On Correcting Inputs: Inverse Optimization for Online Structured Prediction [⋆]

Hal Daumé III, Samir Khuller, Manish Purohit, and Gregory Sanders

University of Maryland, College Park, MD 20742, USA
{hal, samir, manishp, gsanders}@cs.umd.edu

Abstract. Algorithm designers typically assume that the input data is correct, and then proceed to find "optimal" or "sub-optimal" solutions using this input data. However this assumption of correct data does not always hold in practice, especially in the context of online learning systems where the objective is to learn appropriate feature weights given some training samples. Such scenarios necessitate the study of inverse optimization problems where one is given an input instance as well as a desired output and the task is to adjust the input data so that the given output is indeed optimal. Motivated by learning structured prediction models, in this paper we consider inverse optimization with a margin, i.e., we require the given output to be better than all other feasible outputs by a desired margin. We consider such inverse optimization problems for maximum weight matroid basis, matroid intersection, perfect matchings, minimum cost maximum flows, and shortest paths and derive the first known results for such problems with a non-zero margin. The effectiveness of these algorithmic approaches to online learning for structured prediction is also discussed.

1 Introduction

Algorithm designers generally assume that the input data is sacrosanct and correct. Algorithms are then typically run on this input data to compute "optimal" or "sub-optimal" solutions quickly whether it be the computation of a maximum spanning tree, a maximum matching, max weight arborescence, or shortest paths. However, with an increasing reliance on automatic methods to collect data, as well as in systems that learn, this assumption does not always hold. The input data can be erroneous (even though it may be approximately correct), and it becomes important to "adjust" the input data to achieve certain desired conditions.

A simple example can be used to illustrate the main point – suppose we are given a weighted graph $G = (V, E)$ and a spanning tree T, and told that T *should* be a maximum weight spanning tree in G. The goal now is to perturb the edge weights of the graph G, minimizing the L_2 norm of the perturbation, so that T is indeed the optimal spanning tree. This kind of problem has been

[⋆] Partially supported by NSF grant IIS-1451430.

studied previously in the form of “Inverse Optimization” problems. However, we wish to accomplish a stronger goal of making sure that the given tree T is better than *every* other tree in G by a given margin δ.

Our initial motivation for studying this problem comes from the *structured prediction* task in machine learning [1, 2, 3, 4, 5]. For concreteness and ease of exposition, we now describe structured prediction in the context of predicting dependency parse trees for natural language sentences. Given an English sentence, its dependency parse is a rooted, directed tree that indicates the dependencies between different words in the sentence as shown in Figure 4. The input sentence can be represented as a complete, directed graph on the words of the sentence that is parameterized by *features* on the edges. Given a learned model (represented as a vector of parameters), the weight of an edge is computed as the inner product of its feature vector and the model. As linguistic constraints dictate that the required dependency parse must form a rooted, spanning arborescence of the graph, one can use off-the-shelf combinatorial algorithms [6, 7] to find the highest weight arborescence. The learning problem is thus to find a parameter vector such that once the edges are weighted by the inner products, running a combinatorial optimization algorithm would return the desired parse tree. At “training time”, we are given a sentence as well as its correct parse tree and the problem that we need to solve is exactly the inverse optimization problem - given the current model and the parse tree, say T, find the minimum perturbation to the model so that the combinatorial optimization algorithm would return T. It is well established in the learning theory literature that achieving a large margin solution enables better generalization [8]. We consider minimizing the L_2 norm because of connections to prior work [9]. In particular, for applications in structured prediction, the convergence and error bounds (included in Section 6) require L_2 norm minimization.

