Scalability of Struct SVM approaches

Frederic Boileau Elyes Lamouchi

William St-Arnaud

https://github.com/SoliElvis/structuredPredictionProject

1st May 2019

Abstract

.

Contents

1 Introduction **3**

2 Preliminaries **5**

2.1 Structured SVM N-slack formulation 5

3 Extra Gradient Algorithm **6**

3.1 Saddle Point Formualtion and Intuition 6

3.2 Duality and Gap function 8

3.2.1 Non-Euclidean setting 9

3.2.2 Memory-efficient tweak 9

4 Frank Wolfe and Variants **10**

4.1 Frank-Wolfe, the conditional gradient algorithm 10

4.2 Block Coordinate Frank Wolfe 11

5 Empirical **12**

5.1 Experiments 12

5.1.1 Results 13

6 Discussion **15**

7 Conclusion and Further Work **16**

8 Citations and References **17**

Appendices **18**

1 Introduction

Structured prediction in machine learning is tasked with learning predictors where the labels on the datapoints are more than simple tags but have inherent structure which implies the following:

- The number of potential labels for a given feature vector can grow exponentially with the input which makes traditional classification procedures intractable
- A certain intelligibility of the structure; hence a hope to leverage it to improve tractability

It is often quite hard to know in advance wether we can tackle a structured prediction problem with known approaches.

In this paper we discuss two different approaches to solving structured prediction problems, both focusing on a *large-margin approach* which translates into a support-vector machine - like problem formulation.

Let us introduce some notation which we will use throughout the paper. Define the dataset to be

$$S = \{x^{(i)}, y^{(i)})\}_{i=1}^{n} \in (\mathcal{X} \times \mathcal{Y})^n \tag{1}$$

If one were to frame the machine learning goal in the concisest and simplest way possible we could say that the goal is to *learn* a (parametrized) *predictor* function:

$$\begin{aligned} h_w \quad &: \quad \mathcal{X} \to \mathcal{Y} \\ &: \quad \hat{x} \mapsto y \qquad y \in \mathcal{Y}(\hat{x}) \end{aligned}$$

where $\hat{x}$ is just some arbitrary sampled $\hat{x}$ and we abuse notation and mean the set valued mapping which outputs the feasible label set for a given x by $\mathcal{Y}(x)$

The traditional probabilistic approach is to compute h or its parameter by calculating the most likely parameter given conditionned on the observed data. However this is more often than not untractable in the structured prediction context as summed up in Taskar, Lacoste-Julien, and Jordan (2006)

In a large margin approach we wish to compute the predictor function the following way :

$$h_w(x) = \arg\max_{y \in \mathcal{Y}} \langle w, \phi(x, y) \rangle \tag{2}$$

where ϕ is just the feature map for the dataset.

This is a constrained optimization problem and the structure of $\mathcal{Y}$ clearly has a big influence on how well we can solve the problem as well as which method should work well.

The dual extragradient approach presented in Taskar, Lacoste-Julien, and Jordan (ibid.) leverages a saddle-point approach to the problem to tackle the problem with a first order method and moreover "this approach allows us to exploit the structure of W and Z separately, allowing for efficient solutions for a wider range of parameterizations and structures."(ibid.)

Despite its improvements at the time the dual extragradient approach suffers from two main computational draw-backs:

1. It is not seperable/'stochasizable' which is a severe problem as machine learning has seen much of its success in recent years by optimizing through first-order sampled method, i.e. first order method which use a probabilistic approximation of the gradient through sampling instead of computing the actual gradient (which might be unfeasible).
2. The algorithm requires a projection step which can be expensive depending on the structure of $\mathcal{Y}$ and often involves a quadratic problem which doesn't scale too well.

This leads us to consider the Block Seperable Frank Wolfe algorithm presented in Lacoste-Julien et al. (2013)

2 Preliminaries

2.1 Structured SVM N-slack formulation

Let us recall the basic goal; to construct an accurate linear classifier[1]

$$h_w(x) = \arg\max_{y \in \mathcal{Y}} \langle w, \phi(x, y) \rangle \tag{3}$$

In general finding the optimal seperating hyperplane is an ill-defined problem. In the support vector machine (SVM) setting we want to find the seperating hyperplane with the *largest margin*. The support vectors are the datapoints which lie on the the defined margins. It is not obvious that we should strive to completely seperate the sets so we can include some slack variables. Indeed, some set of points might not be linearly separable but adding slack variables might allow one to misclassify some point but get a higher proportion of correctly classified points.

