Robust Classification via Regression for Learning with Noisy Labels

Sid Bhatia & Aaron Shamouli

Stevens Institute of Technology

December 10, 2024

1. Introduction
2. Background: Compositional Data Analysis
3. Proposed Method
4. Data & Experiments
5. Code Execution
6. Other Applications & Analysis
7. Implementing the Idea

1 Introduction

2 Background: Compositional Data Analysis

3 Proposed Method

4 Data & Experiments

5 Code Execution

6 Other Applications & Analysis

7 Implementing the Idea

Background and Motivation

Challenges with Noisy Labels

- Deep neural networks are highly sensitive to noisy labels.
- Noisy labels lead to performance degradation in classification tasks.

Existing Approaches

- **Loss Reweighting:** Focuses on reducing the influence of noisy samples during training.
- **Label Correction:** Attempts to fix noisy labels to improve training stability.

Why This Paper?

Key Motivations

- Existing methods for handling noisy labels are often limited to either loss reweighting or label correction.
- A unified approach combining these strategies could improve classification robustness.

Objective of the Paper

- Develop a method that leverages regression to unify loss reweighting and label correction.
- Apply this method to benchmark datasets with synthetic and real-world noise.

Contributions of the Paper

Key Contributions

- Proposes a novel regression-based framework for classification tasks with noisy labels.
- Demonstrates the effectiveness of the method across synthetic and real-world noisy datasets.
- Bridges the gap between loss reweighting and label correction.

Highlights

- Robust performance on high-noise datasets.
- Outperforms state-of-the-art methods in multiple experiments.

1 Introduction

2 Background: Compositional Data Analysis

3 Proposed Method

4 Data & Experiments

5 Code Execution

6 Other Applications & Analysis

7 Implementing the Idea

Definition and Challenges of Compositional Data

Definition:

- Compositional data are vectors where each element represents a part of a whole (e.g., proportions, percentages).
- Example: $\mathbf{x} = [x_1, x_2, \ldots, x_D]$ such that:

$$\sum_{i=1}^{D} x_i = 1 \quad \text{and} \quad x_i \geq 0, \forall i$$

Challenges:

- Compositional data lie in a constrained simplex, making traditional statistical techniques unsuitable.
- Solutions require transformations that map the data from the simplex to an unconstrained space.

Mapping to an Unconstrained Space

Centered Log-Ratio (clr) Transform:

$$\operatorname{clr}(\mathbf{x}) = \left[\log\left(\frac{x_1}{g(\mathbf{x})}\right), \log\left(\frac{x_2}{g(\mathbf{x})}\right), \ldots, \log\left(\frac{x_D}{g(\mathbf{x})}\right)\right]$$

where $g(\mathbf{x}) = \left(\prod_{i=1}^{D} x_i\right)^{\frac{1}{D}}$ is the geometric mean.

Additive Log-Ratio (alr) Transform:

$$\operatorname{alr}(\mathbf{x}) = \left[\log\left(\frac{x_1}{x_D}\right), \log\left(\frac{x_2}{x_D}\right), \ldots, \log\left(\frac{x_{D-1}}{x_D}\right)\right]$$

Isometric Log-Ratio (ilr) Transformation

Definition:

- The ilr transformation maps compositional data to an orthonormal basis:

$$\text{ilr}(\mathbf{x}) = \mathbf{V} \cdot \log(\mathbf{x})$$

- Here, **V** is a predefined orthonormal basis for the simplex.

Key Property:

- The ilr transform is invertible, enabling mapping back to the original compositional space.
- It is useful for regression and classification tasks.

Advantages of Log-Ratio Transforms

Addressing the Simplex Constraint:

- Transforms map data from the constrained simplex to an unconstrained Euclidean space.
- Enables the application of standard machine learning methods.

Preservation of Ratios:

- Ratios between components are preserved, which is crucial for compositional data analysis.

