Teaching AI to Explain its Decisions Can Affect Class Balance

Kevin Volkert

Cognitive Systems Group, University of Bamberg
https://www.uni-bamberg.de/en/cogsys/
kevin_volkert@stud.uni-bamberg.de

Abstract. *TED* has been proposed as a simple framework for using any supervised classification approach to create Explainable AI models [3]. It requires manual annotation of training data with explanations. The authors of TED report that it yields results as good or better than conventional supervised learning. This paper shows that one possible reason for this is that the extension of training data can introduce bias by changing the class distribution. Experiments on the Iris dataset and a synthetic dataset show that TED performs significantly worse than initially reported when using class weights during training to offset class imbalance in training data.

Keywords: Explainable AI · Supervised classification · Machine learning

1 Introduction

Explainable Artificial Intelligence as a field of research is becoming more popular. While black-box approaches like deep learning have been very successful, good performance alone is not always enough. As AI becomes more widely used it also becomes more important to understand what it is actually doing. A decision may be good if it is correct, but it is better if it is correct and understandable.

There are good, old-fashioned AI approaches, like *Inductive Logic Programming* (ILP), which are inherently white-box and thus interpretable. This can, for example, be used to generate human-readable representations of the decision making of a system. *Dare2Del* [6] is an ILP-based companion system which learns rules that are directly interpretable and can be presented to end users as explanations for suggested actions.

LIME [5] is a technique proposed with the goal of adding explainability to arbitrary classifiers. This has become a popular idea (Google Scholar lists over 2,400 citations at the time of this writing). For example, this can be used to gain insight into black-box approaches like deep neural networks.

Another approach has a similar goal, but a very different design: TED (*Teaching Explanations for Decisions*) is a simple, high-level framework . The basic idea is to use any supervised classification approach to learn not only labels, but also explanations. This approach depends heavily on the quality of the training data. The authors explicitly state the importance of good explanations provided by domain experts [3].

Its high-level, semi-manual nature makes this approach very flexible. The explanations are part of the training data, therefore they can be adapted to a variety of domains and target groups. The authors argue that the requirement to manually create training data with explanations can also help reduce bias, since adding a biased example would be hard to justify [3].

However, bias is not always obvious when considering single examples. Class imbalance is one type of bias that can occur in training data. When classes are not equally distributed, trained classifiers may inadvertently favor more frequent classes when making predictions.

This paper investigates this problem in the context of TED. In section 2 the design of the TED framework is described and the original evaluation by [3] is summarized. Section 3 shortly explains how class imbalance can become an issue when using TED. Section 4 presents experiments that make the issue apparent. Finally, section 5 provides a conclusion.

2 Related Work

TED, unlike traditional classification approaches, learns class labels and explanations from examples [3]. Any supervised classification approach can be used with TED. The training data consists of:

X a set of features
Y a class label for X
E an explanation for Y

The form and content of explanations are not specified and may contain arbitrary values. The only constraint is that they must be enumerable.

Because explanations are meant to be provided as part of the training data, the application of TED is extremely simple. Instead of learning only labels from examples ($X \rightarrow Y$), TED learns labels and explanations ($X \rightarrow YE$). The Cartesian product instantiation described in [3] simply encodes labels and explanations as YE before training. Then the trained model predicts YE which is decoded into a separate label and explanation.

The Cartesian product instantiation of TED has been evaluated on two datasets. The *Tic-Tac-Toe* example is based on the game of tic-tac-toe. The legal non-terminal board positions are labeled with the optimal move to make and an explanation based on a small set of rules for playing the game. For any move, the explanation can be (in order of preference):

1. *win* (if the game is won with the move)

2. *block* (if the move prevents the other player from winning)
3. *threat* (if the game could be won with the following move)
4. *empty* (otherwise).

This results in a dataset of 4,520 examples. There are 36 possible combinations of label and explanation, since each rule can apply to 9 different moves.

The *Loan Repayment* example contains 10,000 applications for credit with two labels (good or bad payment performance). Explanations are not included in the original dataset. In [3] explanations were generated by applying a small rule set on the data. Two rules consisting of three literals each were used to classify the data. The labels of examples that were not consistent with these rules were changed. For positive examples, the rule itself was used as the explanation. For negative examples, the explanation was determined by the rule whose first literal matched, combined with which of the other literals that did not match. This resulted in 8 explanations.