We call the following way to pose problem the n-slack formulation [2] of the problem :

$$\min_{\omega, \xi} \quad \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n} \varepsilon_i \qquad s.t. \quad \langle w, \psi_i(y) \rangle \geq L(y_i, y) - \varepsilon_i, \quad \forall i, \forall y \in \mathcal{Y}(x) = \mathcal{Y}_i \tag{4}$$

Define $\psi(y) \triangleq \phi(x^{(i)}, y^{(i)}) - \phi(x^{(i)}, y)$ and $L_i(y) \triangleq L(y^{(i)}, y)$

Then the following problem is called the *loss-augmented decoding problem.*

$$H_i = \max_{y \in \mathcal{Y}_i} \{L_i(y) - \langle w, \psi_i(y) \rangle\} \tag{5}$$

We assume it can be solved efficiently; usually through ILPs that have a natural convex relaxation with integral optimal solutions which will be discussed later. Note that the structured hinge-loss is a convex upper bound to the task loss; hence it is quite natural to see that learning w amounts to the unconstrained problem,

$$\min_{w} \quad \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n} H_i(w) \tag{6}$$

This last formulation into a non-smooth unconstrained problem is the one we will focus throughout this paper. Indeed it allows us to leverage convex analysis tools which emphasize replacing constraint sets with appropriate additions to the objective value (which often results in non-smooth problems)

[1] clearly we can easily extend to non-linear cases, with kernel maps for example

[2] see Moguerza and Muñoz (2006) for example

3 Extra Gradient Algorithm

3.1 Saddle Point Formualtion and Intuition

Let us focus on the loss-augmented formulation

$$\min_{w} \quad \frac{\lambda.}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} H_i(w) \tag{7}$$

which is an unconstrained non-smooth (strongly) convex problem as it is the sum of a strongly convex function, and the average of the H_i s which are convex by construction. What is important to note is that the *objective function is defined through solutions of another smaller optimization (maximization) problem.* Clearly it would be naive to just solve one after the other in batch. There are multiple approaches to this problem. A decade or so ago one formulation one could find was a small QP formulation where the loss augmented decoding is dualized to obtain a more standard quadratic problem which can be plugged into commercial solvers (Taskar, Lacoste-Julien, and Jordan, 2006).

In Taskar, Lacoste-Julien, and Jordan (ibid.) it is proposed to instead use first-order algorithms designed specifically for (continuous) saddle point problems in order to leverage the intrinsic min max structure. Intuitively one could say that the harder the inference/decoding problem is, the closer we are to a saddle-point problem, whereas if the decoding problem is trivial we are back to straight minimization hence standard classification.

This enables us to consider a richer class of problems to tackle, especially for problems where dualizing yields a quadratic program (QP) that is not necessarily efficiently solvable (even though it scales linearly ibid.).

It is not hard to see that the objective function of the non-smooth formulation of struct-svm is equivalent to the following:

$$\min_{w\in W}\max_{z\in Z}\sum_{i}\left(w^T F_i z_i + c_i^T z_i - w^T\phi_i(y_i)\right) \tag{8}$$

where the z_i's correspond to the relaxation of the binary labels y_i and satisfy the constraints of the structured problem through their restriction to some feasible set Z. The terms F_i correspond to the matrix with columns $\phi(x_i, y)$ over labels yi. The c_i's correspond to the costs of a z_i and can be identified with the loss l for a label y_i'.

In equation 8, the term that is optimized is defined as:

$$L(w, z) \triangleq \sum_{i} w^T F_i z_i + c_i^T - w^T\phi_i(y_i) \tag{9}$$

The latter function is bilinear in both w and z. Once can summon the idea of a duel between two masters of a zero-sum game who are playing agasint one another. The players, named w and z respectively (i.e. the parameters and the labels) play a zero-sum game. They perform updates using gradients of the objective w.r.t. their parameters. They then project the result to the set of feasible points given by the constraints imposed on the structure. The most common projections are obviously euclidean ones which are well studied and have fast implementations. However some problems call for other types of projections, i.e. based on other distance functions, or other divergence functions which are a weaker notion. The most important class for us are Bregman divergences which are Bregman divergences or distances which are *based on strictly convex functions.*

We have the following operator that is used to perform the updates for both players at the same time.

$$\begin{pmatrix} \nabla_w L(w,z) \\ -\nabla_{z_1} L(w,z) \\ \vdots \\ -\nabla_{z_m} L(w,z) \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & F_1 & \dots & F_m \\ -F_1^T & & & \\ \vdots & & 0 & \\ -F_m^T & & & \end{pmatrix}}_{F} \underbrace{\begin{pmatrix} w \\ z_1 \\ \vdots \\ z_m \end{pmatrix}}_{u} - \underbrace{\begin{pmatrix} \sum_i \phi_i(y_i) \\ c_1 \\ \vdots \\ c_m \end{pmatrix}}_{a} = Fu - a \tag{10}$$