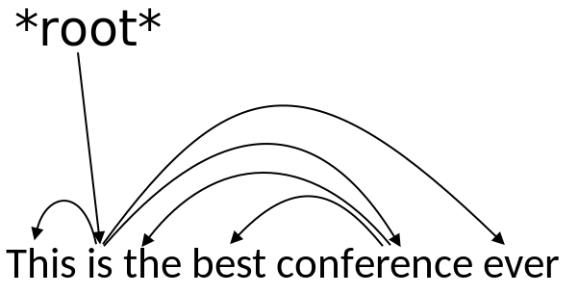

Fig. 1. Example dependency parse tree. The tree describes the relations between head words and their dependents in the sentence

In our work we consider such inverse optimization problems with a margin in a general matroid setting. We consider both the problem of modifying the weights of the elements of a matroid, so that a given basis is a maximum weight basis (with a given margin of δ) and the considerably harder problem of matroid intersection where a given basis of two matroids should have weight higher (by at least δ) than any other set of elements that is a basis in the two matroids. This framework captures two special cases which are useful for structured prediction - namely maximum weight bipartite matching (useful for language translation) and maximum weight arborescence (useful for sentence parsing). We also consider δ-margin inverse optimization problems for a number of other classical combinatorial optimzation problems such as perfect matchings, minimum cost flows and shortest path trees. In addition, we present a generic framework for online learning for structured prediction using the corresponding inverse optimization problem as a subroutine and present convergence and error bounds on this framework.

1.1 Related Work

Inverse optimization problems have been widely studied in the Operations Research literature. Most prior work however has focused on minimizing the L_1 or L_∞ norms between the weight vectors and, more importantly, do not allow non-zero margin (δ). Heuberger [10] provides an excellent survey of the diverse inverse optimization problems that have been tackled. Both the inverse matroid optimization [11] and matroid intersection [12] have previously been studied in the setting of minimizing the L_1 norm and with zero margin. However, they use techniques that are specialized to minimizing the L_1 norm of the perturbation and do not extend to minimizing the L_2 norm. At the same time, these approaches to do not generalize to the general case of inverse optimization with non-zero margins.

In typical global models for structured prediction (for e.g. see [1, 13, 5, 3, 14, 15]), the discrete optimization problem is considered a “black box”. By treating the combinatorial problem as a black box, these methods lose the ability to precisely reason about how certain changes to the underlying parameter vector can affect the eventual output. The simplest approach to solving the online structured prediction problem is the structured perceptron [3]. On each example, the structured perceptron makes a prediction based on its current model. If this prediction is incorrect, the algorithm suffers unit loss and updates its parameters with a simple linear update that moves the predictor closer to the truth and further from the current best guess. While empirically successful in a number of problems, this particular update is relatively imprecise: there are typically an exponential number of possible outputs for any given input, and simply promoting the correct one and demoting the models’ current prediction may do very little to move the model as far as it needs to go. An alternative approach is the large margin discriminative approach [8] that seeks to change the parameters as little as possible subject to the constraint that the true output has a higher score than all incorrect outputs. However, such an approach is often computationally infeasible for structured prediction as there are usually an exponential number of potential outputs. McDonald et al. [15] circumvent this infeasibility by using a k-best list of possible outputs and restrict the set of constraints to require that the true output has a higher score than the incorrect outputs on the k-best list. This has been shown to be effective for small values of k on simple parsing tasks [15]. However, for more complex tasks, like machine translation, one needs more complicated update frameworks [16]. In this work we show that the large margin discriminative approach is applicable to a wide range of problems in structured prediction using techniques from inverse combinatorial optimization.

In the context of online prediction, the most related work to ours is that of Taskar et al. [4], who also consider structured prediction using inverse bipartite matchings. They define a loss function that measures, against a ground truth matching, the number of mispredicted edges in the found matching. This "Hamming distance" style loss function nicely decomposes over the structure of the graph and thereby admits an efficient "loss augmented" inference solution, in which correct edges are penalized during learning. (The idea is that if correct edges are penalized, but the model still produces the correct matching, then it has done so with a sufficiently large margin.) This idea only works in the case of decomposable loss functions, or the simpler 0-margin formulation. In comparison, our approach works both for decomposable loss functions as well as "zero/one loss" over the entire structure. Furthermore, our approach generalizes to arbitrary matroid intersection problems and minimum cost flows and thus is applicable to a much wider range of structured prediction problems.