Table 1 shows the accuracy in the Tic-Tac-Toe and Loan Repayment examples reported in [3]. The results appear promising, though they are unexpected. The performance for label prediction is reported to be as good or better than that of the equivalent models without the TED framework. In the evaluation of the Tic-Tac-Toe example in [3] the authors point out that, "[g]iven the increase in number of classes, one might expect the accuracy to decrease." They also admit that the accuracy in the Loan Repayment example "may be artificially high due to the data generation method" [3].

Table 1. Accuracy for predicting Y and E in Tic-Tac-Toe and Loan Repayment

Training input	Accuracy (%)			
	Tic-Tac-Toe		Loan Repayment	
	Y	E	Y	E
X, Y	96.5		99.2	
X, Y, E	97.4	94.3	99.6	99.4

3 Potential Problems

Adding explanations to the training data and learning the explanations in addition to the labels results in a larger output space. Each class may have multiple possible explanations. The authors of TED describe this as an implicit 2-level hierarchy in the data that could potentially be exploited [3].

In the Cartesian product instantiation the result is simply a larger number of classes and therefore a smaller number of examples per class compared to the same dataset without added explanations. This means that the underlying classifier in a simple TED instance has to learn more from less. Intuitively there should be a decrease in performance.

The addition of explanations to an existing dataset can change the effective class balance. Even if the original class labels are left untouched, it is possible to have a different number of explanations for each class. In that case the combined YE learned by the classifier will not have the same distribution of classes as the original dataset without explanations. This can introduce or change existing bias in the training data.

It seems plausible that the improved performance which TED shows (in comparison to an equivalent classifier trained without explanations) is at least partially due to this altered class balance. Section 4 attempts to test this idea.

4 Experiments

In this section TED is evaluated on two additional datasets: the well-known Iris flower dataset [2] (included in scikit-learn) and the Proactive Retention dataset (included in the TED source code). The performance of different classifiers is compared in different configurations:

- with/without TED (i. e. with/without explanations)
- with/without balanced class weights for the underlying classifiers

TED is included in *IBM AIX360* [1] which is implemented using *scikit-learn* [4]. scikit-learn provides a parameter *class_weight* which can be used to balance classes. When this parameter is set to "balanced", each sample is given a weight that is inversely proportional to the class frequency in the training data. This prevents the classifier from being biased toward a certain class simply because it has the most examples.

Three classifiers were chosen for these experiments: a Support Vector Machine (SVM) classifier, a Decision Tree classifier, and a *DummyClassifier*. The DummyClassifier is used with the *strategy* parameter set to "stratified". It makes predictions by choosing a label randomly, considering the class distribution of the training data. This makes bias through class imbalance obvious, since the DummyClassifier does not have any other biases.

All experiments in this section were executed using AIX360 version 0.2 and scikit-learn version 0.21.

4.1 Iris Dataset

The Iris flower dataset consists of 150 examples of 3 types of flowers (*iris setosa, iris versicolor, iris viginica*). The 3 classes are equally distributed, i. e. there are 50 examples per class. There are 4 real-valued features for petal/sepal length/width.

Proper manual annotation would require domain knowledge. Since this requirement could not be fulfilled for this paper, a decision tree (standard parameters, no maximum depth) was trained on the full dataset. This overfitted decision tree perfectly predicts every example in the dataset. The leaves (identified by the node number) of this tree were used as explanations (see figure 1).

This results in a total of 9 explanations:

– 1 for *iris setosa*
– 3 for *iris versicolor*
– 5 for *iris virginia*

These are also all possible YE combinations.

