Train In Data

Ensemble Methods: Summary

Few methods are open-sourced

Ensemble methods

The `imblearn.ensemble` module include methods generating under-sampled subsets combined inside an ensemble.

Boosting algorithms

`EasyEnsembleClassifier`([n_estimators, ...])	Bag of balanced boosted learners also known as EasyEnsemble.
`RUSBoostClassifier`([base_estimator, ...])	Random under-sampling integrated in the learning of AdaBoost.

Bagging algorithms

`BalancedBaggingClassifier`([base_estimator, ...])	A Bagging classifier with additional balancing.
`BalancedRandomForestClassifier`([...])	A balanced random forest classifier.

Scalability

- Methods based on KNN do not scale well
- In ensemble we repeat the re-sampling in each bag or at each round of boosting
- Ensemble with random over- or under-sampling are the least costly

Comparison

RUSBoost: A Hybrid Approach to Alleviating Class Imbalance

Chris Seiffert, Taghi M. Khoshgoftaar, *Member, IEEE*, Jason Van Hulse, *Member, IEEE*, and Amri Napolitano

- Compared methods in 15 publicly available datasets
- RUSBoost performs similarly to SMOTEBoost but is faster

Comparison: average performance

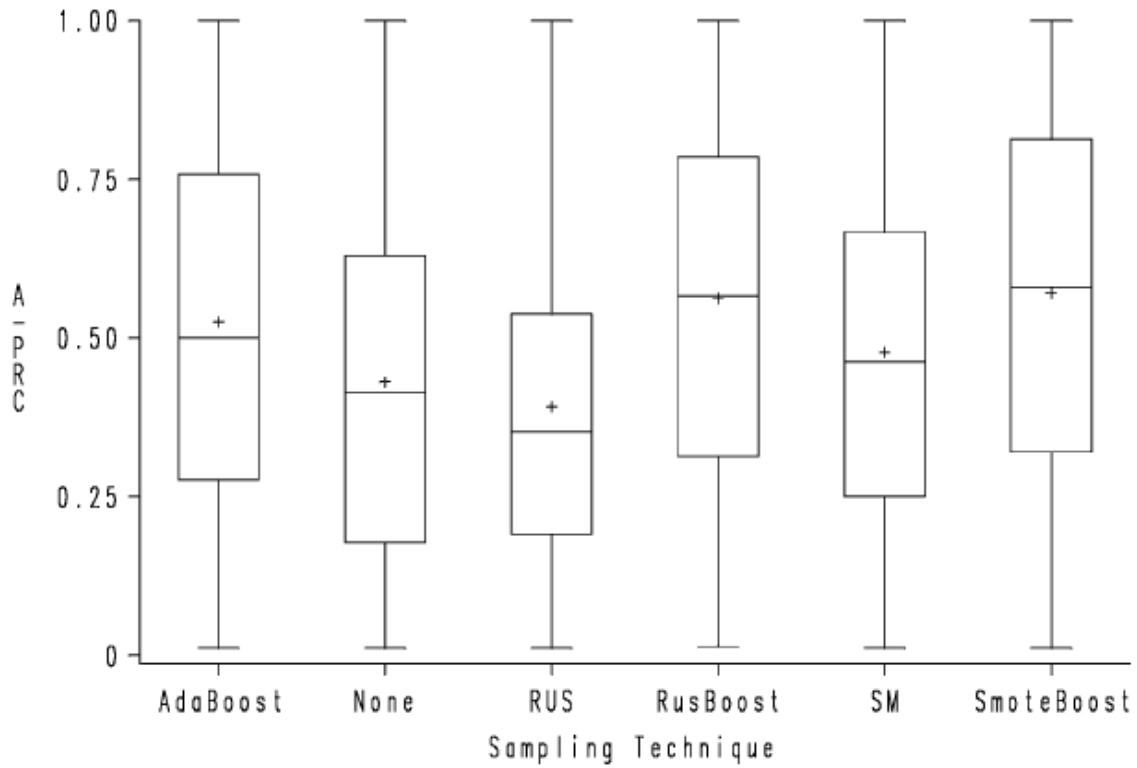

Fig. 3. A-PRC boxplot.

