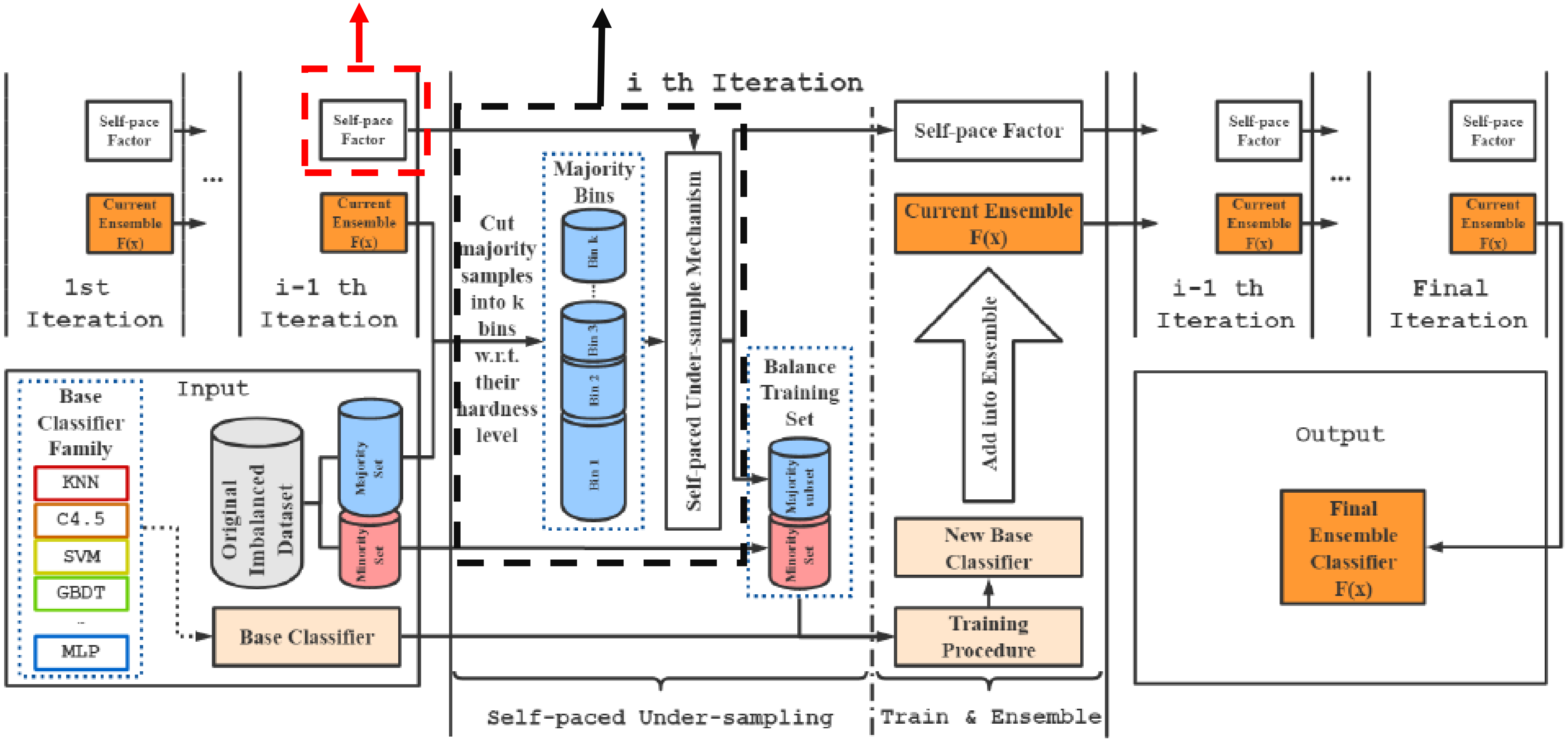
- introduce a self-paced factor α
  - a penalty factor to large population bins
- as α grows, we gradually focus on harder samples
- always keep a reasonable proportion of trivial samples



