Pattern Seekers - Group 45
Mahesh Desai - 50419649
Sagar Sonawane - 50431730
Shriram Ravi - 50419944

Pattern Recognition Project

Fall 2021

ABSTRACT

The goal of this project is to develop a robust learning system based on what we have learned in the pattern recognition class, which operates on a given MNIST dataset. We tried to do this as two separate parts. For the first part, we tried to solve several baselines, which made us get familiar with the whole pipeline and build the necessary fundamentals. The second part is the extension to our baselines.

INTRODUCTION

Modern real-world large-scale datasets often have long-tailed label distributions. On these datasets, conventional machine learning models and deep neural networks have been found to perform poorly on less represented classes. This is particularly detrimental if the testing criterion places more emphasis on minority classes. For example, accuracy on a uniform label distribution or the minimum accuracy among all classes are examples of such criteria. These are common scenarios in many applications due to various practical concerns such as transferability to new domains, fairness, and so on.

On the other hand, traditional machine learning models and deep neural networks (DNNs) are often requiring large-scale datasets with clean label annotations for proper training. However, labeling large-scale datasets is a costly and error-prone process, and even high-quality datasets are likely to contain noisy (incorrect) labels. Therefore, training accurate traditional machine learning models and DNNs in the presence of noisy labels has become a task of great practical importance in machine learning and deep learning.

In this project, we took 2 Machine Learning and 2 Deep Learning models to train the MNIST dataset, which is widely available, and try to optimize, improvise and give a different dimension to 2 models, one in each ML and DL by trying to build few techniques over existing and well-accomplished models.

DESCRIPTION

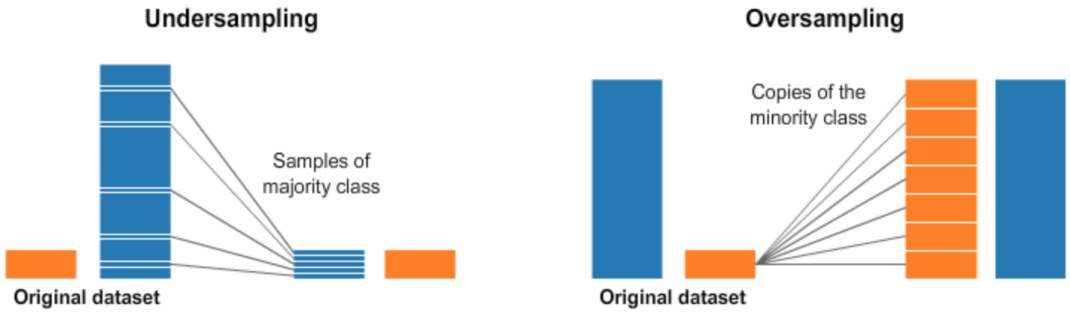
Here, we first explain a few of the terminologies we use and then proceed with giving a detailed overview of implementing the baseline and proposed models.

Dataset

MNIST (Modified National Institute of Standards and Technology) Dataset is a database consisting of handwritten digits. It contains a total of 70,000 images split into two groups – A training dataset of 60,000 images and a Testing dataset of size 10,000 images. All images in the dataset are grayscaled with a size of 28*28 pixels. It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal effort on preprocessing and formatting.

Dataset Imbalancing

We have considered the original MNIST dataset as a balanced dataset. But, in the real world, it's hard to find an equal amount of data for all the classes in a dataset. As a result, minority classes will have worse training and test errors. To solve this, we can do under-sampling or oversampling. We undersampled the dataset to create an imbalanced dataset (i.e. deleting some values to create an uneven distribution in the target classes). The undersampling method seeks to randomly select and remove samples from the majority class, consequently reducing the number of examples in the majority class in the transformed data. We have used the make_imbalance function from the imblearn library to undersample the data.

