Handling Class Imbalance

Introduction

Class imbalance is a common and significant challenge in machine learning and deep learning that arises when the distribution of classes in a dataset is highly skewed. In such scenarios, one or more classes (often called "minority classes") have substantially fewer instances than others ("majority classes"). This phenomenon is often present in real-world applications such as fraud detection, medical diagnosis, anomaly detection, and rare event prediction, where the events of interest are rare but important. Understanding and properly handling class imbalance is essential for developing robust models that can generalize well and perform reliably on poorly-represented classes.

Class imbalance negatively-affects the learning process of many standard machine learning algorithms. Most models are designed under the assumption that the training data is representative of the true distribution of the problem space. When this assumption is invalid due to imbalance, models tend to be biased toward the majority class, as optimizing for overall accuracy favors predictions of the prevalent class. Consequently, this leads to high accuracy theoretically, but poor real-world utility, especially when the minority class carries the greater importance — for instance, identifying malignant tumors in medical images or detecting fraudulent financial transactions. The severity of the imbalance can vary widely. It might be mild, with a class distribution like 70:30, or extreme, with ratios as skewed as 99:1 or worse. As the imbalance increases, the difficulty of learning meaningful patterns for the minority class also increases. This is due to the reduced statistical representation of that class in the training set, which can prevent the model's ability to generalize. In highly imbalanced datasets, minority class instances may be treated as noise or outliers by many standard algorithms.

In the context of supervised learning, the impact of class imbalance is felt across the entire machine learning pipeline. It influences model selection, training dynamics, evaluation metrics, and even data preprocessing decisions. For example, conventional evaluation metrics such as accuracy become misleading in imbalanced settings, as a naive model that always predicts the majority class can still achieve high accuracy despite being useless. Deep neural networks, particularly convolutional and recurrent architectures used in domains such as image analysis and forecasting, can easily overfit to the majority class if exposed to imbalanced data. Despite their high capacity for representation, these networks are not immune to bias introduced by skewed class distributions. Their depth and complexity can sometimes amplify such biases if proper care is not taken during training and evaluation. In deep learning, class imbalance also affects the training dynamics in subtle ways. For example, the gradients computed during backpropagation are dominated by the majority class, which can suppress learning the minority class. This leads to poor feature representations and misclassifications for minority class instances. Moreover, imbalanced datasets in deep learning often result in decision boundaries that are skewed toward the majority class, increasing false negatives and reducing the model’s ability to detect rare but important patterns.

Another important aspect of class imbalance is its impact on model interpretability and trustworthiness. In applications such as healthcare, criminal justice or finance, models that perform poorly on minority classes can have serious ethical and practical consequences. A classifier that misses rare diseases or disproportionately misclassifies marginalized groups may perpetuate harmful biases and erode user trust. Thus, addressing class imbalance is not only a technical challenge but also an important consideration for fairness and accountability in machine learning systems. Furthermore, class imbalance intersects with issues such as dataset quality, label noise, and concept drift. In real-world datasets, minority class examples are not only fewer but also more likely to contain noisy or ambiguous labels, due to the difficulty of annotating rare events. This adds to the challenge of learning accurate decision boundaries.

Symptoms of Class Imbalance

Class imbalance in machine learning and deep learning often manifests through a range of symptoms that can impact model performance, interpretability, and fairness. These symptoms, which emerge throughout the model development pipeline, can be misleading if not properly recognized. While the root cause is a skewed distribution of class labels, the observable consequences affect training dynamics, evaluation outcomes, model generalization, and system-level behavior. Some of the symptoms of class imbalance include:

- **Low recall:** Recall (aka true positive rate) measures the proportion of actual positive cases that are correctly identified. In imbalanced settings, minority class recall often drops significantly. This happens because the model is exposed to so few examples of the minority class during training that it fails to learn distinguishing features. As a result, it frequently misclassifies minority class samples as majority ones, leading to a high false negative rate.
- **Biased decision boundaries:** When training on imbalanced datasets, many learning algorithms develop biased decision boundaries that favor the majority class. These boundaries tend to shift toward the minority class space, effectively marginalizing it. This

symptom is particularly pronounced in models that learn discriminative boundaries, such as logistic regression, SVMs, and deep neural networks. The visual manifestation of this issue (in 2D space) is a decision boundary that encloses very little of the minority class region, even when it is separable (Fig. 1).