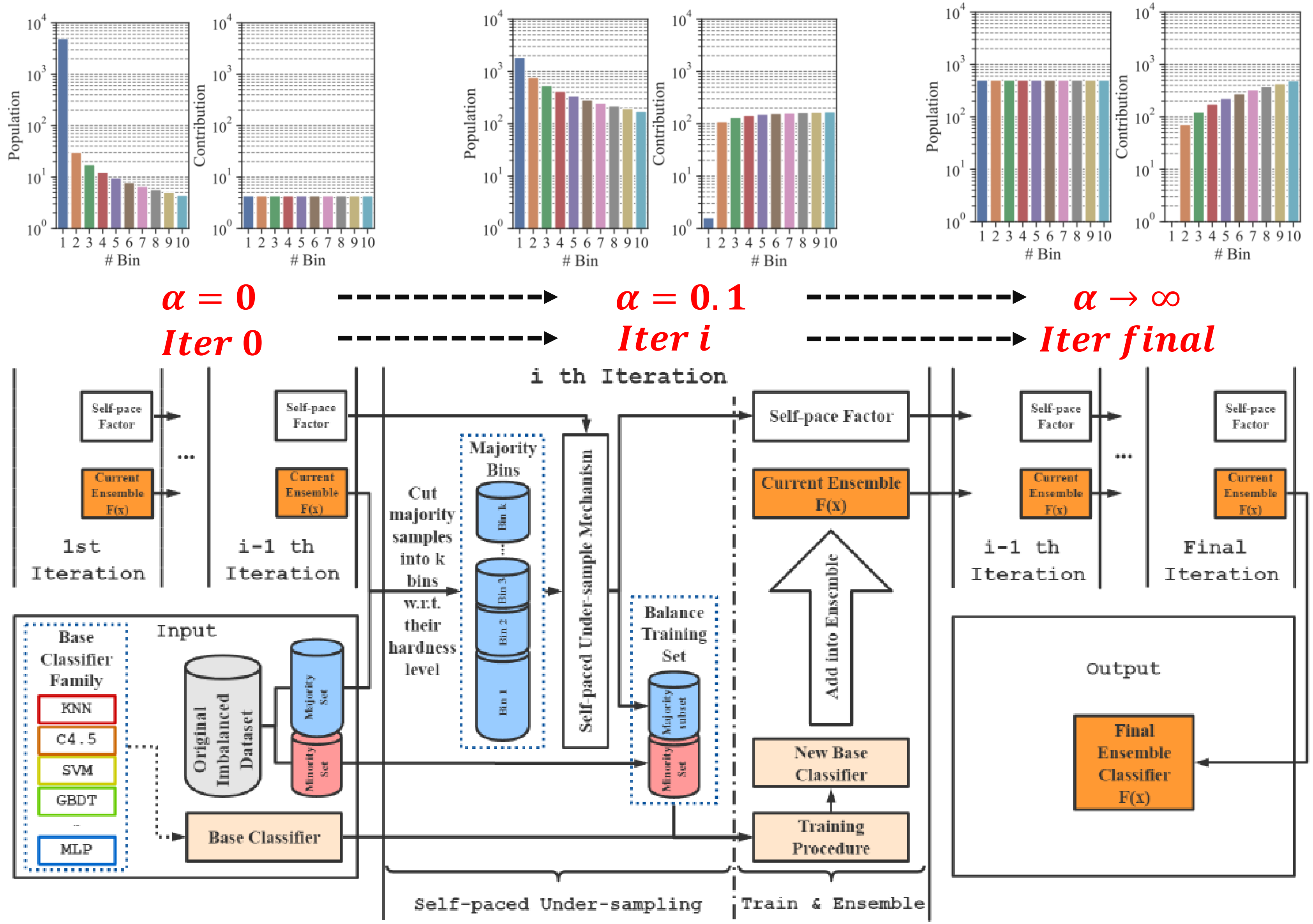
(b) $\alpha = 0$ (c) $\alpha = 0.1$ (d) $\alpha \to \infty$

Self-paced Under-sample

self-paced *hardness harmonization under-sampling*



Self-paced Under-sample



Self-paced Ensemble

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,

Initialize: $\mathcal{P} \Leftarrow$ minority in $\mathcal{D}$, $\mathcal{N} \Leftarrow$ majority in $\mathcal{D}$

Train classifier f_0 using random under-sample majority subsets $\mathcal{N}'_0$ and $\mathcal{P}$, where $|\mathcal{N}'_0| = |\mathcal{P}|$.

for *i=1 to* n **do**

- Ensemble $F_i(x) = \frac{1}{i} \sum_{j=0}^{i-1} f_j(x)$
- Cut majority set into k bins w.r.t. $\mathcal{H}(x, y, F_i)$: $B_1, B_2, \cdots, B_k$

  form the histogram
- Average hardness contribution in ℓ-th bin: $h_\ell = \sum_{s \in B_\ell} \mathcal{H}(x_s, y_s, F_i)/|B_\ell|, \ \forall \ell = 1, \ldots, k$
- Update self-paced factor $\alpha = tan(\frac{i\pi}{2n})$

  control the growth rate of α
- Unnormalized sampling weight of ℓ-th bin: $p_\ell = \frac{1}{h_\ell + \alpha}, \forall \ \ell = 1, \ldots, k$
- Under-sample from ℓ-th bin with $\frac{p_\ell}{\sum_m p_m} \cdot |\mathcal{P}|$ samples

  perform sampling from each hardness bin
- Train f_i using newly under-sampled subset

