Journal of Machine Learning Research 1 (2013) 1-1 Submitted 8/13; Published 10/13

PyStruct - Learning Structured Prediction in Python

Andreas C. Müller amueller@ais.uni-bonn.de
Sven Behnke behnke@cs.uni-bonn.de
Institute of Computer Science, Department VI
University of Bonn
Bonn, Germany

Editor: Mark Reid

Abstract

Structured prediction methods have become a central tool for many machine learning applications. While more and more algorithms are developed, only very few implementations are available.

PyStruct aims at providing a general purpose implementation of standard structured prediction methods, both for practitioners and as a baseline for researchers. It is written in Python and adapts paradigms and types from the scientific Python community for seamless integration with other projects.

Keywords: structured prediction, structural support vector machines, conditional random fields, Python

1. Introduction

In recent years, a wealth of research in methods for learning structured prediction as well as in their application in areas such as natural language processing and computer vision has been performed. Unfortunately, only few implementations are publicly available—many applications are based on the non-free implementation of Joachims et al. (2009).

PyStruct aims at providing a high-quality implementation with an easy-to-use interface, in the high-level Python language. This allows practitioners to efficiently test a range of models, as well as allowing researchers to compare to baseline methods much more easily than this is possible with current implementations. PyStruct is BSD-licensed, allowing modification and redistribution of the code, as well as use in commercial applications. By embracing paradigms established in the scientific Python community and reusing the interface of the widely-used scikit-learn library (Pedregosa et al., 2011), PyStruct can be used in existing projects, replacing standard classifiers. The online documentation and examples help new users understand the somewhat abstract ideas behind structured prediction.

2. Structured Prediction and Casting it into Software

Structured prediction can be defined as making a prediction $f(x)$ by maximizing a compatibility function between an input x and the possible labels y (Nowozin and Lampert, 2011). Most current approaches use linear combinations of feature functions to measure compatibility:

$$f(x) = \arg\max_{y \in \mathcal{Y}} \ \theta^T \Psi(x, y). \tag{1}$$

Here, y is a *structured* label, Ψ is a joint feature function of x and y, and θ are parameters of the model. *Structured* means that y is more complicated than a single output class, for example a label for each word in a sentence or a label for each pixel in an image. Learning structured prediction means learning the parameters θ from training data.
Using the above formulation, learning can be broken down into three sub-problems:

P1: Optimizing the objective with respect to θ.
P2: Encoding the structure of the problem in a joint feature function Ψ.
P3: Solving the maximization problem in Equation 1.

The later two problems are usually tightly coupled, as the maximization in Equation 1 is usually only feasible by exploiting the structure of Ψ, while the first is treated as independent. In fact, when P3 cannot be performed exactly, learning θ strongly depends on the quality of the approximation. However, treating approximate inference and learning as a joint optimization problem is currently out of the scope of the package, and we implement a more modular setup. PyStruct takes an object-oriented approach to decouple the task-dependent implementation of P2 and P3 from the general algorithms used to solve P1.

Estimating θ is done in `learner` classes, which currently support cutting plane algorithms for structural support vector machines (SSVM) (Joachims et al., 2009)), subgradient methods for SSVMs (Ratliff et al., 2007), block-coordinate Frank-Wolfe (BCFW) (Lacoste-Julien et al., 2012), and the structured perceptron and latent variable SSVMs (Yu and Joachims, 2009). The cutting plane implementation uses the cvxopt package (Dahl and Vandenberghe, 2006) for quadratic optimization. Encoding the structure of the problem is done using `model` classes, which compute Ψ and encode the structure of the problem. The structure of Ψ determines the hardness of the maximization in Equation 1 and is a crucial factor in learning. PyStruct implements models (corresponding to particular forms of Ψ) for many common cases, such as multi-class and multi-label classification, conditional random fields with constant or data-dependent pairwise potentials, and several latent variable models. The maximization for finding y in Equation 1 is carried out using external libraries, such as OpenGM (Kappes et al., 2013), LibDAI (Mooij, 2010) and others. This allows the user to choose from a wide range of optimization algorithms, including (loopy) belief propagation, graph-cuts, QPBO, dual subgradient, MPBP, TRWs, LP and many other algorithms. For problems where exact inference is infeasible, PyStruct allows the use of linear programming relaxations, and provides modified loss and feature functions to work with the continuous labels. This approach, which was outlined in Finley and Joachims (2008), allows for principled learning when exact inference is intractable. When using approximate integral solvers, learning may finish prematurely and results in this case depend on the inference scheme and learning algorithm used.

