Dealing with Imbalance in Computer Vision

1st Jeffrey Xu
Texas A&M University
TAMU Computer Science
College Station, US
jeffreyxu@tamu.edu

***Abstract*—Data imbalance is a universal problem in the world of machine learning and computer vision. Classes that are represented less in the data may get overlooked during training resulting in poor performance on those classes. Many different methods and models have been created over the years to help against imbalanced data such as sampling methods or new models. In this paper, we explore the world of imbalanced data and its effects on training and different methods that can help improve the performance of models.**

***Index Terms*—Imbalanced Data, Computer Vision, CNN, Logistical Regression, Random Forests**

I. Introduction

Collecting large amounts of data is very important when training models. However, this data is often imbalanced and some classes are represented more than others. This creates some major problems when training machine learning models as the imbalanced dataset may not lead to amazing performance overall across all the different classes.

Finding efficient and simple solutions to imbalanced data could help data scientists across the world better train their models and achieve better performance. Often the data collected from the real world is imbalanced. For example, collecting data about people's favorite food by surveying people over the internet may only give us data from first-world countries as those are usually the countries where a mass-majority of the population has access to the internet. The amount of people from third-world countries will be greatly under-represented and this poses as an issue if data scientists and machine learning engineers want to use this data to train models to predict people's favorite food.

In this paper, we want to explore the effects of imbalanced datasets in Computer Vision problems. We will specifically be using the MNIST digits dataset to test our models and methods on.

Computer Vision is very useful within real-world applications. It has been applied to problems such as autonomous driving, facial recognition, image compression and many other fields of industry. However, problems in computer vision are still prone to the general issues that imbalanced datasets give.

In the *Difference of Methods* section, we talk about the implementation of our models, datasets and any techniques that were used to help counteract the negative effects of the imbalanced datasets. In the *Test Results* section, we go over the overall performance of the models and the techniques used for imbalanced datasets. Within the *Comparison of Techniques* section, we analyze the performance of each model along with the imbalanced dataset techniques used and determine the effectiveness of them. We also analyze the difference of the performance of models with balanced and imbalanced datasets and see how much imbalanced datasets can affect models. Finally, in our *Conclusion* section, we wrap up our discussion on imbalanced datasets and see how the specified techniques in this paper could help machine learning applications in the future.

II. Difference of Methods

The MNIST digit dataset was used for this paper. The data was first loaded into a torch dataset then converted to a torch data loader (the data loader is used during CNN to load the data in batches for training). We also converted the torch dataset tensors into numpy arrays to make the compatibility with the *sklearn* models an easier transition. To do this, we essentially looped through all the given torch datasets and converted each feature array into a flattened numpy array and the labels tensor into a 1D numpy array containing all the labels.

For this paper, we tested three distinct models on our MNIST dataset. The models used were Logistical Regression, Random Forests and Convolution Neural Networks. All three models can be used for classification problems.

Logistical Regression is a classic model used in classification problems. It uses a non-linear logistical function to convert a linear combination of the input data to likelihoods of each respective output class. It also has some regularization to prevent the coefficients of the linear combination to explode and overfit to the training data. This model is very simple but very powerful within the realm of machine learning.

Random Forests are an extension of decision trees. Decision trees split the dataset into partitions and assigns each equivalence class an output class. Decision trees are very interpretable to the human mind and are often perform well with classification problems. However, decision trees can tend to overfit and perform poorly on output classes that are less represented in the training data. This can be solved using post or pre-pruning but pre-pruning often doesn't give the best performance and post-pruning is very computational heavy as it has to perfectly fit to the training data and then begin reducing the model bottom-up. Luckily, the Random Forest classifier solves this issue by creating multiple decision trees and using this list of decision trees to predict. Within each decision tree, only certain features can be used to predict to prevent every single decision tree from looking the exact same as the others. Doing so allows the Random Forest classifier to handle smaller output classes and overall performs much better than simple decision trees.

Convolutional Neural Networks are a specific form of neural networks that are used for computer vision problems. At each convolutional layer, a filter is used to compute output values for that layer. At the end, fully-connected layers are used to compress the output data from the convolutional layers down to likelihoods of each output class which can then be used to predict. This model is the most complex model out of the three but performs very well with image classification and computer vision problems.

The *sklearn* library was used to implement the Logistic models. Cross Validation was used to determine the best regularization parameter and the *lbfgs* solver was used for multi-class classification. Two different models were fitted for the two different provided datasets (one model for the linear dataset and one model for the step dataset).

The *sklearn* library was also used to implement our Random Forest Classifier and also performed cross validation to determine the optimal number of classifiers to use in our Random Forest Classifier model. Once again, we fitted models for both datasets and for each dataset. During cross validation, we cross validated models between the Gini index and the entropy loss function and use the model that had the best performance during cross validation to create our actual model.

Finally, we used *pytorch* to implement our Convolutional Neural network. We used a 1D convolutional layer, followed by two 2D convolutional layers, finally finishing with two fully connected layers that output the likelihoods for each of the ten output classes. We also used a momentum optimizer along with stochastic batch learning for our CNN when performing gradient descent.

Along with implementing our models, we also implemented some additional sampling methods to counteract the imbalanced dataset. Specifically, undersampling was implemented to help balance the dataset and give better results. For our dataset, *imblearn*[1] was used to implement the undersamping. We used the *RandomUnderSampler* to perform all undersampling. Oversampling wasn't performed in this paper since it is very computationally heavy. Note that access to large servers or GPUs were not possible during the time of implementing these models and methods and computationally efficient algorithms had to be used.

III. Test Results

For our models, the *AUCROC* score was decided to determine the performance of the models since it is a generally unbiased form of measurement. We had to implement a multiclass version of the *sklearn* ROC score function. To do this, every single class had an ROC score (all vs. one) and the scores were averaged all classes to produce one final performance value.

IV. Comparison of Techniques

V. Conclusions

References

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.