1.2 Contribution and Techniques

A lot of prior work in the inverse optimization literature formulates the problem as a linear program and then uses strong duality conditions to find the new perturbed weights. However, such techniques cannot be extended to handle a non-zero margin that is required by the application to structured prediction. We formulate inverse optimization to minimize the L_2 norm of the perturbations as a quadratic program and use problem specific optimality conditions to determine a concise set of linear constraints that are both necessary and sufficient to guarantee the required margin. In particular, one of the key ingredients is a set of polynomially many linear constraints that ensure that an appropriately defined auxiliary graph does not contain small directed cycles. We note that our formulations can easily be adapted to minimize the L_1 norm of the perturbations by simply modifying the objective and using linear programming.

We obtain concise formulations for exactly solving δ-margin inverse optimization problems for (i) maximum weight matroid basis, (ii) maximum weight basis in the intersection of two matroids, (iii) shortest s-t path, (iv) shortest path tree, (v) minimum cost maximum flow in a directed graph.

We also present convergence results for the generic online learning framework for structured prediction motivating our study.

The rest of the paper is organized as follows. In Section 2, we formally define δ-margin inverse optimization. In Sections 3 and 4, we present our results on inverse optimization for matroids, and matroid intersections respectively. In Sections 5, Appendix A, and Appendix B, we discuss inverse optimization for perfect matchings in bipartite graphs, minimum cost flows, and shortest path trees. In Section 6, we describe an online learning framework for structured prediction as an application and the proof of convergence and error bounds for this learning framework are presented in Appendix C. Experimental results for our learning model are presented in Appendix D showing significant improvement over previous techniques.

2 Problem Description

As explained in the introduction, we require a given solution to be better than all other feasible solutions by a margin of δ. We now formalize this notion of δ-optimality.

Definition 1 (δ-Optimality). *For a maximization problem P, let $\mathcal{F}$ denote the set of feasible solutions, let w be the weight vector, $c(w, A)$ denote the cost of feasible solution A under weights w, and let $\delta \geq 0$ be a scalar. A feasible solution $S \in \mathcal{F}$ is called δ-optimal under weights w if and only if*

$$c(w, S) \geq c(w, S') + \delta, \quad \forall S' (\neq S) \in \mathcal{F}.$$

δ-optimality for minimization problems is defined similarly. All problems we consider in this work can be classified as δ-margin inverse optimization.

Definition 2 (δ-Margin Inverse Optimization). *For a given optimization problem P, let $\mathcal{F}$ denote the set of feasible solutions, let w be the weight vector, let $\delta \geq 0$ be a scalar, and let $S \in \mathcal{F}$ be a given feasible solution. δ-Margin Inverse optimization is to find a new weight vector w' minimizing $||w' - w||_2$ (L_2 norm) such that S is the δ-optimal solution of P under weights w'.*

In the following sections we consider δ-margin inverse optimization for a number of problems mentioned earlier.

3 Maximum weight matroid basis

In order to provide intuition about the type of problems we propose to solve in this paper, we first begin with the simple case of Inverse Matroid Optimization. We recall the definition of a matroid.

Definition 3 (Matroid). *A matroid is a pair $\mathbb{M} = (X, \mathcal{I})$ where X is a ground set of elements and $\mathcal{I}$ is a family of subsets of X (called Independent sets) such that - (i) $\mathcal{I} \neq \phi$ (ii) (Hereditary) If $B \in \mathcal{I}$, and $A \subseteq B$, then $A \in \mathcal{I}$. (iii) (Exchange property) If $A, B \in \mathcal{I}$, and $|A| < |B|$, then there exists some element $e \in B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$.*

Definition 4 (Matroid Basis and Circuit). *Let $\mathbb{M} = (X, \mathcal{I})$ be a matroid. Then any maximal independent set in $\mathcal{I}$ is called a basis of the matroid. Conversely, any minimal dependent set is called a circuit.*

For the inverse problem we are given a matroid $\mathbb{M} = (X, \mathcal{I})$, a weight function w on the elements, and a basis B of $\mathbb{M}$. The goal is to find a weight function w' so that B is the δ-optimal basis of $\mathbb{M}$ under the new weights. As it is well known that a spanning tree is a basis of a graphical matroid, this inverse matroid optimization problem directly generalizes the inverse maximum spanning tree problem.