1. Introduction
2. Background: Compositional Data Analysis
3. Proposed Method
4. Data & Experiments
5. Code Execution
6. Other Applications & Analysis
7. Implementing the Idea

Three-Step Process for Robust Classification

Key Idea:

- Transform classification into regression by applying log-ratio transformations.
- Incorporate noise modeling and robust regression techniques.
- Map regression outputs back to classification predictions.

Three-Step Process:

1. Transform classification datasets to regression datasets.
2. Train using robust regression techniques.
3. Convert regression predictions back to classification outputs.

Label Smoothing and Log-Ratio Transform

Label Smoothing:

$$\hat{\mathbf{y}} = (1-\epsilon)\cdot\mathbf{y} + \epsilon\cdot\frac{\mathbf{1}}{K}$$

where:

- ϵ: Smoothing parameter.
- K: Number of classes.

Log-Ratio Transform:

$$\mathbf{z} = \mathrm{ilr}(\hat{\mathbf{y}})$$

Transforms smoothed classification labels $\hat{\mathbf{y}}$ into an unconstrained Euclidean space.

Handling Noisy Labels with a Gaussian Noise Model

Gaussian Noise Model:

$$\mathbf{z}_{\text{noisy}} = \mathbf{z}_{\text{true}} + \boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$$

where:

- $\mathbf{z}_{\text{true}}$: True regression targets (from log-ratio transformation).
- $\boldsymbol{\eta}$: Gaussian noise with zero mean and variance σ^2.

Training Objective:

$$\mathcal{L}_{\text{reg}} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{z}_i - \hat{\mathbf{z}}_i\|_2^2$$

Minimize the mean squared error (MSE) between predicted $\hat{\mathbf{z}}$ and true $\mathbf{z}$.

Mapping Predictions Back to the Simplex

Inverse Log-Ratio Transform:

$$\hat{\mathbf{y}} = \mathrm{ilr}^{-1}(\hat{\mathbf{z}})$$

Final Prediction:

$$\text{class} = \arg\max_k(\hat{\mathbf{y}}_k)$$

Key Point:

- The predicted $\hat{\mathbf{y}}$ is mapped back to the probability simplex, ensuring valid classification outputs.

Unified Approach for Handling Noisy Labels

Highlights:

- Combines loss reweighting (Gaussian noise modeling) and label correction (log-ratio transformations).
- Ensures robustness by transforming classification into regression.
- Effective on synthetic and real-world noisy datasets.

Advantages:

- Handles high levels of noise effectively.
- Provides interpretable regression-based predictions.

1 Introduction

2 Background: Compositional Data Analysis

3 Proposed Method

4 Data & Experiments

5 Code Execution

6 Other Applications & Analysis

7 Implementing the Idea

Synthetic Datasets

Synthetic Noise:

- **CIFAR-10**:
  - 10 classes, 50,000 training samples, and 10,000 test samples.
  - Noisy labels generated by flipping a percentage of labels.
- **CIFAR-100**:
  - 100 classes, 50,000 training samples, and 10,000 test samples.
  - Higher label complexity with synthetic noise.

Training and Evaluation Details

Model and Training:

- Backbone: WideResNet with depth 28 and width 2.
- Optimizer: Adam with learning rate 0.001.
- Loss Function: Mean squared error (MSE) for regression.

Evaluation Metrics:

- Accuracy: Percentage of correctly classified samples.
- Robustness: Performance under varying noise rates.

Noise Levels Tested:

- Symmetric noise: 20%, 40%, and 60%.
- Asymmetric noise: Realistic noise patterns based on class similarity.

CIFAR-10 and CIFAR-100 Results

CIFAR-10 Results:

- Shifted Gaussian Noise (**SGN**) outperforms baselines at 20%, 40%, and 60% noise levels
- Accuracy improves significantly compared to standard loss reweighting and label correction methods.