TODO averaging to prevent oscillations and $eta < ||F||^2$

3.2 Duality and Gap function

We can measure the "goodness" of the parameters using the gap function G:

$$G(w,z) \triangleq \left[\max_{z' \in Z} L(w,z') - L^*\right] + \left[L^* - \min_{w' \in W} L(w',z)\right] \tag{11}$$

where L^* gives the result of the min-max of the objective L. When we have a non-optimal point (i.e. not a saddle point), the gap is strictly positive. At at an optimal point, the gap is exaclty equal to 0. Now the restricted gap is exactly the same but the min and max are computed over a set of parameters that are within a certain distance of the start point $(\hat{u}_w, \hat{u}_z) \in U$:

$$G_{D_w,D_z}(w,z) = \max_{z' \in Z} \left[L(w',z') : d(z,z') \leq D_z\right] - \left[\min_{w' \in W} L(w',z) : d(w,w') \leq D_{w'}\right] \tag{12}$$

The motivation for using this restricted gap function is that if we start "close" to an optimal point, of course we will converge more rapidly to it. This can be seen in the convergence analysis of the method.

The dual extragradient algorithm from Nesterov gives a convergence guarantee for the objective L.

This algorithm has a lookahead step (i.e. v) that serves the peform the actual gradient update u^t. The intuition behind the lookahead step is that given a function to optimize that is Lipschitz, Nesterov was able to show that we can upper bound $f_D(\bar{u}^n) = \max_y \{\langle g(y), \bar{u}^n - y\rangle : d(\hat{u}, y) \leq D\}$, where $\bar{u}^n$ is the weighted average over all the updates u^t up to iteration n. The function g corresponds to the objective L in our setting. When value of $f_D(\bar{u}^n)$ gets close to 0, we have that the value $g(y^*)$ for an optimal y^* is close to 0, which signifies that we have reached saddle point (i.e. what we wanted). Note that in the definition of f_D, we used a distance metric d. This corresponds to the Euclidean distance (or Bregman distance in non-Euclidean setting). The rojection operator Π_U in the algorithm simply projects a point back to the set U by finding the nearest point with respect to the distance metric used.

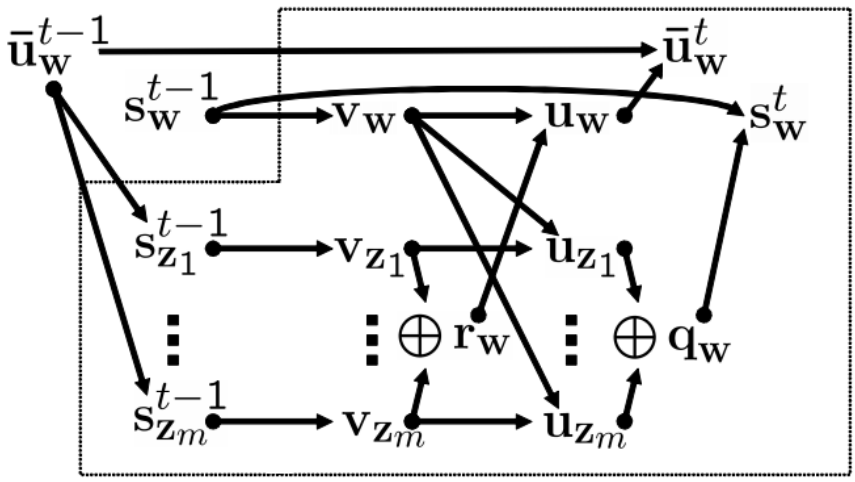

Figure 1: Memory efficient algorithm

3.2.1 Non-Euclidean setting

The main problem with the Euclidean projection operator is that for many problems, it is hard to compute the projection. Indeed for min-cut, we need to compute the partition function first, which is #P-complete. Thus, the authors of the paper introduced the Bregman operator, which computes the projection using the Bregmand divergence. Using this operator has the great advantage of being easier to compute. We can see this for $L1$ regularization. Computing a projection using $L1$ distance is hard since it is not differentiable. Using the negative entropy as our function h (see 8), we get that the Bregman divergence is the KL divergence. This implies that we can differentiate the divergence to get the parameter that minimizes it. It is worth mentioning that for some problems, projections can still be hard to compute. This is why it may be wise to move to the FW algorithm especially if solving a linear problem over constraints the constraints is easy (or "easier").

3.2.2 Memory-efficient tweak

In the dual extragradient algorithm, both a vector s^t and a vector $\bar{u}^t$ are maintained. However, we can observe that the s_t's can be found using the running average $\bar{u^t}$ since $s^t = -(t+1)\sum_{i=0}^{t}(F\bar{u}^t - a)$. We only have to store the vector $\bar{u}^t$. We can even do better when $|Z| \gg |W|$ since $\bar{u}^t = \{\bar{u}_w^t, \bar{u}_z^t\}$ and we only care about the part that corresponds to w. $\bar{u}_z^t$ is maintained implicitly by storing a vector of size $|W|$ (although we now need to store s_w^t). It can be reconstructed using $\bar{u}_w^t$.