- Average PR-Curve of each method across the 15 datasets.
- AdaBoost, RUSBoost and SMOTEBOOST perform very similarly

Comparison

Exploratory Undersampling for Class-Imbalance Learning

Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou, *Senior Member, IEEE*

- Authors of EasyEnsemble and BalanceCascade

Comparison: speed

TABLE III
RUNNING TIMES (IN SECONDS). THE ROW ***avg***. SHOWS THE AVERAGE RUNNING TIME OF EACH METHOD

	CART	Bagg	Ada	Asym	SMB	Under	Over	SMOTE	Chan	Easy	Casc
abalone	0.21	8.39	18.06	18.04	35.51	3.47	39.11	7.78	4.83	3.72	6.22
balance	0.03	0.96	2.16	2.20	2.72	0.53	3.41	0.99	0.97	0.65	0.85
car	0.09	2.50	6.42	5.71	9.70	3.18	9.10	5.14	4.20	3.10	4.79
cmc	0.25	6.64	8.64	9.01	11.54	4.42	11.44	7.85	6.06	4.51	7.33
haberman	0.03	0.71	1.35	1.15	1.32	0.84	1.26	1.11	1.26	0.80	0.86
ionosphere	0.09	2.11	2.30	2.19	2.77	1.75	2.30	2.87	2.46	1.70	2.83
letter	0.41	19.70	153.47	138.11	1120.99	3.92	549.73	9.72	5.19	3.87	5.62
phoneme	0.34	9.72	23.20	22.87	150.09	12.03	38.78	30.18	16.67	11.64	20.12
pima	0.07	2.51	3.42	3.58	4.91	2.51	3.97	4.22	4.24	2.37	2.38
sat	0.78	27.74	54.83	53.29	102.84	9.62	116.83	21.24	11.66	10.08	13.96
wdbc	0.06	2.05	2.53	2.42	3.44	1.70	2.62	3.03	2.47	1.93	2.63
wpbc	0.06	1.97	2.04	1.85	2.26	1.00	2.29	2.01	1.17	1.27	1.50
vehicle	0.13	4.54	5.82	5.67	7.63	2.90	6.58	5.90	4.28	3.17	3.69
housing	0.08	2.17	2.66	2.92	3.84	1.32	3.35	2.42	1.89	1.15	0.77
mf-morph	0.06	2.06	5.69	5.78	11.62	1.08	11.22	2.34	1.60	1.17	2.57
mf-zernike	0.50	17.47	24.74	23.28	35.65	5.01	37.47	10.26	5.39	4.91	11.81
avg.	0.20	6.95	19.83	18.63	94.18	3.45	52.47	7.31	4.65	3.50	5.50

Comparison: performance

TABLE IV
AUC OF THE COMPARED METHODS (PART 1). THIS TABLE SHOWS RESULTS FOR DATA SETS ON WHICH ADABOOST'S AUC IS HIGHER THAN 0.95. FOR EACH METHOD AND EACH DATA SET, THE AVERAGE AUC IS FOLLOWED BY A STANDARD DEVIATION. THE COLUMN ***avg.*** SHOWS THE AVERAGE AUC OF EACH METHOD

AUC	*car*	*ionosphere*	*letter*	*phoneme*	*sat*	*wdbc*	***avg.***
Bagg	.995 ± .000	.962 ± .004	.997 ± .001	.955 ± .001	.946 ± .001	.987 ± .001	.974 ± .020
Ada	.998 ± .000	.978 ± .003	1.000 ± .000	.965 ± .000	.953 ± .001	.994 ± .001	.981 ± .018
Asym	.998 ± .000	.979 ± .002	1.000 ± .000	.965 ± .001	.953 ± .001	.994 ± .000	.982 ± .018
Under	.989 ± .001	.973 ± .002	1.000 ± .000	.953 ± .001	.941 ± .001	.993 ± .001	.975 ± .021
SMOTE	.995 ± .000	.978 ± .002	1.000 ± .000	.964 ± .000	.946 ± .001	.994 ± .001	.979 ± .019
Chan	.996 ± .000	.979 ± .002	1.000 ± .000	.960 ± .000	.955 ± .000	.993 ± .000	.980 ± .018
Cascade	.996 ± .000	.976 ± .002	1.000 ± .000	.962 ± .000	.949 ± .001	.994 ± .000	.979 ± .019
Easy	.994 ± .000	.974 ± .002	1.000 ± .000	.958 ± .000	.947 ± .000	.993 ± .000	.978 ± .020