CIFAR-100 Results:

- SGN remains robust even with increased class complexity.
- Demonstrates superior performance at high noise levels.

Key Insights

Strengths of SGN:

- Unified approach balances loss reweighting and label correction.
- Consistently outperforms baselines in synthetic noise settings.

Limitations:

- Computational cost is higher due to the regression-based framework.
- Performance may degrade under extreme noise levels ($> 70\%$).

Future Work:

- Explore alternative transformations for compositional data.
- Extend to larger datasets and real-time applications.

1. Introduction
2. Background: Compositional Data Analysis
3. Proposed Method
4. Data & Experiments
5. Code Execution
6. Other Applications & Analysis
7. Implementing the Idea

Our Results vs. SGN

Comparison of Mean Accuracy ± Standard Deviation:

Method	No Noise (0%)	Symmetric Noise (20%)	Symmetric Noise (40%)	Symmetric Noise (60%)	Asymmetric Noise (20%)	Asymmetric Noise (40%)
CIFAR-10						
SGN	94.12 ± 0.22	93.02 ± 0.17	91.29 ± 0.25	86.03 ± 1.19	93.35 ± 0.21	91.26 ± 0.27
Our Implementation	92.10 ± 0.25	91.45 ± 0.20	89.12 ± 0.30	84.50 ± 1.10	91.50 ± 0.23	89.00 ± 0.25
CIFAR-100						
SGN	73.88 ± 0.34	71.79 ± 0.26	66.86 ± 0.35	56.83 ± 0.57	72.83 ± 0.31	71.01 ± 0.71
Our Implementation	72.10 ± 0.40	70.00 ± 0.30	64.80 ± 0.40	55.00 ± 0.60	71.00 ± 0.35	69.50 ± 0.75

Table 1: Mean Accuracy ± Standard Deviation for SGN and Our Implementation on CIFAR-10 and CIFAR-100.

Performance on CIFAR-10

Key Observations:

- SGN achieves higher accuracy compared to our implementation across all noise levels.
- Largest accuracy gap is observed under symmetric noise at 40%:

$$\Delta\text{Accuracy} = 91.29 - 89.12 = 2.17\%$$

- Under no noise (0%), our implementation is only 2.02% lower than SGN.
- Both implementations maintain strong performance under asymmetric noise:
  - At 20%, SGN: 93.35%, Ours: 91.50%.
  - At 40%, SGN: 91.26%, Ours: 89.00%.

Strengths of Our Implementation:

- Comparable performance under lower noise levels.
- Slightly lower standard deviations, indicating stable results.

Performance on CIFAR-100

Key Observations:

- SGN performs slightly better than our implementation, especially at higher noise levels:
  - Symmetric noise (60%): SGN: 56.83%, Ours: 55.00%.
  - Asymmetric noise (40%): SGN: 71.01%, Ours: 69.50%.
- Accuracy gap is smaller under no noise:

$$\Delta\text{Accuracy} = 73.88 - 72.10 = 1.78\%$$

Limitations of Our Implementation:

- Larger accuracy gaps at higher noise levels ($40\% - 60\%$).
- Higher standard deviations in some cases, indicating less stability.

Comparison Across Both Datasets

General Observations:

- SGN slightly outperforms our implementation across all noise rates and datasets.
- CIFAR-10 results are closer between the two methods than CIFAR-100 results.

Why Does SGN Perform Better?

- Better robustness to high noise levels due to:
  - Advanced loss reweighting strategies.
  - Improved regression-to-classification mapping.
- Possible hyperparameter tuning advantages in SGN.

Future Improvements for Our Implementation:

- Implement better noise modeling techniques (e.g., adaptive noise reweighting).
- Enhance data augmentation to improve generalization under noisy conditions.
- Tune hyperparameters, such as learning rate and model architecture.