4 Frank Wolfe and Variants

4.1 Frank-Wolfe, the conditional gradient algorithm

Throughout this section we consider the problem of minimizing a a continusouly differentiable convex function - say f - over some convex set and compact set $\mathcal{M}$.

Definition 1. *A linear minimization oracle over a set $\mathcal{M}$ is a map defined the following way:*

$$LMO_{\mathcal{M}}(d) \triangleq \arg\min_{s \in \mathcal{M}} \langle d, s \rangle \tag{13}$$

Intuitively the classical FW is a descent method where starting from some arbitrary feasible point we repeatdly take convex combinations with outputs of the oracle $s_t = LMO_{\mathcal{M}}(\nabla f(x_t))$. A reasonable step size can be chosen easily without having to do a line search as we are parametrizing the update by the convex combination "weight" (denoted γ below – as per convention.) Under convexity assumptions on and the function and the feasible set we can tune this parameter between zero and one and be guaranteed to output a feasible iterate. This is a key feature of the algorithm as many other require to compute the gradient's lipschitz constant and/or the strong-convexity parameter, usually denoted μ. All in all parametrizing iterations through convex combinations is a strikingly simple and elegant to the feasibility problem. Moreover we also get sparsity for free, which is key in the application of its block-seperable version to the lagrangian dual of the loss augmented struct-svm.

Let $\alpha \in \mathcal{M}$

for $k = 0$ **to** K **do**

Compute $s = argmin_{s \in \mathcal{M}} \langle s, \nabla f(\alpha^k) \rangle$

Let $\gamma = \frac{2}{k+2}$ (simple version) or optimize for line-search

Update $\alpha^{k+1} = (1 - \gamma)\alpha^k + \gamma s$

end for

Algorithm 1: Classical Frank-Wolf

Sparsity is obviously a key feature for preventing the memory required to explode with the number of training samples. Moreover projection and proximal operations can usually involve solving a quadratic problem which is clearly harder to keep tractable. Many variants and extensions of the basic classical FW algorithm but Jaggi lists the main ones and the active research areas at the time of writing his paper in 2014. we mainly focus on the Block-Seperable variant applied to struct-svm problems as presented in Lacoste-Julien et al. (2013)

While we can't necessarily compute the actual duality gap a linear surrogate can be used and has become widely standard in the literature, see for example Jaggi's review (Jaggi, 2013), it is defined as follows:

$$g(x) \triangleq \max_{s \in \mathcal{D}} \langle x - s, \nabla f(x) \rangle \tag{14}$$

By convexity we immediately have that $g(x) \geq f(x) - f(x^*)$ making exactly validating the tolerance of a solutionn very easy to compute.

4.2 Block Coordinate Frank Wolfe

Due to the exponential number of dual variables in the structured SVM setting, classical algorithms like projected gradient are intractable. Stochastic subgradient methods, on the other hand, achieve a sublinear convergence rate while only requiring a single call to the maximization oracle every step. They are nonetheless very sensitive to the sequence of stepsizes and it is unclear when to terminate the iterations.

In fact many algorithms with good theoretical garantees introduce a dependency between the stepsize to be chosen and some constants which caracterize the function, e.g. the lipschitz constant of the gradient (L) and/or the strong-convexity constant (μ) in order to provide some convergence garantees. For example Nesterov (2003) in his analysis of dual extragradient the step size is bounded above by the gradient's Lipschitz constant which happens to correspond to the norm of the F matrix as defined in the previous sextion. Projections and proximal operators can involve quadratic programs but the natural convex combination approach ensures feasibility at no cost. Since the dual objective function for structured-svm is a quadratic problem we can compute analytically exactly the optimal line-search step size. The BCFW algorithm presented in Lacoste-Julien et al. (2013) also shows a couple equivalencies, the most important being that solving the loss-augmented problem is equivalent to a call to the linear minimization oracle in the dual.

Essentially BCFW can be seen as generalizing some subgradient and cutting plane methods. Tractability in the end, in the current literature, depends on how the inference problem (usually cast as an ILP since we are mostly dealing we underlying structures of combinatorial nature) and the task loss interact. It is shown in Taskar, Lacoste-Julien, and Jordan, 2006 that in many interesting cases such as when the weighted hamming distance is used as task loss for bipartite matching; the inference augmented loss inherits tractability and we have an LP with optimal integral solutions. It is a well known fact in combinatorial optimization that the incidence matrix of a bipartite graph is awlays totally unimodular which is a sufficient condition for optimal integral solution in its convex relaxation LP. The loss-augmented problems defined by the interaction of task loss and its convexification through the hinge loss "operator" are thus related in a very natural way to diverse combinatorial problems and the BCFW provides a theoretical pivot between those two problems.

