Quantile-Based Balanced Sampling: A Novel Algorithm for Addressing Class Imbalance in Machine Learning

Abstract

Class imbalance is a prevalent issue in many real-world classification problems, where the distribution of samples among classes is skewed. Traditional machine learning algorithms may exhibit poor performance on minority classes, as they are primarily designed for balanced data. In this paper, we propose a novel resampling technique called Quantile-Based Balanced Sampling (QBS) for addressing class imbalance in multi-class classification problems. The QBS algorithm selects representative samples from the majority class based on quantiles of feature distributions, generating a balanced dataset without the need for any assumptions about the underlying data distribution. We evaluate the performance of QBS against established undersampling methods using benchmark datasets and demonstrate its effectiveness in achieving improved classification results.

Introduction

Class imbalance is a common issue encountered in various classification problems, such as fraud detection, medical diagnosis, and text classification. Standard classification algorithms, including decision trees, support vector machines, and neural networks, often produce suboptimal results when applied to imbalanced datasets. To tackle this problem, researchers have developed several methods to balance the dataset, such as oversampling, undersampling, and cost-sensitive learning. However, these methods have their limitations, such as generating synthetic samples, introducing noise, or requiring adjustments to the learning algorithms. This paper introduces a novel algorithm called Quantile-Based Balanced Sampling (QBS), which leverages the quantiles of feature distributions to select representative samples from the majority class, thus generating a balanced dataset.

Quantile-Based Balanced Sampling Algorithm

The QBS algorithm consists of the following steps:

1. Count the unique non-minority class labels (c), minority class samples (m), and features (f).

2. Create an empty set (d) and add all minority class samples to it.
3. Calculate the number of quantiles (q) such that $f^q = c \cdot m$.
4. Calculate the 'q' quantiles for each feature.
5. Generate a set of all permutations of 'c' quantiles (p).
6. Sort the non-minority class samples by their distance to each quantile for each feature.
7. For each quantile permutation in 'p', add the closest non-minority class sample to set 'd'.
8. Return the balanced dataset 'd'.

Experiments and Results

We compared the performance of QBS against established undersampling methods on two benchmark datasets, Iris and Wine, using a decision tree classifier. The results demonstrated that QBS achieved the highest F1-score among all compared methods, highlighting its effectiveness in addressing class imbalance. Furthermore, QBS produced a balanced dataset without introducing synthetic samples or noise, which could potentially improve the performance of other classification algorithms as well.

Conclusion

In this paper, we presented the Quantile-Based Balanced Sampling (QBS) algorithm, a novel approach to address class imbalance in multi-class classification problems. The QBS algorithm demonstrated improved classification performance compared to existing undersampling methods and successfully generated a balanced dataset without relying on any distributional assumptions. Future research can investigate the applicability of QBS in conjunction with other classification algorithms, as well as exploring the algorithm's potential in dealing with large-scale, high-dimensional datasets.

References

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263-1284.

López, V., Fernández, A., García, S., Palade, V., & Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. Information Sciences, 250, 113-141.

Appendix

Datasets

Iris Dataset

- Source: UCI Machine Learning Repository
- Number of samples: 150
- Number of features: 4
- Number of classes: 3

Wine Dataset

- Source: UCI Machine Learning Repository
- Number of samples: 178
- Number of features: 13
- Number of classes: 3

Comparative Under-sampling Methods

- AllKNN: All K-Nearest Neighbors
- ClusterCentroids: Cluster Centroids
- CondensedNearestNeighbour: Condensed Nearest Neighbor
- EditedNearestNeighbours: Edited Nearest Neighbors
- InstanceHardnessThreshold: Instance Hardness Threshold
- NearMiss: Near Miss
- NeighbourhoodCleaningRule: Neighborhood Cleaning Rule
- OneSidedSelection: One-Sided Selection
- RandomUnderSampler: Random Under-Sampler
- RepeatedEditedNearestNeighbours: Repeated Edited Nearest Neighbors
- TomekLinks: Tomek Links

Experimental Setup

Data Preparation

We begin by loading the datasets and splitting them into training and testing sets using a 50% split ratio. The training set is then artificially imbalanced by reducing the number of minority class samples by 50%. This simulates the presence of class imbalance in real-world data.

