Journal of Machine Learning Research 1 (2015) 1-?? Submitted 1/15; Published 1/15

Dissolve$^{\text{struct}}$– A Library for Distributed Structured Prediction

Tribhuvanesh Orekondy TOREKOND@STUDENT.ETHZ.CH
ETH Zurich

Martin Jaggi JAGGI@INF.ETHZ.CH
ETH Zurich

Aurelien Lucchi AURELIEN.LUCCHI@INF.ETHZ.CH
ETH Zurich

Editor: t.b.d.

Abstract

This paper describes DISSOLVE$^{\text{struct}}$, a modular and flexible open source software package for distributed training of structured prediction models, such as structured SVMs. Project website: github.com/dalab/dissolve.

We support a broad range of applications, and interfaces to scala, java and python. Our framework is empowered by the fault tolerant SPARK computing platform, and automatically adopts to the existing tradeoffs of computation vs communication cost on real world systems. The proposed distributed algorithm combines the recent communication efficient COCOA scheme (Jaggi et al., 2014; Ma et al., 2015) with the state of the art primal-dual structured prediction solvers (Lacoste-Julien et al., 2013), and improves further by adding some new ideas for caching oracle answers. The framework allows approximate inference, and provides a similar standard interface as SVM$^{\text{struct}}$ for the user.

Keywords: Structured Prediction, Structured SVM, Distributed Training

1. Introduction

Structured prediction has gained a lot of popularity over the past few years due to its ability to process structured objects such as images or text documents.

The structured support vector machine (SSVM) is a particularly successful variant of this approach that can be optimized in various ways, including cutting-plane methods, stochastic gradient descent, or a primal-dual scheme such as (Lacoste-Julien et al., 2013).

The main contribution of our work is to extend the primal-dual COCOA$^+$ framework (Jaggi et al., 2014; Ma et al., 2015) also to the structured SVM case, and combine it with BCFW (Lacoste-Julien et al., 2013) used as a local solver.
Remainder omitted in this sample. See http://www.jmlr.org/papers/ for full paper.

2. Structured SVM formulation

A structured model predicts the labeling $\mathbf{y}$ for a given input $\mathbf{x}$ by maximizing some score function $S_{\boldsymbol{w}} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, i.e.,

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} S_{\boldsymbol{w}}(Y) = \operatorname*{argmax}_{Y \in \mathcal{Y}} \boldsymbol{w}^T \Psi(\mathbf{x}, \mathbf{y}), \tag{1}$$

where $\boldsymbol{w}$ are the model parameters and $\Psi(\mathbf{x}, \mathbf{y})$ is the *feature map* corresponding to the input $\mathbf{x}$ and the labeling $\mathbf{y}$.

Given a set of n training examples $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$ where $\mathbf{x}_i \in \mathcal{X}$ and $\mathbf{y}_i \in \mathcal{Y}$ is the associated labeling, the learning task consists in finding model parameters $\boldsymbol{w}$ that achieve low empirical loss subject to some regularization. In other words, we seek

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}} \sum_{n=1}^{n} \ell(\mathbf{x}_i, \mathbf{y}_i, \boldsymbol{w}) + R(\boldsymbol{w}), \tag{2}$$

where ℓ is the *surrogate loss* function, a quantity that is usually related to and often upper-bounds the training error, and $R(\boldsymbol{w})$ is the regularizer (such as the L2 norm of $\boldsymbol{w}$) that helps prevent overfitting. The most common choice of ℓ is the hinge loss, as used in (Taskar et al., 2003; Tsochantaridis et al., 2005), which is defined as

$$l(\mathbf{y}_i, \mathbf{y}^*_{\boldsymbol{w}}, \boldsymbol{w}) = [S_{\boldsymbol{w}}(\mathbf{y}^*_{\boldsymbol{w}}) + \Delta(\mathbf{y}_i, \mathbf{y}^*_{\boldsymbol{w}}) - S_{\boldsymbol{w}}(\mathbf{y}_i)]_+ \tag{3}$$

The most popular method to solve problems of the form (2) is the stochastic subgradient method (SGD) (Ratliff et al., 2007; Shalev-Shwartz et al., 2010). Although this method has been quite successful it requires tuning parameters to achieve good performance. An experimental comparison of our proposed approach with distributed variants of SGD is provided in Section 5 below.