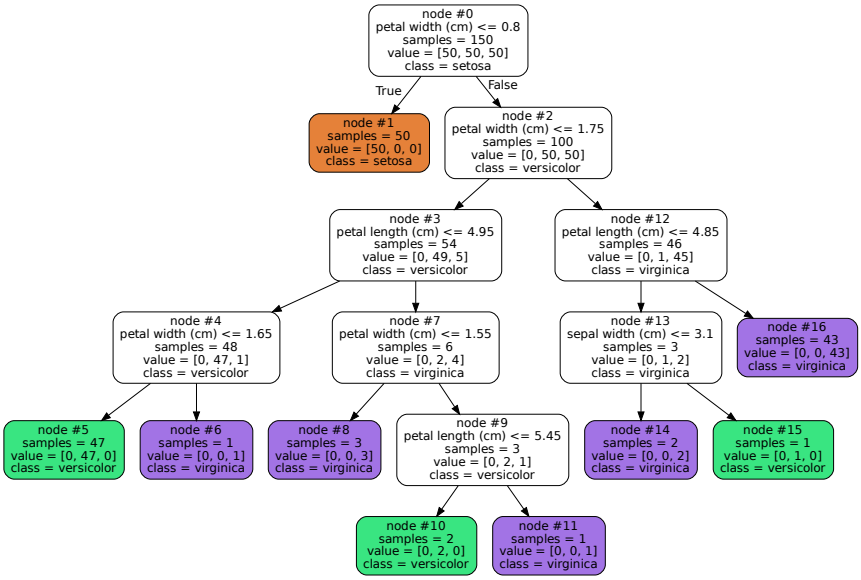

Fig. 1. The overfitted decision tree for the Iris dataset. *value* shows the number of *setosa*, *versicolor*, and *virginica* samples in the sub-tree.

Unlike the labels, the combined labels and explanations are not equally distributed (see table 2).

An SVM classifier (default parameters), a decision tree (maximum depth of 2 to prevent overfitting), and a DummyClassifier were trained using a 33% test data split (i. e. 100 training examples). The results of the classifiers without balanced class weights can be seen in table 3. The accuracy of the SVM and decision tree are consistent with the evaluation in [3]. Performance on label prediction is either as good (SVM) or even better (a 4 percentage point increase for decision tree) when explanations are included in the training data. The prediction accuracy on explanations is also quite high.

The DummyClassifier shows a surprising increase in accuracy. Without explanations, labels are predicted with 34% accuracy, which is to be expected from a classifier that chooses randomly from 3 equally distributed labels. With explanations, however, the performance increases by over 10 percentage points.

Table 2. Label/explanation distribution for the complete annotated Iris dataset

Label	Explanation	Count	Percentage (%)
Iris setosa	1	50	33.33
Iris versicolor	5	47	31.33
	10	2	1.33
	15	1	0.67
Iris virginica	6	1	0.67
	8	3	2.00
	11	1	0.67
	14	2	1.33
	16	43	28.67

Table 3. Accuracy for classifiers on the Iris dataset without class weights

Classifier	Training input	Accuracy (%)	
		Y	E
SVM	X, Y	98	
	X, Y, E	98	92
Decision tree	X, Y	92	
	X, Y, E	96	92
DummyClassifier	X, Y	34	
	X, Y, E	46	40

With balanced class weights, the SVM and decision tree yield noticably different results (see table 4). Accuracy without explanations is the same. With explanations the prediction performance for both labels and explanations decreases significantly. Label accuracy drops by 12–30 percentage points, and the prediction of explanations is only about two-thirds as good.

Table 4. Accuracy for classifiers on the Iris dataset with **balanced** class weights

Classifier	Training input	Accuracy (%)	
		Y	E
SVM	X, Y	98	
	X, Y, E	86	64
Decision tree	X, Y	92	
	X, Y, E	62	60

The Iris dataset is admittedly quite small. The next section repeats the same experiments on a larger, synthetic dataset.

4.2 Proactive Retention Dataset

The TED source code in the IBM AIX360 repository (https://github.com/IBM/AIX360/) includes a synthetic dataset that is used in the *Proactive Retention*

tutorial. The intended use case is finding employees who are at risk of leaving a company.

The data is generated according to distribution functions. There are 8 features:

- Position in the company
- Organization
- Potential
- Rating
- Rating slope
- Salary competitiveness
- Tenure (number of months employed)
- Position tenure (number of months at current position)

Each feature has a non-uniform distribution from which the values are randomly sampled.