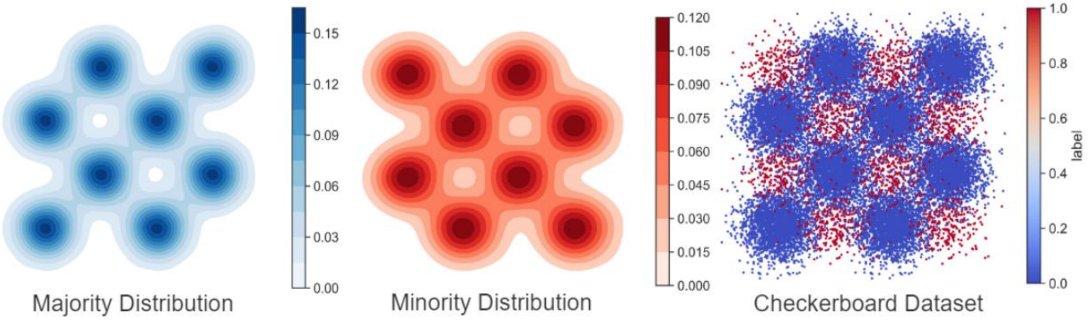
end

return *final ensemble* $F(x) = \frac{1}{n} \sum_{m=1}^{n} f_m(x)$

Table of contents

Experiment

- Synthetic Dataset
  - 4×4 checkerboard with different level of class overlapping



- 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}

Experiment

- Base Classifiers
  - K Nearest Neighbor classifier (KNN)
  - Decision Tree (DT)
  - Logistic Regression (LR)
  - Multi-Layer Perceptron (MLP)
  - Support Vector Machine (SVM)
  - Adaptive boosting (Adaboost)
  - Bootstrap aggregating (Bagging)
  - Random Forest (RandForest)
  - Gradient Boosting Decision Tree (GBDT)
- Evaluation Criteria
  - Area under precision-recall curve (AUCPRC)
  - F1-score (F1)
  - Geometric Mean (GM)
  - Matthews correlation coefficient (MCC)
- Baseline Methods
  - 7 under-sampling methods (RandUnder, NearMiss, Clean, ENN, TomekLink, AllKNN, OSS)
  - 4 over-sampling methods (RandOver, SMOTE, Border-SMOTE, ADASYN)
  - 2 hybrid-sampling methods (SMOTE-ENN, SMOTE-Tomek)
  - 6 ensemble methods (RUSBoost, SMOTEBoost, UnderBagging, SMOTEBagging, EasyEnsemble, BalanceCascade)

Experiment

- Table: performance (AUCPRC) on the synthetic dataset.

Model	Hyper	RandUnder	Clean	SMOTE	Easy_{10}	Cascade_{10}	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

Experiment

- Table: performance on 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

Experiment

- Table: performance of resampling methods (on CreditFraud task).