The Lagrange dual of the above n-slack-formulation (2) has $m := \sum_i |\mathcal{Y}_i|$ variables or potential 'support vectors'. Writing $\alpha_i(\boldsymbol{y})$ for the dual variable associated with the training example i and potential output $\boldsymbol{y} \in \mathcal{Y}_i$, the dual problem is given by

$$\begin{aligned} \min_{\substack{\boldsymbol{\alpha} \in \mathbb{R}^m \\ \boldsymbol{\alpha} \geq 0}} \quad & f(\boldsymbol{\alpha}) := \frac{\lambda}{2} \left\| A\boldsymbol{\alpha} \right\|^2 - \boldsymbol{b}^T \boldsymbol{\alpha} \\ \text{s.t.} \quad & \textstyle\sum_{\boldsymbol{y} \in \mathcal{Y}_i} \alpha_i(\boldsymbol{y}) = 1 \quad \forall i \in [n] \ , \end{aligned} \tag{4}$$

where the matrix $A \in \mathbb{R}^{d \times m}$ consists of the m columns $A := \left\{ \frac{1}{\lambda n} \boldsymbol{\psi}_i(\boldsymbol{y}) \in \mathbb{R}^d \;\middle|\; i \in [n], \boldsymbol{y} \in \mathcal{Y}_i \right\}$, and the vector $\boldsymbol{b} \in \mathbb{R}^m$ is given by $\boldsymbol{b} := \left(\frac{1}{n} L_i(\boldsymbol{y}) \right)_{i \in [n], \boldsymbol{y} \in \mathcal{Y}_i}$. Given a dual variable vector $\boldsymbol{\alpha}$, we can use the Karush-Kuhn-Tucker optimality conditions to obtain the corresponding primal variables $\boldsymbol{w} = A\boldsymbol{\alpha} = \sum_{i, \boldsymbol{y} \in \mathcal{Y}_i} \alpha_i(\boldsymbol{y}) \frac{\boldsymbol{\psi}_i(\boldsymbol{y})}{\lambda n}$, see Appendix **??**. The gradient of f then takes the simple form $\nabla f(\boldsymbol{\alpha}) = \lambda A^T A\boldsymbol{\alpha} - \boldsymbol{b} = \lambda A^T \boldsymbol{w} - \boldsymbol{b}$; its $(i, \boldsymbol{y})$-th component is $-\frac{1}{n} H_i(\boldsymbol{y}; \boldsymbol{w})$, cf. (**??**). Finally, note that the domain $\mathcal{D} \subset \mathbb{R}^m$ of (4) is the product of n probability simplices, $\mathcal{D} := \Delta_{|\mathcal{Y}_1|} \times \ldots \times \Delta_{|\mathcal{Y}_n|}$.

3. Distributed Algorithm

Local Subproblem per Machine Following the idea in (Ma et al., 2015), we can define a data-local subproblem of the original dual optimization problem (4), which can be solved on machine k and only requires accessing (and decoding) data examples which are already available locally, i.e., examples with $i \in \mathcal{P}_k$. More formally, each machine k is assigned the following local subproblem, depending only on the previous shared primal vector $\boldsymbol{w} \in \mathbb{R}^d$, and the change in the local dual variables α_i with $i \in \mathcal{P}_k$:

$$\max_{\Delta \boldsymbol{\alpha}_{[k]} \in \mathbb{R}^n} \ \mathcal{G}_k^{\sigma'}(\Delta \boldsymbol{\alpha}_{[k]}; \boldsymbol{w}, \boldsymbol{\alpha}_{[k]}) \tag{5}$$

where

$$\mathcal{G}_k^{\sigma'}(\Delta\boldsymbol{\alpha}_{[k]}; \boldsymbol{w}, \boldsymbol{\alpha}_{[k]}) := -\frac{1}{n}\sum_{i\in\mathcal{P}_k} \ell_i^*(-\alpha_i - (\Delta\boldsymbol{\alpha}_{[k]})_i)$$
$$-\frac{1}{K}\frac{\lambda}{2}\|\boldsymbol{w}\|^2 - \frac{1}{n}\boldsymbol{w}^T A\Delta\boldsymbol{\alpha}_{[k]} - \frac{\lambda}{2}\sigma'\left\|\frac{1}{\lambda n}A\Delta\boldsymbol{\alpha}_{[k]}\right\|^2 \tag{6}$$

note that ℓ_i^* here is a simplex-constraint.

write a few words here that this is really an extension of the primal-dual structure assumed in the CoCoA papers. those have just allowed ℓ_i^*'s depending on a single dual variable. here we have simplex blocks of variables, one per ℓ_i^*. one of the newer SDCA papers also has this generalized duality structure

Algorithm: Two variants essentially, see discussion in (Ma et al., 2015)

- CoCoA Averaging, $\gamma := \frac{1}{K}$, using the safe subproblem parameter $\sigma' := 1$
- CoCoA$^+$ Adding, $\gamma := 1$, using the safe subproblem parameter$\sigma' := K$