We first state a simple optimality condition for a given basis B of a matroid $\mathbb{M}$. An easy generalization of [17] for $\delta \geq 0$ gives the following lemma.

Lemma 1. *A given basis B of a matroid $\mathbb{M}$ is δ-optimal (under weight function w) if and only if for any $f \notin B$, and each $e \in C_B(f)$, $w(e) - w(f) \geq \delta$, where $C_B(f)$ denotes the unique circuit in $B \cup \{f\}$.*

We thus have a set of polynomially many linear constraints that are necessary and sufficient for the given basis B to be δ-optimal. The inverse matroid optimization problem can then be formulated as a linearly constrained quadratic problem as follows -

$$\min_{w'} \sum_{e \in X} (w'(e) - w(e))^2 \quad \textbf{subj. to:} \tag{1}$$

$$w'(e) - w'(f) \geq \delta, \quad \forall f \notin B, \forall e \in C_B(f) \tag{2}$$

Such a program with a quadratic objective and linear constraints can be solved in polynomial time and a number of practical solvers such as [18] are available.

4 Matroid Intersection

Similar to the case with a single matroid, we need to derive a necessary and sufficient condition for a common basis B of two matroids to be δ-optimal. We can establish such an optimality condition with the help of an exchange graph associated with the basis B and matroids $\mathbb{M}_1$ and $\mathbb{M}_2$.

Definition 5 (Exchange Graph). *Given two matroids $\mathbb{M}_1 = (X, \mathcal{I}_1)$ and $\mathbb{M}_2 = (X, \mathcal{I}_2)$, a weight function $w : X \to \mathbb{R}^+$, and a common basis B, an* exchange *graph is a directed, bipartite graph $G = (V, A)$ with a length function l on edges that is defined as follows.*

$$V = B \cup X \setminus B \tag{3}$$

$$A = A_1 \cup A_2 \tag{4}$$

$$A_1 = \{(x, y) | x \in B, y \in X \setminus B, B - \{x\} + \{y\} \in \mathcal{I}_1\} \tag{5}$$

$$A_2 = \{(y, x) | x \in B, y \in X \setminus B, B - \{x\} + \{y\} \in \mathcal{I}_2\} \tag{6}$$

$$l(s) = \begin{cases} w(x) & \text{if } s = (x, y) \in A_1 \\ -w(y) & \text{if } s = (y, x) \in A_2 \end{cases} \tag{7}$$

The above graph captures the exchange operations that can be performed. An edge (e, f) implies that deleting e and adding f to B preserves independence w.r.t matroid $\mathbb{M}_1$ and similarly for the other direction. As the graph is bipartite, every cycle is of even length - a cycle $C = (x_1, y_1, x_2, y_2, \ldots x_k, y_k, x_1)$ corresponds to constructing a set $B'' = B - \{x_1, x_2, \ldots x_k\} \cup \{y_1, y_2, \ldots, y_k\}$. Further

$$w(B'') = w(B) - \sum_{i=1}^{k} w(x_i) + \sum_{i=1}^{k} w(y_i) = w(B) - l(C)$$

where $l(C) = \sum_{e \in C} l(e)$ is the sum of lengths of edges in the cycle C. We are now in a position to present the δ-optimality condition of B in terms of the exchange graph. Fujishige [19] shows the following lemma for the case of $\delta = 0$. We include the extended proof for general δ margin here for completeness. It is important to note that while there are other optimality conditions for matroid intersection such as the weight decomposition theorem by Frank [20], these conditions do not easily generalize for non-zero δ.

Lemma 2 (Matroid Intersection δ-optimality condition). *The given common basis B is δ-optimal if and only if the exchange graph G contains no directed cycle C such that $\sum_{e \in C} l(e) \leq \delta$.*

Proof. We'll refer to two well-known lemmas [17] regarding the relationship between bases of a matroid and matchings in the exchange graph. Let $G_1 = (V, A_1)$ and $G_2 = (V, A_2)$ be the subgraphs of G induced by the two matroids respectively. Further for $B' \subset X$, let $G(B, B')$ denote the subgraph induced on the G by the vertex sets $B \setminus B'$ and $B' \setminus B$.

