`libconform v0.1.0`: a Python library for conformal prediction

Jonas Faßbender jonas@fassbender.dev

1. Introduction

This paper introduces the Python library `libconform`, implementing concepts defined in Vovk et al. (2005), namely the conformal prediction framework and Venn prediction for reliable machine learning. These algorithms address a weakness of more traditional machine learning algorithms which produce only bare predictions, without their confidence in them/the probability of the prediction, therefore providing no measure of likelihood, desirable and even necessary in many real-world application domains.

The conformal prediction framework is composed of variations of the conformal prediction algorithm (CP), first described in Vovk et al. (1999); Saunders et al. (1999). A conformal predictor provides a measurement of confidence in its predictions. A Venn predictor, on the other hand, provides a multi-probabilistic measurement, making it a probabilistic predictor. Below in the text, Venn predictors are included if only "conformal prediction framework" is written, except stated otherwise.

The conformal prediction framework is applied successfully in many real-world domains, for example face recognition, medical diagnostic and prognostic and network traffic classification (see Balasubramanian et al., 2014, part 3).

It is build on traditional machine learning algorithms, the so called underlying algorithms (see Papadopoulos et al., 2007), which makes Python the first choice for implementation, since its machine learning libraries are top of the class, still evolving and improving due to the commitment of a great community of developers and researchers.

`libconform`'s aim is to provide an easy to use, but very extensible API for the conformal prediction framework, so developers can use their preferred implementations for the underlying algorithm and can leverage the library, even in this early stage. `libconform v0.1.0` is **not** yet stable; there are still features missing and the API is very likely to change and improve. The library is licensed under the MIT-license and its source code can be downloaded from `https://github.com/jofas/conform`.

This paper combines `libconform`'s documentation with a outline of the implemented algorithms. Paragraphs marked with **i** contain general information about the library and descriptions of the internal workings, while paragraphs marked with **!** describe changes in future versions.

Appendix A provides an overview over `libconform`'s API and Appendix B contains examples on how to use the library.

2. Conformal predictors

Like stated in the introduction, this chapter will only outline conformal prediction (CP). For more details see Vovk et al. (2005).

CP—like the name suggests—determines the label(s) of an incoming observation based on how well it conforms with previous observed examples. Let $\wr z_1, \ldots, z_n \int$ be a bag, also called multiset[1], of examples, where each example $z_i \in \mathbf{Z}$ is a tuple $(x_i, y_i); x_i \in \mathbf{X}, y_i \in \mathbf{Y}$. $\mathbf{X}$ is called the observation space and $\mathbf{Y}$ the label space. For this time $\mathbf{Y}$ is considered finite, making the task of prediction a classification task, rather than regression, which will be considered in chapter 2.2.

A conformal predictor can be defined as a confidence predictor Γ. For this an input $\epsilon \in (0, 1)$, the significance level is needed. $1 - \epsilon$ is called the confidence level. A conformal predictor Γ^ϵ is conservatively valid under the exchangeability assumption, which means, as long as exchangeability holds, it makes errors at a frequency of ϵ or less. For more on that refer to Vovk et al. (2005, chapters 1,2,7).

CP, in its original setting, produces nested prediction sets. Rather than returning a single label as its prediction, it returns a set of elements $\mathbf{Y}' \in 2^{\mathbf{Y}}$, $2^{\mathbf{Y}}$ being the set of all subsets of $\mathbf{Y}$, including the empty set. The prediction sets are called nested, because, for $\epsilon_1 \geq \epsilon_2$, the prediction of Γ^{ϵ_1} is a subset of Γ^{ϵ_2} (see Vovk et al., 2005, chapter 2).

In order to predict the label of a new observation x_{n+1}, set $z_{n+1} := (x_{n+1}, y)$, for each $y \in \mathbf{Y}$ and check how z_{n+1} conforms with the examples of our bag $\wr z_1, \ldots, z_n \int$.

This is done with a nonconformity measure $A_{n+1} : \mathbf{Z}^n \times \mathbf{Z} \to \mathbb{R}$. First, z_{n+1} is added to the bag, then A_{n+1} assigns a numerical score to each example in z_i:

$$\alpha_i = A_{n+1}(\wr z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_{n+1} \int, z_i). \tag{1}$$

One can see in this equation that z_i is removed from the bag. It is also possible to compute α_i with z_i in the bag, which means for $A_{n+1} : \mathbf{Z}^{n+1} \times \mathbf{Z} \to \mathbb{R}$ the score is computed as:

$$\alpha_i = A_{n+1}(\wr z_1, \ldots, z_{n+1} \int, z_i). \tag{2}$$

Which one is preferable is case-dependent (see Shafer and Vovk, 2008, chapter 4.2.2).

α_i is called nonconformity score. The nonconformity score can now be used to compute the p-value for z_{n+1}, which is the fraction of examples from the bag which are at least as nonconforming as z_{n+1}:

$$\frac{|\{i = 1, \ldots, n+1 : \alpha_i \geq \alpha_{n+1}\}|}{n+1}. \tag{3}$$

1. It is typical in machine learning to denote this as the training set, even though examples do not have to be unique, making the so called set a multiset. A multiset is not a list, since the ordering of the elements is not important.

Another way to determine the p-value is through smoothing, in which case the nonconformity scores equal to α_{n+1} are multiplied by a random value τ_{n+1}:

$$\frac{|\{i = 1, \dots, n+1 : \alpha_i > \alpha_{n+1}\}| + \tau_{n+1}|\{i = 1, \dots, n+1 : \alpha_i = \alpha_{n+1}\}|}{n+1} \tag{4}$$

A conformal predictor using the smoothed p-value is called a smoothed conformal predictor and is exactly valid under exchangeability, which means it makes errors at a rate exactly ϵ (see Vovk et al., 2005, chapter 2). If the p-value of z_{n+1} is larger than ϵ, y is added to the prediction set.

Algorithm 1 : Conformal predictor $\Gamma^\epsilon(\lbag z_1, \dots, z_n \rbag, x_{n+1})$

for all $y \in \mathbf{Y}$ **do**
  set $z_{n+1} := (x_{n+1}, y)$ and add it to the bag
  for all $i = 1, \dots, n+1$ **do**
    compute α_i with (1) or (2)
  end for
  set p_y with (3) or (4)
  if $p_y > \epsilon$ **then**
    add p_y to prediction set
  end if
end for
return prediction set

2.1 Nonconformity measures based on underlying algorithms

2.2 Conformal predictor for regression: ridge regression confidence machine

3. Inductive conformal predictors

4. Mondrian (inductive) conformal predictors

5. Probabilistic prediction: Venn predictors

6. Meta-conformal predictors

7. Conclusion

Appendices

A. API reference

B. Examples

References

Vineeth Balasubramanian, Shen-Shyang Ho, and Vladimir Vovk. *Conformal Prediction for Reliable Machine Learning: Theory, Adaptations and Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2014. ISBN 0123985374, 9780123985378.

Harris Papadopoulos, Vladimir Vovk, and Alex Gammerman. Conformal prediction with neural networks. volume 2, pages 388 – 395, 11 2007. ISBN 978-0-7695-3015-4. doi: 10.1109/ICTAI.2007.47.

Craig Saunders, Alex Gammerman, and Vladimir Vovk. Transduction with confidence and credibility. 1999.

Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9:371–421, jun 2008. ISSN 1532-4435.

Vladimir Vovk, Alex Gammerman, and Craig Saunders. Machine-learning applications of algorithmic randomness. 1999.

Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2005. Springer, New York.