5 Empirical

5.1 Experiments

In this section, we describe the implementation we performed for this project. The goal was to see how the Dual Extragradient algrithm compared to the Block-coordinate Frank-Wolfe algorithm. The task on which we performed the evaluation was word alignment in machine translation. We extracted the dataset from the Europarl dataset *Europarl Parallel Corpus* 2019. The data consisted of approximately 2 million sentence pairs in both english and french. Each the sentences in each pair were translations of one another. We extracted all sentences and performed a clean by splitting longer sentences into shorter ones. The goal of this step was to reduce the eventual number of matchings in training, which could take long to solve using a LP solver. Of course, each sentence was tokenized beforehand.

We then proceeded to implement the Dual Extragradient and BCFW algorithms using a SVM. We had to define a feature mapping for an input sentence pair. The features were extracted using the fastText library **fastText**. This library included a model that was previously trained to learn embeddings of words in both english and french. We later combined the embeddings in the two languages by applying a transformation found in Chojnacki and Kłopotek (2010). This transformation consisted in applying a matrix to each vector in each language (matrix W for english and Q for french). These matrices are in fact orthogonal (i.e. $Q^T W = I$). The idea behind such a transformation is that we sort of “put" or “align" both languages in the same vector space, a sort of “middle ground". This way we can better compare the words “cat" and “chat" by getting their cosine similarity measure. We combined the cosine measure of each pair of words in the alignment by summing. The following blog post *Aligning Vector Representations* (2017) provides a good intuition using maps that are aligned. As an example, consider the following two sentences:

- This assignment was hard
- Ce travail etait ardu

We would compute the cosine distance for each word tuple (e.g. this/ce, this/travail, . . . , hard/ardu). We were then able to get the highest match of each english word for for french translation. This was how we extracted the “labels". As the dataset was not annotated with alignments, we had to compute those according to the procedure mentioned.

For the features, we used the concatenated embeddings of each word pairs in the alignment. This gave us our edge score. Then, we simply performed a weighted combination of these vectors using the edge labels as weights. To clarify what we mean by edge labels, suppose that the edge linking “assignment” and “travail” has a value of 1. Then, we weight the vector extracted from these two words with 1. As another example, if the edge between “ce” and “ardu” has a value of 0 (i.e. no link between the words), we do not include the vector computed from the statistics of this word pair. This is exactly what was done in Taskar Taskar, Lacoste-Julien, and Jordan, n.d. modulo some other features.

In the implementation of BCFW, we used the solver from scipy with the simplex method. Since the constraint matrix given by the optimization of H_i in the algorithm is unimodular, when we relax the LP, we still get a solution to the ILP without relaxation. Thus, we take advantage of this fact and indeed use the LP solver. The loss that we used was the L_1 distance between the two labels, the proposal and the ground truth.

5.1.1 Results

Since running the experiments was computationally intensive and we did not dispose of a lot of computing power, we had to restrict the training set to a number a relatively small number of sentence pairs (100). This sanity check was simply a hint to the general applicability of the method as we scale up the number of training examples. We used the default λ value of 0.01 as our regularizer since, we did not want to train for too many iterations before getting decent results as per Theorem 3 found in Lacoste-Julien et al. (2013). We were able to get the following results for the BCFW algorithm:

To further motivate our intuition and to make sure the algorith ran properly, we used the local scene dataset Müller and Behnke, 2014 from PyStruct to train a SVM using BCFW. It consisted of vectors of size 294 for the features of each training example (images). Each image was labeld according to the type of objects that were present in the scene.
These were:

$$beach \cdot sunset \cdot fall \cdot foilage \cdot field \cdot mountain \cdot urban \tag{15}$$

These 6 classes were not mutually exclusive so we had a total of $2^6 = 64$ possible labels. Training using all possible edges allowed to fully capture the complex relationships between them. The duality gap on the training data is given by the figure 2.

For the dual extragradient, we were only able to run the algorithm on a dataset of images Vemulapalli and Agarwala (2018). These were tuples of 3 images and a person hand-picked two images in each 3-tuple that resembled each other the most. Using the extragradient algorithm, we were able to get convergence but the results were not satisfying. We only obtained 46% accuracy on the prediction, which would compare with 33% if we had a random predictor (3 choose 2 gives 3 possibilites hence 33%). Thus, it motivated our change of dataset to obtain better results. We also moved on to the BCFW algorithm as we had not implemented it yet.