Lemma 3. *If B' is a basis of matroid $\mathbb{M}_1$ $[\mathbb{M}_2]$, then $G_1(B, B')[G_2(B, B')]$ contains a perfect matching.* $\square$

Lemma 4. *For $B' \subseteq X$, if $G_1(B, B')[G_2(B, B')]$ has a* unique *perfect matching, then B' is a basis of $\mathbb{M}_1$ $[\mathbb{M}_2]$.* $\square$

Sufficiency: This is the easy direction. Let B' be any common basis other than B. Applying Lemma 3, we know that $G(B, B')$ has two perfect matchings (one each in $G_1(B, B')$ and $G_2(B, B')$). Union of these two perfect matchings yields a collection of cycles $\mathcal{C}$. Further, by construction, by traversing these cycles, one can transform $B \to B'$ and hence, we have $w(B') = w(B) - \sum_{C \in \mathcal{C}} l(C)$. Therefore, since we have $l(C) > \delta$ for all cycles, we are guaranteed that $w(B') < w(B) - \delta$ as desired.

Necessity: Ideally, we would like to say that every cycle in G leads to a swapping such that the set so obtained is also independent in both the matroids. This would immediately imply that a cycle of small length would lead to a common basis B' which is not much smaller than B.

However, the presence of a cycle simply implies the presence of a perfect matching (one in each direction) which may not be *unique*. For example, Figure 2 shows an instance of an arborescence problem (left), and the associated exchange

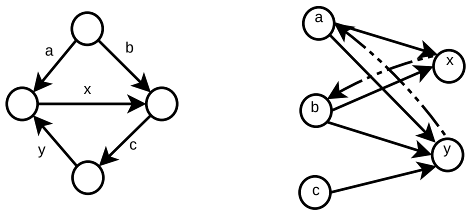

Fig. 2. Instance showing every cycle in G need not lead to a common basis

graph (right). Here G contains a cycle a-x-b-y-a which leads to a new set x, y, c which is not an arborescence.

In the previous example, observe that if the cycle a-x-b-y-a were to have small weight, that would imply that at least one of a-y-a or b-x-b cycles too has small weight both of which lead to a feasible solution. This observation motivates us to look at the *smallest* cycle of weight less than δ and hope that it does induce an *unique* perfect matching.

Suppose that the graph has a cycle having weight less than δ. Let C be the smallest (in terms of number of arcs) such cycle. Look at the graph induced by the vertex set of the cycle. We claim that this induced subgraph has a unique perfect matching (one in each direction). Here we prove the claim for one direction. C being an even cycle trivially contains a perfect matching M from B-side to $X \setminus B$-side. Suppose there exists another perfect matching M'. For every edge (x, y) in $M' \setminus M$, the edge along with the path between y and x in C cause a cycle. Further, each such cycle is smaller (number of edges) than C.

Let $\bar{M}$ denote the matching M with edge directions reversed. The union of M' and $\bar{M}$ now forms a collection of cycles. Consider any such cycle D. WLOG let the cycle be $(x_0, y_0, x_1, y_1, \ldots, x_k, y_k, x_0)$ such that the (x_{i+1}, y_i) are edges in M (i.e. $(y_i, x_{i+1}) \in \bar{M}$) and $(x_i, y_i) \in M'$. [All arithmetic is modulo $k + 1$]. We'll now be interested in the length of the path between these vertices in the original cycle C. Let C_i denote the cycle formed by the edge (x_i, y_i) and the path between y_i and x_i in C. We have,

$$l(C_i) = l(C) - l(\text{Path from } x_i \text{ to } y_i \text{ in } C) + l((x_i, y_i))$$

Since $(x_i, y_{i-1}) \in M$,

$$l(\text{Path from } x_i \text{ to } y_i \text{ in } C) = l((x_i, y_{i-1})) + l(\text{Path from } y_{i-1} \text{ to } y_i \text{ in } C)$$