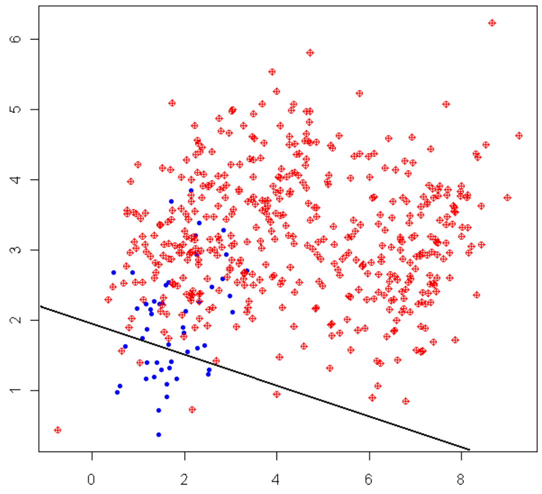

Fig. 1. Example of bias towards the majority class in linear classification. The decision boundary (the line) results in poor predictions for the moniroty class (blue points).

- **Imbalanced confusion matrix:** In such a case, the matrix shows a very high number of true positives/true negatives for the majority class but low true positives for the minority class. Most minority class samples are classified into the majority class, resulting in high false negatives (Fig. 2.). This pattern reflects a model that has failed to learn meaningful distinctions for the minority class, despite potentially high overall performance metrics.

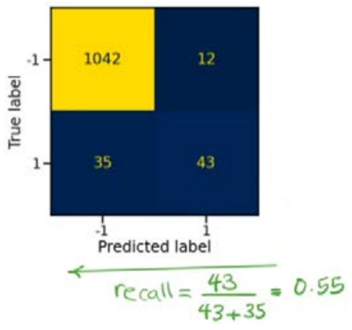

Fig. 2. Confusion matrix of a binary classifier applied to an unbalanced data set.

- **Model overconfidence in majority class predictions:** The model tends to assign high probability scores to majority class predictions, even when inputs are ambiguous or resemble minority class instances. This results from the model learning representations of the problem space where the minority class has little influence on the learned feature distribution.

- **Poor generalization to minority class instances:** Even when a model appears to perform reasonably on test data, it may generalize poorly to unseen instances of the minority class. This becomes evident when the model is deployed or evaluated on real-world data containing new, slightly varied instances of the minority class. Because the model has seen few examples of the minority class during training, it fails to generalize to variations of usch examples.

Imbalance Mitigation Strategies

Addressing class imbalance is a central concern in the development of effective and fair machine learning and deep learning systems. Once an imbalance in class distribution has been identified and its symptoms understood, appropriate strategies must be used to mitigate its negative effects. These strategies aim to enhance the model's capacity to learn meaningful patterns from both majority and minority classes, and to ensure that performance metrics more accurately reflect the model's utility, especially in contexts where the minority class represents critical or high-risk outcomes.

There are many categories of strategies for handliong class imbalances, but they can be roughly categorized into: data-level, algorithm-level and evaluation-level approaches.

- **Data-level strategies** aim to rebalance the class distribution within the dataset, typically before training occurs. These methods operate by modifying the input data to either increase the representation of the minority class or decrease that of the majority class.
- **Algorithm-level strategies** address class imbalance by directly modifying the learning process, rather than the data. These strategies change how the model perceives errors or updates its parameters during training, typically by adjusting loss functions, decision thresholds, or optimization dynamics to prioritize minority classes.
- **Evaluation-level strategies** focus on the fair and relevant assessment of model performance in the presence of class imbalance. In many imbalanced scenarios, conventional evaluation metrics such as accuracy, which measure overall correctness, become misleading because they are dominated by the performance on the majority class. Evaluation-level interventions aim to shift focus toward metrics that provide a clearer picture of how well the model is performing on the minority class.

Concrete Methods for Imbalance Management

In the following paragraphs, we briefly explain several techniques that are meant to alleviate the effects of class imbalance, that fall into the above-mentioned strategy categories.

Cost-sensitive Learning

This approach involves modifying the training objective so that errors make on the minority (positive) class are penalized more heavily tha those on the majority (negative) class. The idea is to guide the learning algorithm to focus more on correctly classifying rare but important instances.
In an imbalanced dataset, the model tends to focus on the majority class because those instances dominate the loss function during training. Therefore, we assign higher weights to the minority class, thus encouraging the model to reduce errors on rarer instances from that class.
Assuming a binary classification problem with labels $y \in \{0, 1\}$, where $y = 1$ is the minority class, for a Loss function $L(y, \hat{y})$, a common weighted formulation is:

$$L_{\text{weighted}}(y,\hat{y}) = w_1 \cdot y \cdot L(1,\hat{y}) + w_0 \cdot (1-y) \cdot \mathcal{L}(0,\hat{y})$$

where:
w_1 is the weight assigned to the positive (minority) class
w_0 is the weight assigned to the negative (majority) class
$\hat{y}$ is the predicted output
L is a standard loss function (binary cross entropy)