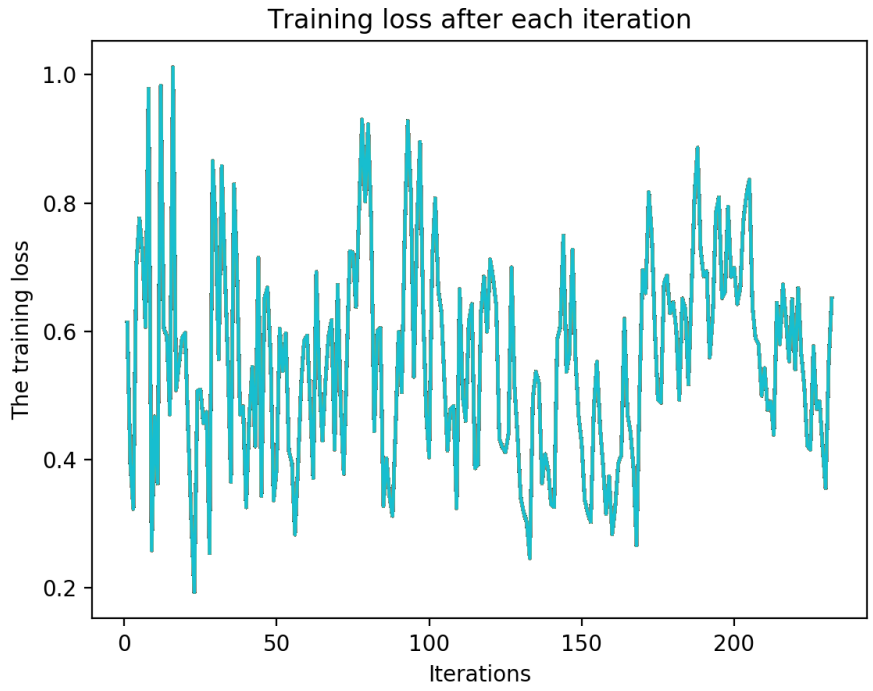

Figure 2: BCFW loss

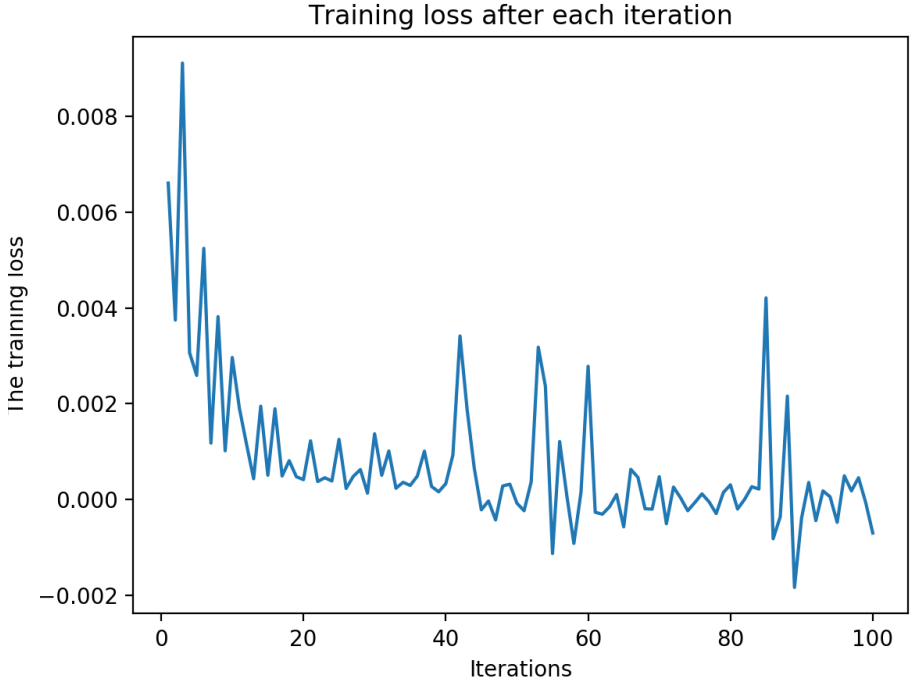

Figure 3: pystruct scene

6 Discussion

In this paper we have discussed mainly two methods which were first proposed in the papers by Taskar, Lacoste-Julien, and Jordan (2006) and Lacoste-Julien et al. (2013). Both of these papers contain insights that go beyond technicalities but try to cast problems in some new way and leverage strong guarantees with what seem to be, at the end of the day, quite strinkingly simple solutions.

The essence of BCFW lies in the simplicity and elegance of convex structures but also shows how structure can be modular, in the abstraction of the LMO for computational costs and its equivalence to max oracles, through functional dependencies it contributes to a principled approach in optimization which is now more and more emphasized. The dual extra gradient algorithm presented in Taskar, Lacoste-Julien, and Jordan, 2006 casts some of the most common max margin estimation problems for structured output as a bilinear saddle point which opened the door for first order already known techniques and using different generalizations of the concept of projection could provide a new angle to attack those problems. All those approaches have in common that they leverage structures through specific black-boxable oracles to assemble algorithms with strong guarantees or tractability.