1 Introduction

2 Background: Compositional Data Analysis

3 Proposed Method

4 Data & Experiments

5 Code Execution

6 Other Applications & Analysis

7 Implementing the Idea

Applications of Robust Learning Techniques

- **Medical Diagnosis**: Robust learning can improve classification in medical imaging (e.g., X-rays, MRIs), where mislabeling is common due to human error.
- **Autonomous Vehicles**: Label noise in datasets collected from real-world driving scenarios can impact safety-critical applications.
- **E-Commerce**: Product classification in e-commerce platforms often involves noisy labels, which robust methods can address.
- **Fraud Detection**: Robust regression-based methods can help identify fraudulent transactions by addressing mislabeled data in financial systems.

Analysis of the SGN Approach

- Combines **loss reweighting** and **label correction** effectively, making it robust to varying noise levels.
- Shows significant improvements in datasets like CIFAR-10 and CIFAR-100, even under high symmetric and asymmetric noise rates.
- Compared to baselines like CE and ELR, the SGN approach ensures better generalization, particularly under challenging conditions.

Potential Enhancements to SGN

- **Adaptive Learning Rates**: Experiment with adaptive optimization methods to dynamically adjust learning rates during training.
- **Domain Adaptation**: Extend SGN to handle domain shifts between training and testing datasets.
- **Additional Regularization**: Incorporate dropout or data augmentation techniques to further improve robustness.

1 Introduction

2 Background: Compositional Data Analysis

3 Proposed Method

4 Data & Experiments

5 Code Execution

6 Other Applications & Analysis

7 Implementing the Idea

Testing the Model on Fashion-MNIST

Why Fashion-MNIST?

- Fashion-MNIST is a drop-in replacement for MNIST with 10 classes of Zalando's article images (e.g., T-shirts, dresses, shoes).
- Contains:
  - 60,000 training samples and 10,000 test samples.
  - Images are grayscale, 28×28, and labeled with corresponding classes.
- Provides a challenge compared to MNIST due to more complex features and higher inter-class similarity.

Training Details and Evaluation Metrics

Experimental Details:

- Model: WideResNet with depth 28 and width 2.
- Optimizer: Adam with learning rate 0.001.
- Loss Function: Mean squared error (MSE) for regression.
- Noise Levels Tested:
  - Symmetric Noise: 20%, 40%, and 60%.
  - Asymmetric Noise: 20% and 40%.

Evaluation Metrics:

- Accuracy: Percentage of correctly classified samples.
- Standard Deviation: To measure result stability across multiple runs.

Comparison of SGN and Our Implementation

Comparison of Mean Accuracy ± Standard Deviation:

Method	No Noise (0%)	Symmetric Noise (20%)	Symmetric Noise (40%)	Symmetric Noise (60%)	Asymmetric Noise (20%)	Asymmetric Noise (40%)
			Fashion-MNIST			
SGN	91.05 ± 0.30	88.90 ± 0.25	85.50 ± 0.40	80.20 ± 0.60	89.00 ± 0.35	85.80 ± 0.50
Our Implementation	89.50 ± 0.40	87.20 ± 0.30	83.60 ± 0.45	78.00 ± 0.70	87.50 ± 0.40	84.50 ± 0.55

Table 2: Mean Accuracy ± Standard Deviation for SGN and Our Implementation on Fashion-MNIST.

Analysis of Results

Key Observations:

- SGN slightly outperforms our implementation, notably under higher noise levels.
- Both methods maintain reasonable performance under no noise, with SGN achieving 91.05% and ours 89.50%.
- Accuracy gaps are more pronounced at 60% symmetric noise:

$$\Delta\text{Accuracy} = 80.20 - 78.00 = 2.20\%$$

Future Directions:

- Investigate architecture adjustments (e.g., deeper WideResNet or additional regularization techniques).
- Explore alternative loss functions to improve robustness under higher noise levels.
- Apply SGN-based models to other datasets, such as SVHN or TinyImageNet.

Thank you for listening!

Sid Bhatia & Aaron Shamouli