Table 1 lists algorithms and models that are implemented in PyStruct and compares them to other public structured prediction libraries: Dlib (King, 2009), SVM$^{\text{struct}}$ (Joachims et al., 2009), and CRFsuite (Okazaki, 2007). We also report the programming language and the project license.

Package	Language	License	Algorithms				Models			
			CP	SG	BCFW	LV	ML	Chain	Graph	LDCRF
PyStruct	Python	BSD*	✓*	✓	✓	✓	×	✓	✓	✓
SVM^{struct}	C++	non-free	✓	×	×	✓	×	×	×	×
Dlib	C++	boost	✓	×	×	×	×	✓	✓	×
CRFsuite	C++	BSD	×	×	×	×	✓	✓	×	×

CP—cutting plane optimization of SSVM, SG—online subgradient optimization of SSVM, LV— latent variable SSVM, ML—maximum likelihood learning, Chain—chain-structured models with pairwise interactions, Graph—arbitrary graphs with pairwise interactions, LDCRF—latent dynamic CRF (Morency et al., 2007). *PyStruct itself is BSD licensed, but uses the GPL-licensed package cvxopt for cutting-plane learning.

Table 1: Comparison of structured prediction software packages.

3. Usage Example: Semantic Image Segmentation

Conditional random fields are an important tool for semantic image segmentation. We demonstrate how to learn an n-slack support vector machine (Tsochantaridis et al., 2006) on a superpixel-based CRF on the popular Pascal dataset. We use unary potentials generated using TextonBoost from Krähenbühl and Koltun (2012). The superpixels are generated using SLIC (Achanta et al., 2012).[1] Each sample (corresponding on one entry of the list `X`) is represented as a tuple consisting of input features and a graph representation.

```
1 model = crfs.EdgeFeatureGraphCRF(
2     class_weight=inverse_frequency, symmetric_edge_features=[0, 1],
3     antisymmetric_edge_features=[2], inference_method='qpbo')
4
5 ssvm = learners.NSlackSSVM(model, C=0.01, n_jobs=-1)
6 ssvm.fit(X, Y)
```

Listing 1: Example of defining and learning a CRF model.

The source code is shown in Listing 1. Lines 1-3 declare a model using parametric edge potentials for arbitrary graphs. Here, `class_weight` re-weights the hamming loss according to inverse class frequencies. The parametric pairwise interactions have three features: a constant feature, color similarity, and relative vertical position. The first two are declared to be symmetric with respect to the direction of an edge, the last is antisymmetric. The inference method used is QPBO-fusion moves. Line 5 creates a `learner` object that will learn the parameters for the given model using the n-slack cutting plane method, and line 6 performs the actual learning. Using this simple setup, we achieve an accuracy of 30.3 on the validation set following the protocol of Krähenbühl and Koltun (2012), who report 30.2 using a more complex approach. Training the structured model takes approximately 30 minutes using a single Intel Core i7 core.

1. The preprocessed data can be downloaded at `http://www.ais.uni-bonn.de/download/datasets.html`.

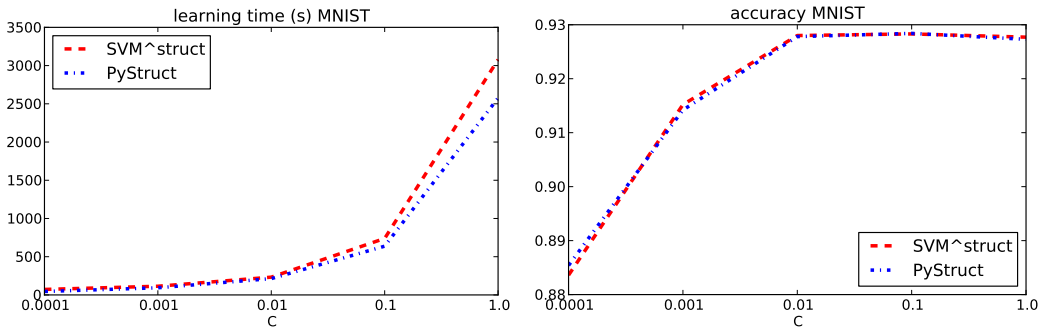

Figure 1: Runtime comparison of PyStruct and SVM$^{\text{struct}}$ for multi-class classification.