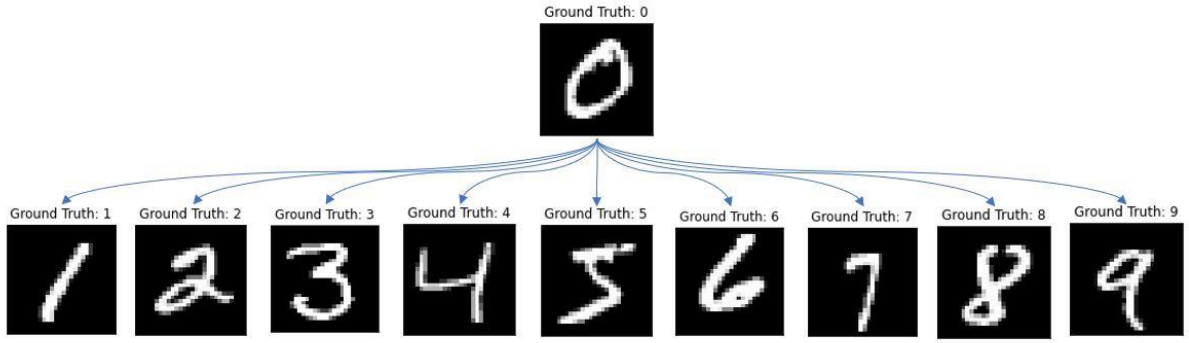


Adding Label Noise

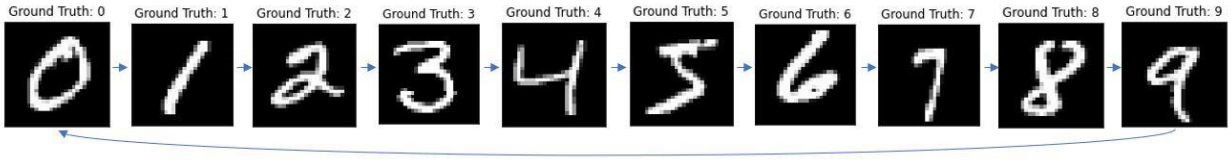
We have implemented two types of label noises – Symmetrical and Asymmetrical. Symmetrical Label Noise is when some randomly picked 'n' values from a class 'c' are assigned to a different

class ‘ci’. Asymmetrical Noise is when some randomly picked ‘n’ values from any class and are assigned to random different classes. We have set the noise ratio as 20% in our implementation.

Symmetric Noise:



Asymmetric Noise:



Machine Learning Models

We chose 2 Machine Learning models for training the MNIST dataset and thereby establishing out baseline measurements of how the models perform. We’re trying to classify a given data as a set of numbers. So, we chose 2 classification models. The 2 models we chose are,

1. Logistic Regression
2. Support Vector Machine (SVM)

Logistic Regression

Logistic Regression is a statistical model which uses a logistic function to classify a certain event in two classes. Extension of logistic regression is a Multi-class model.

Sigmoid Function: Takes an input and maps it between 0 and 1 in an “S” shaped manner.

Gradient Descent: Finds the optimal values of the parameters.

Cost: Calculates the cost that occurred to run the model.

Parameters: This multi-class model has been over 500 epochs with a learning rate of 0.25.

Support Vector Machine (SVM)

A support vector machine (SVM) is a machine learning algorithm that analyzes data for classification and regression analysis. SVM is a supervised learning method that looks at data and sorts it into one of two categories. We have extended SVM to be a multi-class model. We are implementing a pipeline class that allows us to run multiple processes such as fit, predict, and score.

Standard Scaler: Subtracts the mean from each feature and then scales the unit variance.

Parameters: This multi-class SVM model takes two parameters C = 0.001 and Gamma = 10

Deep Learning Models

Deep learning algorithm performance suffers heavily from a heavy class imbalance and noise in the datasets. We chose 2 Deep Learning models for training the MNIST dataset and thereby establishing out baseline measurements of how the models perform. We're trying to classify a given data as a set of numbers. So, we chose 2 classification models. The 2 models we chose are,

1. Symmetric cross-entropy Learning (SL)
2. LDAM-DRW

Symmetric cross-entropy Learning (SL)

The approach of Symmetric cross-entropy Learning (SL) was proposed to avoid overfitting to noisy labels. Cross entropy requires an extra noise-tolerant term to facilitate the learning of hard classes. We have used an SGD optimizer. Stochastic Gradient Descent (SGD) is an iterative method for optimizing an objective function with suitable smoothness properties. For better loss performance we used symmetric cross-entropy function. We have used Keras for evaluation metrics such as accuracy, precision, recall, and roc AUC score.

Parameters: This SL model iterates for 10 epochs with hyper parameters alpha = 1 (over-fitting issue), beta = 1 (for flexible exploration on robustness), noise ratio = 20%, batch size = 128

LDAM-DRW

Large-scale datasets are often long-tailed. As a result, minority classes will have worse training and testing errors. To solve this, the author has suggested the use of Label-Distribution-Aware Margin Loss with Deferred Re-weighting. LDAM only works for Imbalanced datasets. For minority samples, it uses stronger regularization and lenient regularization for majority samples. It uses

stronger models to train majority classes and weaker models to train minority classes. Deferred Re-weighting is the process of recombining the learning rate after each iteration. LDAM – DRW handles noisy datasets well but it leads to more training and testing errors.

Parameters: LDAM – DRW model runs for 10 epochs with 20% noise ratio where applicable, imbalance type is exponential and imbalance factor is 0.01.

Proposed extension on ML Models

In theory, the 2 classification models we used, logistic regression and SVM can only classify up to two classes. We have extended the given two ML models by making them viable for multi-class classification.

Proposed extension on DL Models

In theory, the LDAM-DRW model is designed to handle imbalanced datasets and the SL model is designed to handle noisy datasets. We extended the LDAM-DRW model to handle noisy datasets along with imbalanced ones. Also, we've made the SL model to handle imbalanced datasets. Furthermore, we propose a Neural Network model with a large number of layers as it would help with the complexity of the MNIST dataset. We think a model like VGG – 19 with hyperparameter training would perform a great deal better.