Usually, the weights are chosen inversely-proportional to class frequencies:

$$w_c = \frac{N}{C \cdot N_c}$$

where:
N is the total number of instances
C is the number of classes (typically 2)
N_c the number of instances in class C

Oversampling

Oversampling is a data-level strategy where the idea is to increase the number of instances in the minority class so that it is better represented during training. We present three of the most popular oversampling techniques:

- **Random Oversampling** – the simplest form of oversampling, it involves duplicating existing instances of the minority class until the class distribution is more balanced (or fully balanced) (Fig. 3, Fig. 4) . The duplicated instances are exact copies of original ones.

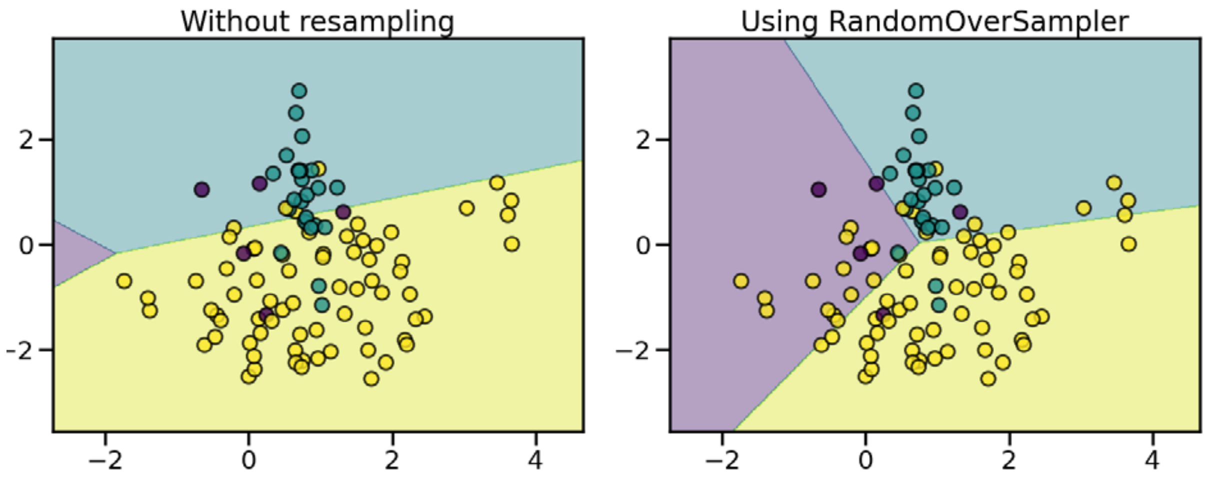

Fig. 3. Examples of decision regions learned with and without random oversampling

- **SMOTE (Synthetic Minority Over-sampling Technique)** – generates minority class instances rather than duplicating them. For each minority class sample, SMOTE selects one or more of its nearest neighbors (in feature space) from the same class, and generates new samples by interpolating between feature vectors (Fig. 4). For a minority instance *x*, and one of its k-nearest neighbors x_{nn} , the new instance is created as:

$$x_{\text{new}} = x + \delta \cdot (x_{nn} - x), \quad \text{where } \delta \sim \mathcal{U}(0,1)$$

- **ADASYN (Adaptive Synthetic Sampling)** - a refined version of SMOTE that adapts the number of samples generated for each minority instance based on how hard it is to learn. Specifically, it generates more synthetic samples for minority class instances that are difficult to classify, for example those surrounded by majority class samples. ADASYN computes a density distribution based on the number of majority class neighbors and allocates more synthetic data to regions with higher classification difficulty. This makes the model focus on challenging areas of the decision space (Fig. 4).