Algorithm 1 CoCoA$^+$ Framework

Input: Datapoints A distributed according to partition $\{\mathcal{P}_k\}_{k=1}^K$. Aggregation parameter $\gamma \in (0, 1]$, subproblem parameter σ' for the local subproblems $\mathcal{G}_k^{\sigma'}(\Delta\boldsymbol{\alpha}_{[k]}; \boldsymbol{w}, \boldsymbol{\alpha}_{[k]})$ for each $k \in [K]$.
Starting point $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$, $\boldsymbol{w}^{(0)} := \mathbf{0} \in \mathbb{R}^d$.
for $t = 0, 1, 2, \ldots$ **do**
  for $k \in \{1, 2, \ldots, K\}$ **in parallel over computers do**
    call the local solver, computing a Θ-approximate solution $\Delta\boldsymbol{\alpha}_{[k]}$ of the local subproblem (6)
    update $\boldsymbol{\alpha}_{[k]}^{(t+1)} := \boldsymbol{\alpha}_{[k]}^{(t)} + \gamma\,\Delta\boldsymbol{\alpha}_{[k]}$
    return $\Delta\boldsymbol{w}_k := \frac{1}{\lambda n}A\Delta\boldsymbol{\alpha}_{[k]}$
  end for
  reduce $\boldsymbol{w}^{(t+1)} := \boldsymbol{w}^{(t)} + \gamma\sum_{k=1}^K \Delta\boldsymbol{w}_k$. (7)
end for

4. Software package

The use of the DissolveStruct requires users to implement the following functions:

- The joint feature map $\Psi(X, Y)$ which encodes the input/output pairs.
- The structured loss function $\Delta(Y_i, Y)$.
- A maximization oracle which computes the most violating constraint by solving Eq. X.
- A prediction function that computes X. This operation is usually performed with the maximization oracle.

TODO: Can we show a simple example like the pystruct paper?

Table 1: Comparison of structured prediction software packages

Package	**Language**	**License**	**Distributed**	**Models**		
				ML	Chain	Graph
Dissolve $^{\text{struct}}$	Scala	?	✓	✗	✓	✓
PyStruct	Python	BSD	✓	✗	✓	✓
SVM$^{\text{struct}}$	C++	non-free	✓	✗	✗	✗
Dlib	C++	boost	✓	✗	✓	✓
CRFsuite	C++	BSD	✓	✓	✓	✗

5. Experiments

Run experiments on one or two standard structured prediction tasks.

Experiment 1: scaling number of machines, on same dataset.

Experiment 2: cocoa+-bcfw vs standard distributed mini-batch SGD

(Compare to other frameworks... ? most are just single machine. could include to show more insights on our single machine performance as well)

Acknowledgments

We would like to thank Jan Deriu, Bettina Messmer, Thijs Vogels and Ruben Wolff for contributing to the code and discussions to improve readability.

Appendix A.

In this appendix we prove the following theorem from Section 6.2:
Theorem *Let u, v, w be discrete variables such that v, w do not co-occur with u (i.e., $u \neq 0 \;\Rightarrow\; v = w = 0$ in a given dataset $\mathcal{D}$). Let N_{v0}, N_{w0} be the number of data points for which $v = 0, w = 0$ respectively, and let I_{uv}, I_{uw} be the respective empirical mutual information values based on the sample $\mathcal{D}$. Then*

$$N_{v0} \;>\; N_{w0} \;\;\Rightarrow\;\; I_{uv} \;\leq\; I_{uw}$$

with equality only if u is identically 0. ■
Proof. We use the notation:

$$P_v(i) \;=\; \frac{N_v^i}{N}, \quad i \neq 0; \quad P_{v0} \;\equiv\; P_v(0) \;=\; 1 - \sum_{i \neq 0} P_v(i).$$

These values represent the (empirical) probabilities of v taking value $i \neq 0$ and 0 respectively. Entropies will be denoted by H. We aim to show that $\frac{\partial I_{uv}}{\partial P_{v0}} < 0$....

Remainder omitted in this sample. See http://www.jmlr.org/papers/ for full paper.

References

Martin Jaggi, Virginia Smith, Martin Takáč, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-Efficient Distributed Dual Coordinate Ascent. In *NIPS 2014 - Advances in Neural Information Processing Systems 27*, pages 3068–3076, 2014.

Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-Coordinate Frank-Wolfe Optimization for Structural SVMs. In *ICML 2013 - Proceedings of the 30th International Conference on Machine Learning*, 2013.

Chenxin Ma, Virginia Smith, Martin Jaggi, Michael I Jordan, Peter Richtárik, and Martin Takáč. Adding vs. Averaging in Distributed Primal-Dual Optimization. In *ICML 2015 - Proceedings of the 32th International Conference on Machine Learning*, pages 1973–1982, 2015.

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

Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal Estimated Sub-Gradient Solver for SVM. *Mathematical Programming*, 127(1):3–30, October 2010.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-Margin Markov Networks. In *NIPS 2014 - Advances in Neural Information Processing Systems 27*, 2003.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large Margin Methods for Structured and Interdependent Output Variables. *The Journal of Machine Learning Research*, 6, December 2005.