Datasets with good overall class separability

Comparison: performance

TABLE V
AUC OF THE COMPARED METHODS (PART 2). THIS TABLE SHOWS RESULTS FOR DATA SETS ON WHICH ADABOOST'S AUC IS LOWER THAN 0.95

AUC	*abalone*	*balance*	*cmc*	*haberman*	*housing*	*avg.*
Bagg	.824 ± .002	.439 ± .018	.705 ± .004	.669 ± .014	.825 ± .011	.757 ± .129
Ada	.811 ± .001	.616 ± .009	.675 ± .008	.641 ± .015	.815 ± .010	.760 ± .088
Asym	.812 ± .003	.619 ± .012	.675 ± .010	.639 ± .015	.815 ± .010	.761 ± .088
SMB	.818 ± .002	.599 ± .010	.687 ± .011	.646 ± .006	.824 ± .014	.763 ± .092
Under	.830 ± .002	.617 ± .011	.671 ± .007	.646 ± .010	.805 ± .007	.769 ± .100
Over	.817 ± .002	.540 ± .010	.675 ± .008	.637 ± .017	.821 ± .010	.751 ± .103
SMOTE	.831 ± .001	.617 ± .015	.680 ± .008	.647 ± .017	.816 ± .008	.772 ± .097
Chan	.850 ± .001	.652 ± .011	.696 ± .006	.638 ± .008	.811 ± .010	.781 ± .097
Cascade	.828 ± .002	.637 ± .011	.686 ± .007	.653 ± .012	.808 ± .008	.778 ± .093
Easy	.847 ± .002	.633 ± .008	.704 ± .008	.668 ± .011	.825 ± .008	.787 ± .096

Datasets with less well defined class separability

Comparison

RAMOBoost: Ranked Minority Oversampling in Boosting

Sheng Chen, *Student Member, IEEE,* Haibo He, *Member, IEEE,* and Edwardo A. Garcia

Optimize the metric of interest

Dataset	Methods	OA	Precision	Recall	F-measure	G-mean	AUC
Page_Blocks	RAMOBoost	**0.9702**	0.8326	0.8928	**0.8614**	**0.9349**	**0.98899**
	SMOTEBoost	0.9696	**0.8340**	0.8825	0.8573	0.9297	0.98772
	SMOTE	0.9594	0.7781	0.8563	0.8140	0.9118	0.97993
	ADASYN	0.9251	0.5862	**0.9414**	0.7223	0.9322	0.97621
	AdaCost	0.9704	0.7912	0.8559	0.8469	0.9175	0.98861
	BorderlineSMOTE	0.9463	0.7853	0.8713	0.8171	0.912	0.97063
	SMOTE-Tomek	0.9576	0.7832	0.8627	0.8168	0.9139	0.97754
Satimage	RAMOBoost	0.9195	0.5671	0.7127	**0.6312**	0.819	**0.94860**
	SMOTEBoost	**0.923**	**0.5867**	0.6717	0.6276	0.7986	0.94678
	SMOTE	0.8977	0.4791	0.606	0.5327	0.7465	0.89748
	ADASYN	0.8422	0.3645	0.8431	0.5084	**0.8424**	0.92234
	AdaCost	0.9217	0.552	0.5426	0.371	0.7118	0.93255
	BorderlineSMOTE	0.8938	0.685	**0.9652**	0.3752	0.7598	0.90189
	SMOTE-Tomek	0.8957	0.701	0.9361	0.3679	0.773	0.90251
Mf_Zernike	RAMOBoost	0.8718	0.3608	0.369	0.3645	0.584	0.89452
	SMOTEBoost	0.8701	0.3544	0.364	0.3584	0.5798	0.89537
	SMOTE	0.8838	0.4409	0.592	0.5045	0.7356	0.8922
	ADASYN	0.8634	0.408	**0.809**	**0.5419**	**0.838**	0.90609
	AdaCost	0.8851	0.3604	0.446	0.3935	0.6441	**0.90656**
	BorderlineSMOTE	0.8827	0.3678	0.598	0.4217	0.7377	0.8924
	SMOTE-Tomek	**0.8877**	**0.3839**	0.745	0.4518	0.8199	0.90194

In summary

- Boosting methods tend to work better (with or without re-sampling)
- Choose the least costly re-sampling technique to scale

Train In Data

THANK YOU

www.trainindata.com