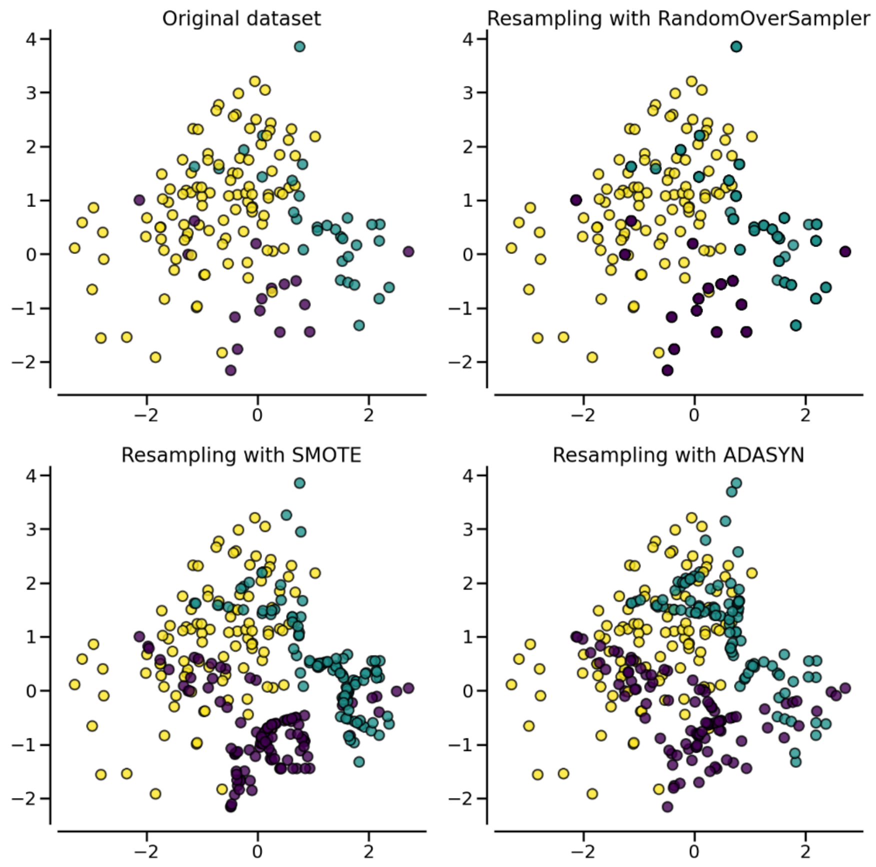

Fig. 4. Examples of the results of oversampling using random oversampling, SMOTE and ADASYN.

Undersampling

Undersampling is a data-level strategy for dealing with class imbalance that focuses on reducing the number of majority class instances in the training dataset. The aim is to rebalance the class distribution by discarding some of the majority class examples, thereby preventing the model from being biased toward the dominant class. Unlike oversampling, which generates or duplicates minority class data, undersampling simplifies the training data by eliminating excess examples. We present three of popular undersampling techniques:

- **Random Undersampling** – randomly remove instances form the majority class until the class distribution is more balanced (for ex., equal class sizes)
- **Tomek Links –** a method that removes borderline or potentially-ambiguous majority class instances. A Tomek link is defined as a pair of nearest neighbor instances (x_i, x_j) such that x_i and x_j belong to different classes and they are each other's nearest neighbor. If a Tomek link exists between a majority and a minority class instance, it indicates a possible class overlap or noise near the boundary. Consequently, the majority class instance in such pairs is removed.
- **NearMiss** – a family of undersampling techniques that selects majority class instances based on their proximity to minority class instances, with the goal of retaining majority instances that are hard to distinguish from the minority class. In the simplest form (NearMiss-1), it selects majority class instances that have the smallest average distance to the k-nearest minority class instances. It keeps only those majority instances that are closest to the minority class. The idea is to preserve the decision boundary by keeping majority instances close to it, while removing redundant or distant majority instances, which may not contribute to learning.

Tasks

For completing the tasks, the code samples that are provided with this documentation can be used as starting points.

Train a neural-network-based model on an imbalanced dataset for binary classification. Evaluate the dataset by computing the following metrics:

- accuracy
- precision = TP / (TP + FP)
- recall = TP / (TP + FN)
- F1 = (2 * precision * recall) / (precision + recall)

  where:

  TP = true positives (number of times the network predicts label 1 when the correct label is also 1)

  FP = false positives (number of times the network predicts label 1 when the correct label is 0)

FN = false negatives (number of times the network predicts label 0 when the correct label is 1)

If there is a significant class imbalance not all of these statistics will be maximized – for instance, expect high accuraccies but low recall values.

Use various strategies for handling class imbalance, such as:

- Weighing the positive class – assign a higher weight to the positive class when computing the loss – the loss function *BCEWithLogitsLoss* allows such a weight to be specified as one of its parameters. A typical weight could be:

w_p = number of instances in the majority class / number of instances in the minority class

- Using various oversampling / undersampling techniques (on the training set), such as:
    - Random oversampling
    - SMOTE
    - ADASYN
    - Tomek Links
    - NearMiss

These methods are available in the *imbalanced-learn* library: https://imbalanced-learn.org/

The objective is to maximize the four above-mentioned metrics, used for evaluation.