4. Experiments

While PyStruct focuses on usability and covers a wide range of applications, it is also important that the implemented learning algorithms run in acceptable time. In this section, we compare our implementation of the 1-slack cutting plane algorithm (Joachims et al., 2009) with the implementation in SVM$^{\text{struct}}$. We compare performance of the Crammer-Singer multi-class SVM with respect to learning time and accuracy on the MNIST dataset of handwritten digits. While multi-class classification is not very interesting from a structured prediction point of view, this problem is well-suited to benchmark the cutting plane solvers with respect to accuracy and speed.

Results are shown in Figure 1. We report learning times and accuracy for varying regularization parameter C. The MNIST dataset has 60 000 training examples, 784 features and 10 classes.[2] The figure indicates that PyStruct has competitive performance, while using a high-level interface in a dynamic programming language.

5. Conclusion

This paper introduced PyStruct, a modular structured learning and prediction library in Python. PyStruct is geared towards ease of use, while providing efficient implementations. PyStruct integrates itself into the scientific Python eco-system, making it easy to use with existing libraries and applications. Currently, PyStruct focuses on max-margin and perceptron-based approaches. In the future, we plan to integrate other paradigms, such as sampling-based learning (Wick et al., 2011), surrogate objectives (for example pseudo-likelihood), and approaches that allow for a better integration of inference and learning (Meshi et al., 2010).

Acknowledgments

The authors would like to thank Vlad Niculae and Forest Gregg for their contributions to PyStruct and Andre Martins for his help in integrating the AD^3 solver with PyStruct. This work was partially funded by the B-IT research school.

References

R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *PAMI*, 2012.

2. Details about the experiment and code for the experiments can be found on the project website.

J. Dahl and L. Vandenberghe. Cvxopt: A python package for convex optimization. 2006.

T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *ICML*, 2008.

T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural SVMs. *JMLR*, 77(1), 2009.

J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, J. Lellmann, N. Komodakis, et al. A comparative study of modern inference techniques for discrete energy minimization problems. In *CVPR*, 2013.

D. E. King. Dlib-ml: A machine learning toolkit. *JMLR*, 10, 2009.

P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. In *NIPS*, 2012.

S. Lacoste-Julien, M. Schmidt, and F. Bach. A simpler approach to obtaining an o (1/t) convergence rate for projected stochastic subgradient descent. *arXiv preprint arXiv:1212.2002*, 2012.

O. Meshi, D. Sontag, T. Jaakkola, and A. Globerson. Learning efficiently with approximate inference via dual losses. In *ICML*, 2010.

J. M. Mooij. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *JMLR*, 2010.

L.-P. Morency, A. Quattoni, and T. Darrell. Latent-dynamic discriminative models for continuous gesture recognition. In *CVPR*, 2007.

S. Nowozin and C. H. Lampert. *Structured learning and prediction in computer vision*. Now publishers Inc, 2011.

N. Okazaki. CRFsuite: a fast implementation of conditional random fields (crfs), 2007. URL `http://www.chokkan.org/software/crfsuite/`.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *JMLR*, 2011.

N. Ratliff, J. A. D. Bagnell, and M. Zinkevich. (Online) Subgradient Methods for Structured Prediction. In *AISTATS*, 2007.

I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, and Y. Singer. Large margin methods for structured and interdependent output variables. *JMLR*, 6(2), 2006.

M. Wick, K. Rohanimanesh, K. Bellare, A. Culotta, and A. McCallum. Samplerank: Training factor graphs with atomic gradients. In *ICML*, 2011.

C.-N. J. Yu and T. Joachims. Learning structural SVMs with latent variables. In *ICML*, 2009.