Further since by construction $l((x, y_i)) = l((x, y_j))(= \pm w(x))$, we have

$$l(C_i) = l(C) - l(\text{Path from } y_{i-1} \text{ to } y_i \text{ in } C)$$

Let $P_{i-1 \to i}$ denote this path. Summing over all (x_i, y_i) edges in D, we get

$$\begin{aligned}\sum_{i=0}^{k} l(C_i) &= kl(C) - (l(P_{k \to 0}) + l(P_{0 \to 1}) + \ldots + l(P_{k-1 \to k})) \\ &= kl(C) - k'l(C)\end{aligned}$$

$\uparrow$ Since we start from y_k, go around the C and reach y_k back

$$\begin{aligned} &= k'' l(C) \\ &< k'' \delta \end{aligned}$$

The sum of k weights is less than $k''\delta$ with $k'' < k$, which implies

$$\exists C_i, \text{such that } l(C_i) < \delta$$

But this is a contradiction since C was the smallest cycle having weight less than δ. Hence, the perfect matching M is unique. Similarly, the perfect matching induced by C in the other direction too is unique. Applying Lemma 4 successively on both sides, we know that B' obtained by exchanging as per C is a common basis for both matroids. Further, we have

$$\begin{aligned} w(B') &= w(B) - l(C) \\ w(B') &> w(B) - \delta \end{aligned}$$

Hence we have proved that if G has a cycle with small weight, then B is not δ-optimal, thus proving the necessity of the claim.

4.1 Lower bounding cycles

In order to use Lemma 2 to solve the inverse matroid intersection problem efficiently using quadratic programming, we need a way to formulate this condition as a polynomial number of linear constraints. We now explore a technique to express the condition that a given graph has no small (of length less than δ) cycles concisely. Say we are given a directed graph $G = (V, A)$ and our task is to assign edge-lengths so that all cycles in G have weight at least δ. Letting the edge-lengths to be variables, the feasible region in this case is unbounded and is defined by a constraint for every cycle in G, i.e. we have the region R_1 in m dimensions defined by -

$$\begin{aligned} &\mathbf{R_1}: \\ &\sum_{e \in C} l_e \geq \delta \qquad \text{For all cycles C} \qquad (8) \end{aligned}$$

Of course, this formulation has an exponential number of constraints. Although the ellipsoid algorithm can be used to solve the quadratic program in polynomial time, it is often too slow for practical use. We now show that we can obtain a concise extended formulation by adding a few extra variables.

Suppose we have variables d_{xy} representing the shortest distance between vertices x and y. In this case, the graph has no cycle of weight less than δ if and only if $d_{xx} \geq \delta$ for all vertices x (assume $d_{xx} = \infty$, if x is not in any cycle).

Consider the region R_2 in $m + n^2$ dimensions.

$$\begin{aligned}
&\mathbf{R_2}: \\
&d_{xy} \le l_{(xy)} && \text{For all } (x,y) \in A && (9)\\
&d_{xz} \le d_{xy} + l_{(yz)} && \text{For all } x,z \in V \text{ and } y \text{ s.t. } (y,z) \in A && (10)\\
&d_{xx} \ge \delta && \text{For all } x \in V && (11)
\end{aligned}$$

Constraints (9) and (10) enforce triangle inequality, and (11) enforce the condition that all cycles are large. We now prove that optimizing any function of l over R_1 is equivalent to optimizing the same over R_2.

Lemma 5. *R_1 is identical to the projection of R_2 on the m dimensions corresponding to the edge-lengths.*

Proof. $\mathbf{R_1} \subseteq \mathbf{Projection(R_2)}$: Let $l : E \to \mathbb{R}$ denote a point in R_1. Since the constraints (9) and (10) are always valid for a *true* distance function, let $d : V \times V \to \mathbb{R}$ denote the actual distance function in the graph induced by l. Such a d definitely satisfies constraints (9) and (10). Additionally, for all vertices x belonging to some cycle, since all cycles under l have weight at least δ, we have $d_{xx} \ge \delta$. For a vertex x which does not belong to any cycle, one can simply set $d_{xx} = \infty$.