SPECIFICATIONS

We trained the model in the following environments. The technical overview of the environment is briefly explained here.

1. Windows Laptop (Core i7) and MacBook (M1)
    a. Logistic Regression
    b. SVM
2. Google Colab (with GPU environment)
    a. SVM
    b. LDAM-DRW
    c. SL

Our laptops and MacBooks are incapable of running GPU-related tasks which forms the basis for training Deep Learning models. Hence, we used Colab with GPU enabled environment specifically for training the DL models.

Note: The Readme.md file in the code has all the steps to run the models and generate the datasets.

EXPERIMENTS

Logistic Regression

	Original Dataset (Balanced)	Balanced Symmetrical Noisy	Balanced Asymmetrical Noisy	Imbalanced Dataset	Imbalanced Symmetrical Noisy	Imbalanced Asymmetrical Noisy
Accuracy	90.43%	89.84%	87.33%	90.19%	87.99%	87.02%
Precision	90.43%	89.84%	87.33%	90.19%	87.99%	87.02%
Recall	90.43%	89.84%	87.33%	90.19%	87.99%	87.02%
ROC AUC	94.61%	94.27%	92.81%	94.49%	93.26%	92.67%

Support Vector Machine (SVM)

	Original Dataset (Balanced)	Balanced Symmetrical Noisy	Balanced Asymmetrical Noisy	Imbalanced Dataset	Imbalanced Symmetrical Noisy	Imbalanced Asymmetrical Noisy
Accuracy	97.58%	94.07%	88.90%	78.94%	88.13%	84.46%
Precision	97.55%	94.33%	89.19%	89.23%	90.14%	84.92%
Recall	97.56%	94.05%	88.57%	98.27%	97.83%	84.32%
ROC AUC	97.61%	94.27%	88.81%	85.61%	91.57%	84.50%

Symmetric cross-entropy Learning (SL)

	Original Dataset (Balanced)	Balanced Symmetrical Noisy	Balanced Asymmetrical Noisy	Imbalanced Dataset	Imbalanced Symmetrical Noisy	Imbalanced Asymmetrical Noisy
Accuracy	90.16%	97.81%	97.69%	96.57%	94.81%	97.06%
Precision	90.49%	98.74%	97.99%	96.73%	96.86%	97.46%
Recall	89.92%	96.49%	97.35%	96.51%	91.10%	96.81%
ROC AUC	90.61%	97.57%	97.41%	96.34%	92.38%	96.42%

LDAM-DRW

	Original Dataset (Balanced)	Balanced Symmetrical Noisy	Balanced Asymmetrical Noisy	Imbalanced Dataset	Imbalanced Symmetrical Noisy	Imbalanced Asymmetrical Noisy
Accuracy	98.22%	97.32%	97.66%	73.15%	94.14%	95.69%
Precision	98.25%	97.37%	97.75%	85.24%	94.30%	96.01%
Recall	98.21%	97.31%	97.67%	73.13%	94.08%	95.67%
ROC AUC	99.01%	94.27%	98.71%	85.14%	96.73%	97.60%

VGG-19

	Original Dataset (Balanced)	Balanced Symmetrical Noisy	Balanced Asymmetrical Noisy	Imbalanced Dataset	Imbalanced Symmetrical Noisy	Imbalanced Asymmetrical Noisy
Accuracy	98.56%	97.59%	97.89%	97.43%	97.72%	96.47%
Precision	98.93%	97.31%	97.82%	97.98%	97.39%	96.25%
Recall	98.63%	96.51%	96.67%	96.81%	96.99%	96.00%
ROC AUC	99.97%	99.95%	99.91%	99.96%	99.96%	99.89%

CONTRIBUTION

Mahesh Desai

- Symmetric cross-entropy Learning (SL)
- Imabalancing of datasets
- VGG-19
- Support Vector Machine (SVM)

Sagar Sonawane

- Logistic Regression
- Adding Noise to datasets
- Finding Metrics

Shriram Ravi

- LDAM-DRW
- VGG-19
- Support Vector Machine (SVM)

REFERENCES

[1] Code - https://github.com/kaidic/LDAM-DRW.

[2] Code - https://github.com/YisenWang/symmetric_cross_entropy_for_noisy_labels. 2

[3] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, pages 1567–1578, 2019.

[4] Yann LeCun, L´eon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[5] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross-entropy for robust learning with noisy labels. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 322–330, 2019

[6] https://teddykoker.com/2019/06/multi-class-classification-with-logistic-regression-in-python/

[7] https://www.kaggle.com/muerbingsha/mnist-vgg19/notebook

[8] https://github.com/qandeelabbassi/python-svm-sgd/blob/master/svm.py

[9] https://towardsdatascience.com/svm-implementation-from-scratch-python-2db2fc52e5c2