There are two labels, *Yes* and *No*, expressing whether proactive retention is necessary for a given employee or not. Explanations are generated by applying 25 rules based on the features. If any of these rules applies to an example, it is labeled *Yes* with the rule number as explanation. If no rule applies, the example is labeled *No*. The dataset in the TED source code contains 10,000 examples of which 33.8% are labeled *Yes*.

The SVM classifier was trained with default parameters. The decision tree was trained with a maximum depth of 5.

The results are similar to the ones in section 4.1 with regard to class balance (see table 5), but more in line with the expectation stated in section 3. Without balanced class weights, the TED-augmented versions of the classifiers perform worse than their counterparts without explanations by about 6 percentage points. The DummyClassifier improves very slightly in label prediction. The accuracy for explanations is at roughly the same level as for labels.

Table 5. Accuracy for classifiers on the Proactive Retention dataset without class weights

Classifier	Training input	Accuracy (%)	
		Y	E
SVM	X, Y	86.15	
	X, Y, E	80.5	77.1
Decision tree	X, Y	91.2	
	X, Y, E	85.55	81.25
DummyClassifier	X, Y	54.3	
	X, Y, E	54.75	43.85

With balanced class weights, however, both the SVM and the decision tree again perform much worse. Label prediction performance decreases by 18–40 percentage points.

Table 6. Accuracy for classifiers on the Proactive Retention dataset with **balanced** class weights

Classifier	Training input	Accuracy (%)	
		Y	E
SVM	X, Y	86.5	
	X, Y, E	68.75	59.9
Decision tree	X, Y	92.25	
	X, Y, E	48.35	37.4

5 Conclusion

This paper performs additional experiments based on the work by [3] and attempts to investigate the unexpected performance gains that arise from making a machine learning problem harder.

The experiments show that class imbalance can be a problem. If training data, especially small datasets, is not carefully created or extended, the distribution of classes can become an issue. The simple Cartesian product instantiation of TED can be very sensitive to training data. Especially when augmenting existing data with explanations to make it compatible with TED, it can be easy to overlook the introduced or changed bias.

Using balanced class weights to counteract the bias in the class distribution of the training data results in a large decrease in performance in both experiments (see section 4). This shows that at least some of the accuracy increases in label prediction that TED achieves compared to equivalent classifiers trained without explanations may be based on the aforementioned bias.

Since manual work on datasets is time-consuming, methods that automatically generate explanations for existing datasets or create synthetic datasets from scratch are very appealing. Such methods have been used in the original work by [3] as well as in this paper. It would, however, be more interesting to see how TED performs on a more realistic, representative dataset.

Acknowledgments

Prof. Dr. Ute Schmid, head of Cognitive Systems Group at University of Bamberg, held a seminar on Explainable AI during the winter semester 2019/2020 out of which this paper emerged.

References

1. Arya, V., Bellamy, R.K.E., Chen, P.Y., Dhurandhar, A., Hind, M., Hoffman, S.C., Houde, S., Liao, Q.V., Luss, R., Mojsilović, A., Mourad, S., Pedemonte, P., Raghavendra, R., Richards, J., Sattigeri, P., Shanmugam, K., Singh, M., Varshney, K.R., Wei, D., Zhang, Y.: One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques. arXiv:1909.03012 [cs, stat] (Sep 2019)

2. Fisher, R.A.: The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics **7**(2), 179–188 (1936). https://doi.org/10.1111/j.1469-1809.1936.tb02137.x
3. Hind, M., Wei, D., Campbell, M., Codella, N.C.F., Dhurandhar, A., Mojsilović, A., Natesan Ramamurthy, K., Varshney, K.R.: TED: Teaching AI to Explain its Decisions. In: Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society. pp. 123–129. ACM (Jan 2019). https://doi.org/10.1145/3306618.3314273
4. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
5. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why Should I Trust You?": Explaining the Predictions of Any Classifier. arXiv:1602.04938 [cs, stat] (Aug 2016)
6. Siebers, M., Schmid, U.: Please delete that! Why should I?: Explaining learned irrelevance classifications of digital objects. KI - Künstliche Intelligenz **33**(1), 35–44 (Mar 2019). https://doi.org/10.1007/s13218-018-0565-5