$\mathbf{Projection(R_2)} \subseteq \mathbf{R_1}$**:** Consider a point in R_2. We now have the lengths of edges l_e as well as some d_{xy} values. Consider any cycle $C = (x_1, x_2, \ldots, x_k, x_1)$ in the graph. Applying constraint (10) repeatedly we get

$$d_{x_1x_1} \le l_{(x_1x_2)} + l_{(x_2x_3)} + \ldots + l_{(x_{k-1}x_k)} + l_{(x_kx_1)} \tag{12}$$

and also by constraint (11), we have

$$d_{x_1x_1} \ge \delta \tag{13}$$

Hence we have, $l_{(x_1x_2)} + l_{(x_2x_3)} + \ldots + l_{(x_{k-1}x_k)} + l_{(x_kx_1)} \ge \delta$, i.e. $\sum_{e \in C} l_e \ge \delta$ which means that the l_e values are feasible in R_1.

Hence, optimizing any function of the l_e variables over R_1 is equivalent to optimizing it over R_2. However, R_2 has only $m + mn + n$ constraints and $n^2 + m$ variables.

4.2 Putting it together

Lemmas 2 and 5 suggest a way to solve the δ-margin inverse matroid intersection problem. As per the requirements of Lemma 2, given the two matroids and the common basis B, construct the exchange graph $G = (V, A = A_1 \cup A_2)$. Let $w : X \to \mathbb{R}^+$ be the original weight function and let w' be the new weight function which we desire. If l is the arc lengths of G, according to the construction of Lemma 2, $l_{xy} = w'(x)$ and $l_{yx} = -w'(y)$ where $x \in B, y \in S \setminus B$. Further, the objective that we minimize is the L_2 norm of $w - w'$. We can now add these additional constraints and the objective to the region R_2 as per Lemma 5 to obtain the minimum change on the weights of elements so that the exchange graph has no small cycles and hence B is $\delta-$optimal.

$$\min_{w'} \sum_{e \in X} (w'(e) - w(e))^2 \qquad \textbf{subj. to:} \tag{14}$$

$$l_{xy} = w'(x), \qquad \forall (x,y) \in A_1 \tag{15}$$

$$l_{yx} = -w'(y), \qquad \forall (y,x) \in A_2 \tag{16}$$

$$d_{xy} \leq l_{xy}, \qquad \forall (x,y) \in A \tag{17}$$

$$d_{xz} \leq d_{xy} + l_{yz}, \qquad \forall x, z \in V, \forall (y,z) \in A \tag{18}$$

$$d_{xx} \geq \delta, \qquad \forall x \in V \tag{19}$$

4.3 Maximum Weight Arborescence

Given a directed graph, a r-arborescence (also known as a branching) is the directed analogue of a spanning tree and is defined as a set of edges T spanning all vertices such that every vertex (except r) has exactly one incoming edge in T. It is well known that an arborescence in a directed graph is a basis in the intersection of a graphical matroid and a partition matroid. We analyze the complexity of the above technique for the special case of maximum weight arborescence. Let G denote the graph in question having n vertices and m edges.

The exchange graph G_{ex} has a vertex for every edge of G, i.e., $n_{ex} = m$. The bipartition of G_{ex} is such that we have components of size n and $m - n$ respectively. Hence we have $m_{ex} = O(mn)$. As seen in Section 4.1, we use $O(n_{ex}^2)$ variables and $O(m_{ex}n_{ex})$ contraints. Thus, putting it all together, we have a quadratic program with $O(m^2)$ variables and $O(m^2 n)$ constraints.

The inverse maximum weight arborescence problem is important as it can used as a subroutine in the online learning for dependency parsing [21]. The dependency parse tree of a sentence can be represented as an arborescence over a graph consisting of every word in the sentence as a node. In Appendix D, we show experimental results for dependency parsing using our framework.