The numerical experiments provided with this paper are clearly not reliable benchmarks. However going through the process of trying to reproduce results underscored the danger of neglecting a framework for consistent, solid, and accurate benchmarks. We found ourselves biasing towards solutions that are easier to preprocess and load into memory and this might be a harder bias to estimate than the common statistical one.

7 Conclusion and Further Work

8 Citations and References

References

[1] *Aligning Vector Representations*. May 27, 2017. URL: https://www.samtalksml.net/aligning-vector-representations/ (visited on 04/29/2019).

[2] Szymon Chojnacki and Mieczysław Kłopotek. "Random Graph Generator for Bipartite Networks Modeling". In: (Oct. 28, 2010). arXiv: 1010.5943 [physics]. URL: http://arxiv.org/abs/1010.5943 (visited on 04/29/2019).

[3] *Europarl Parallel Corpus*. URL: http://www.statmt.org/europarl/ (visited on 04/29/2019).

[4] Martin Jaggi. "Frank-Wolfe Optimization Algorithms". In: (2014), p. 33.

[5] Martin Jaggi. "Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization". In: (2013), p. 12.

[6] Simon Lacoste-Julien et al. "Block-Coordinate Frank-Wolfe Optimization for Structural SVMs". In: (2013), p. 31.

[7] Javier M. Moguerza and Alberto Muñoz. "Support Vector Machines with Applications". In: *Statist. Sci.* 21.3 (Aug. 2006), pp. 322–336. ISSN: 0883-4237. DOI: 10.1214/088342306000000493. arXiv: math/0612817. URL: http://arxiv.org/abs/math/0612817 (visited on 04/29/2019).

[8] Andreas C. Müller and Sven Behnke. "Pystruct - Learning Structured Prediction in Python". In: *Journal of Machine Learning Research* 15 (2014), pp. 2055–2060. URL: http://jmlr.org/papers/v15/mueller14a.html.

[9] Ben Taskar, Simon Lacoste-Julien, and Michael I Jordan. "Structured Prediction via the Extragradient Method". In: (), p. 12.

[10] Ben Taskar, Simon Lacoste-Julien, and Michael I Jordan. "Structured Prediction, Dual Extragradient and Bregman Projections". In: (2006), p. 27.

[11] Raviteja Vemulapalli and Aseem Agarwala. "A Compact Embedding for Facial Expression Similarity". In: (Nov. 27, 2018). URL: https://arxiv.org/abs/1811.11283v2 (visited on 04/30/2019).

Appendices

Proximal step operator

We define the proximal step operator as follows:

$$T_\eta(\mathbf{u}, \mathbf{s}) = \max_{\mathbf{u} \in U} \left\{ \langle \mathbf{s}, \mathbf{u}' - \mathbf{u} \rangle - \frac{1}{\eta} d(\mathbf{u}, \mathbf{u}') \leq D \right\} \tag{16}$$

The operator is useful to compute projections since when we have a strongly convex function $h(\mathbf{u})$, we can find its convex conjugate $h^*(\mathbf{u}) = \max_{\mathbf{u} \in U} [\langle \mathbf{s}, \mathbf{u} \rangle - h(\mathbf{u})]$. From the definition of a strongly convex function, we have that:

$$h(\mathbf{u}') \geq h(\mathbf{u}) + \langle \nabla h(\mathbf{u}), \mathbf{u}' - \mathbf{u} \rangle + \frac{\sigma}{2} \|\mathbf{u}' - \mathbf{u}\|^2 \tag{17}$$

where σ is the strong convexity parameter. Rearranging, we can define an upper bound on the squared norm of $\mathbf{u}' - \mathbf{u}$. This comes out as:

$$d(\mathbf{u}', \mathbf{u}) \triangleq h(\mathbf{u}') - h(\mathbf{u}) - \langle \nabla h(\mathbf{u}), \mathbf{u}' - \mathbf{u} \rangle \geq \frac{\sigma}{2} \|\mathbf{u}' - \mathbf{u}\|^2 \tag{18}$$