Preprocessing

The data is preprocessed using standard techniques. Features are scaled using StandardScaler from scikit-learn to ensure that all features have the same scale.

Model Training and Evaluation

For each undersampling method, we resample the imbalanced training dataset and train a DecisionTreeClassifier on the resampled data. We chose a decision tree classifier due to its interpretability and ability to handle both numerical and categorical features. The trained classifier is then evaluated on the untouched testing dataset.

Performance Comparison

Table A.1: Performance metrics for each under-sampling method on the Iris dataset.

method	samples	class_distribution	accuracy	precision	recall	f1_score
QBS	39	{2: 15, 1: 13, 0: 11}	1	1	1	1
NeighborhoodCleaningRule	58	{1: 24, 2: 23, 0: 11}	0.96	0.961538	0.956522	0.956336
TomekLinks	63	{1: 26, 2: 26, 0: 11}	0.946667	0.944127	0.942029	0.941919
RandomUnderSampler	33	{0: 11, 1: 11, 2: 11}	0.946667	0.950617	0.942029	0.941587
AllKNN	57	{1: 24, 2: 22, 0: 11}	0.933333	0.940476	0.927536	0.92667
CondensedNearestNeighbour	14	{0: 11, 1: 2, 2: 1}	0.933333	0.940476	0.927536	0.92667
EditedNearestNeighbours	54	{1: 22, 2: 21, 0: 11}	0.933333	0.940476	0.927536	0.92667
InstanceHardnessThreshold	36	{2: 14, 0: 11, 1: 11}	0.933333	0.940476	0.927536	0.92667
RepeatedEditedNN	54	{1: 22, 2: 21, 0: 11}	0.933333	0.940476	0.927536	0.92667
ClusterCentroids	33	{0: 11, 1: 11, 2: 11}	0.88	0.90625	0.869565	0.864373

NearMiss	33	{0: 11, 1: 11, 2: 11}	0.84	0.885714	0.826087	0.813387
OneSidedSelection	14	{0: 11, 1: 3}	0.693333	0.5	0.666667	0.555556

Table A.2: Performance metrics for each under-sampling method on the Wine dataset.

method	samples	class_distribution	accuracy	precision	recall	f1_score
QBS	41	{1: 17, 0: 13, 2: 11}	0.932584	0.941126	0.934938	0.93779
TomekLinks	65	{1: 31, 2: 21, 0: 13}	0.910112	0.922454	0.915033	0.918255
InstanceHardnessThreshold	39	{0: 13, 1: 13, 2: 13}	0.910112	0.908586	0.900178	0.903062
NearMiss	39	{0: 13, 1: 13, 2: 13}	0.853933	0.861871	0.860665	0.860147
NeighborhoodCleaningRule	55	{1: 26, 2: 16, 0: 13}	0.853933	0.875238	0.867201	0.85311
RandomUnderSampler	39	{0: 13, 1: 13, 2: 13}	0.842697	0.862043	0.825312	0.833182
ClusterCentroids	39	{0: 13, 1: 13, 2: 13}	0.808989	0.815108	0.820559	0.817629
CondensedNearestNeighbour	21	{0: 13, 1: 4, 2: 4}	0.775281	0.769542	0.781937	0.766123
OneSidedSelection	25	{0: 13, 2: 9, 1: 3}	0.730337	0.825	0.743316	0.736667
EditedNearestNeighbours	38	{1: 18, 0: 13, 2: 7}	0.719101	0.690453	0.695781	0.690821
RepeatedEditedNN	38	{1: 18, 0: 13, 2: 7}	0.719101	0.690453	0.695781	0.690821
AllKNN	45	{1: 22, 0: 13, 2: 10}	0.662921	0.583276	0.609329	0.588449

GitHub Repository

https://github.com/splch/qbs