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

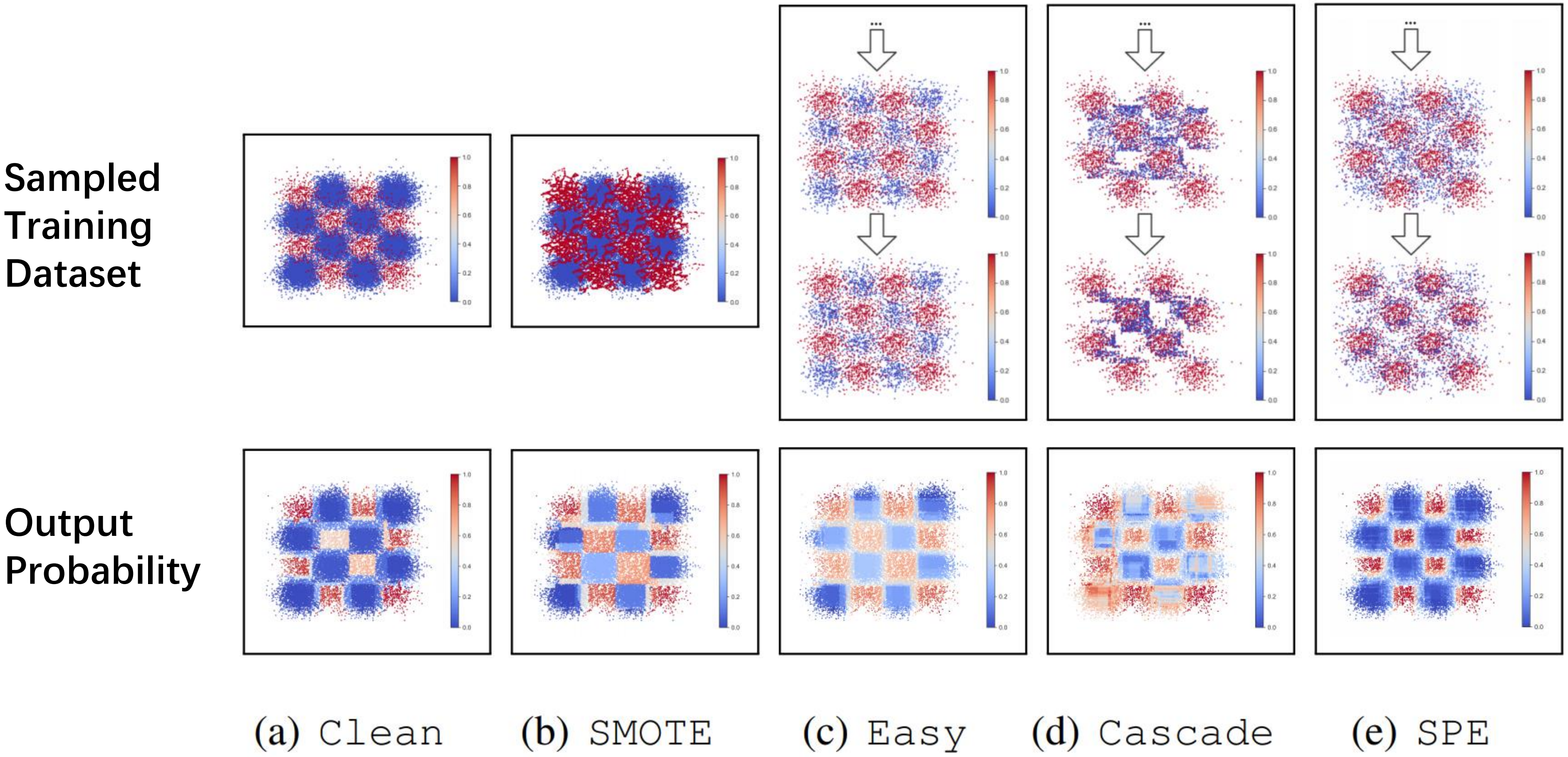
Experiment

- Table: performance of ensemble methods (on CreditFraud task).

# 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

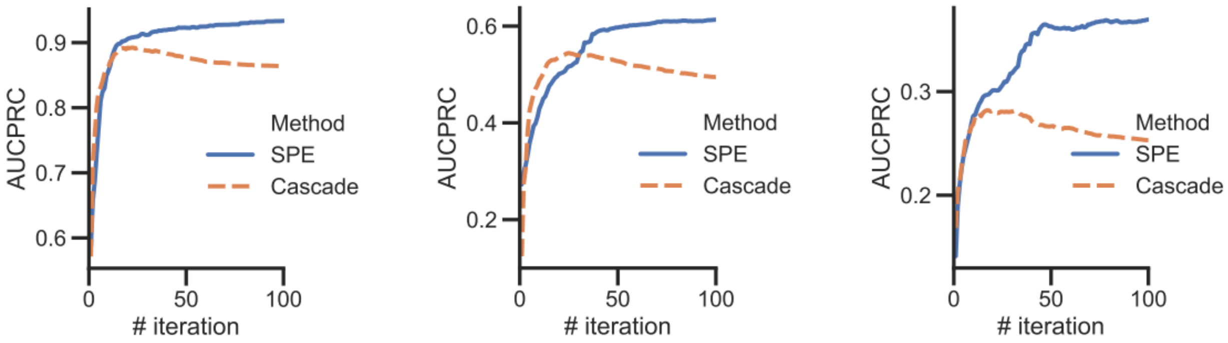
Experiment

- Visualization on the synthetic dataset:



Experiment

- Robust to class overlapping & outliers:



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

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

- Robust to 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

Thanks!

Code URL: github.com/ZhiningLiu1998/self-paced-ensemble