The distance metric d is called the Bregman divergence. The link between the Bregman diverence and the proximal step operator is that if we are given the function h inside the definition of the proximal step update, this induces the Bregman divergence, which in turn induces the update that is performed at each iteration of the extragradient algorithm. For example, if we have $h(\mathbf{u}) = \frac{1}{2}\|\mathbf{u}\|_2^2$, the Bregman divergence becomes $d(\mathbf{u}', \mathbf{u}) = \frac{1}{2}\|\mathbf{u}' - \mathbf{u}'\|_2^2$. We might wonder why we care about the Bregman divergence when the definition still includes the usual norm. After all, we still optimize the term $\langle \mathbf{s}, \mathbf{u}' - \mathbf{u} \rangle - \frac{1}{\eta} d(\mathbf{u}', \mathbf{u})$. This is because $h*$ is differentiable at every point of its domain by the strong convexity of h. Thus, it is easy to compute a projection in the usual fashion: we can compute the derivative of the termn inside the projection operator and set it to 0. It is impossible to do for matchings for example as the distance is not even differentiable. We provide the steps to compute a projection:

$$\mathbf{s} - \nabla_{\mathbf{u}'} d(\mathbf{u}', \mathbf{u}) = \mathbf{s} - \frac{1}{\eta} \nabla_{\mathbf{u}'} d(\mathbf{u}, \mathbf{u}') = \mathbf{s} - \frac{1}{\eta} \left[\nabla h(\mathbf{u}') - \nabla h(\mathbf{u}) \right] \tag{19}$$

By setting this equation to 0, it is possible to recover the optimal $\mathbf{y}'$ when, let's say, $h(\mathbf{u}) = \frac{1}{2}\|\mathbf{u}'\|^2$.

ExtraGradient

We present a simple formulation of the algorithm:

Initialize: Choose $\hat{u} \in U$, set $s^{-1} = 0$.

for $t = 0$ to $t = \tau$ **do**

$v = \mathbf{\Pi}_U(\hat{u} + \eta s^{t-1})$

$u^t = \mathbf{\Pi}_U(v - \eta(Fv - a))$

$s^t = s^{t-1} - (Fu^t - a)$

end for

return $\overline{u^\tau} = \frac{1}{1+\tau} \sum_{t=0}^{\tau} u^t$

Algorithm 2: Dual ExtraGradient

Convergence analysis

The restricted gap function $\mathcal{G}_{D_v, D_z}$ is upper bounded by:

$$\mathcal{G}_{D_w, D_z}(\overline{w^\tau}, \overline{z^\tau}) \leq \frac{(D_w + D_z)\, L}{\tau + 1} \tag{20}$$

$$\mathcal{G}_{D_w, D_z}(w, z) = \begin{pmatrix} \sum_i F_i z_i^* \\ -F_1^T w^* \\ \vdots \\ -F_m^T w^* \end{pmatrix}^T \begin{pmatrix} w - w^* \\ z_1 - z_1^* \\ \vdots \\ z_m - z_m^* \end{pmatrix} - \begin{pmatrix} \sum_i f_i(y_i) \\ c_1 \\ \vdots \\ c_m \end{pmatrix}^T \begin{pmatrix} w - w^* \\ z_1 - z_1^* \\ \vdots \\ z_m - z_m^* \end{pmatrix}$$

We have the righ tools for convergence of the algorithm. The Lipschitz constant of the operator $Fu-$ is given by L is equal to $\max_{u \in \mathcal{U}} \|F(u - u')\|_2 / \|u - u'\|_2 \leq \|F\|_2$. Of course, a point w, z that satisifies $\|w\|_2 \leq D_w$ and $\|z\|_2 \leq D_z$ also satisifies $\|(w, z)\|_2 \leq D$ when $D = \sqrt{D_w^2 + D_z^2}$ since $(w, 0) \perp (0, z)$. It is then easy to see that $f_D \geq \mathcal{G}_{D_w, D_z}$. Thus, the function $\mathcal{G}_{D_w, D_z}$ is upper bounded by the right-hand side of equation 20.

Frank-Wolfe

Let $w^0 = w_i^0 = \overline{w}^0 = 0$, $l^0 = l_i^0 = 0$

for $k = 0...K$ **do**

Pick i at random in $\{1, ..., n\}$

Solve $y_i^* = \underset{y_i \in \mathcal{Y}_i}{max} \quad H_i(y, w^k)$

Let $w_s = \frac{1}{n\lambda}\psi_i(y_i^*)$, and $l_s = \frac{1}{n}L_i(y_i^*)$

Let $\gamma = \frac{\lambda(w_i^k - w_s)^T w^k - l_i^k + l_s}{\lambda||w_i^k - w_s||^2}$, and clip to $[0, 1]$

Update $w_i^{k+1} = (1-\gamma)w_i^k + \gamma w_s$, and $l_i^{k+1} = (1-\gamma)l_i^k + \gamma l_s$

Update $w^{k+1} = w^k + w_i^{k+1} - w_i^k$, and $l_i^{k+1} = (1-\gamma)l_i^k + \gamma l_s$

end for

Algorithm 3: Block-Coordinate Frank-Wolfe – Lacoste-Julien et al. (2013)