Shortest $s-t$ paths. Given a weighted graph $G = (V, E, w)$, a path P between terminals s and t, and a margin δ, the inverse shortest s-t path problem is to find a minimum perturbation to w (minimizing the L_2 norm) so that P is shorter than all other paths between s and t by at least δ under the new weight function. As shown by [22], the inverse shortest s-t path problem can be reduced to the inverse arborescence problem. Let G' be G augmented by adding zero weight edges from t to all other vertices. It can be easily observed that P is the shortest s-t path in G if and only if P and a subset of the zero weight edges form the minimum weight s-arborescence of G'. Thus we can use an algorithm for inverse minimum weight arborescence to solve the inverse shortest path problem.[⋆]

[⋆] Inverse minimum weight arborescence problem can be solved similar to the inverse maximum weight arborescence problem

5 Perfect Matchings in Bipartite Graphs

For the bipartite maximum weight perfect matching inverse problem, the previous technique yields a quadratic program having $O(m^2)$ variables and $O(m^2)$ constraints as the exchange graph is sparse. In this section we show that we can in fact obtain more concise formulations. Recall that for a given edge weighted, bipartite graph $G = (X \cup Y, E, w)$, and a perfect matching M, an alternating cycle is a cycle in G in which edges alternate between those that belong to M and those that do not. An alternating cycle C is called δ-augmenting, if $\sum_{e \in C \cap M} w(e) < \sum_{e \in C \setminus M} w(e) + \delta$. The following characterization of a δ-optimal perfect matching is well known.

Lemma 6. *A perfect matching M is δ-optimal if and only if the graph contains no δ-augmenting cycles.*

The central idea is to construct a directed graph H on just the nodes of X such that any directed cycle in H will correspond to an alternating cycle in G (w.r.t to the matching M) and vice versa. We construct $H = (X, A)$ to be a directed graph such that $(x, z) \in A$ if and only if $\exists y \in Y$ such that $(x, y) \in M$ and $(y, z) \in E$; further let $l(x, z) = w(x, y) - w(y, z)$. Figure 3 shows an example of this construction.

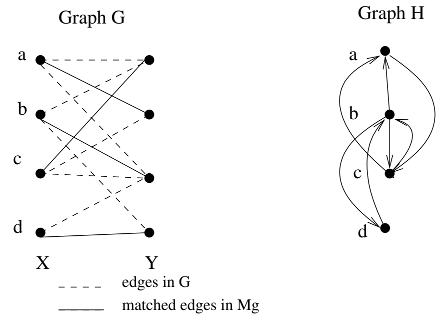

Fig. 3. Example to show construction of H from a bipartite graph G and matching M

Proposition 1. *The auxiliary graph H has a directed cycle of length less than δ if and only if G has a δ-augmenting alternating cycle.*

Proof. **If:** Let $C = (x_0, y_0, x_1, y_1, \ldots, x_k, y_k, x_0)$ be a δ-augmenting cycle in G where all $(x_i, y_i) \in M$. By construction, H has a cycle $C' = (x_0, x_1, \ldots, x_k, x_0)$ and $l(C') = \sum_{i=0}^{k} (w(x_i, y_i) - w(y_i, x_{i+1}))$ (modulo $k+1$) $= \sum_{e \in C \cap M} w(e) - \sum_{e \in C \setminus M} w(e) < \delta$.

Only If: Let $C = (x_0, x_1, \ldots, x_k, x_0)$ be a cycle in H with $l(C) < \delta$. By construction, $\exists$ cycle $C' = (x_0, y_0, x_1, y_1, \ldots, x_k, y_k, x_0)$ in G. Now, $l(C) =$

$\sum_{i=0}^{k}(w(x_i, y_i) - w(y_i, x_{i+1}))$ (modulo $k+1$) $= \sum_{e \in C' \cap M} w(e) - \sum_{e \in C' \setminus M} w(e)$. Thus C' is a δ-augmenting cycle in G.

Using Lemma 6 and Proposition 1 along with Lemma 5, we can formulate the inverse perfect matching problem as a quadratic program having $O(n^2)$ variables and $O(mn)$ constraints.

6 Application : Online learning for structured prediction

In this section, we present a framework for online learning using inverse combinatorial optimization. The structured prediction task is to predict a discrete combinatorial structure (such as an arborescence) given a structured input (such as a graph). The learning task is to learn model parameters so that solving a combinatorial optimization problem on the input instance would return the desired output structure. Structured prediction is extensively used in natural language processing tasks such as obtaining parse trees of a sentence, or automatic language translation.

In the online learning setting, we are presented with a set of T training samples. These consist of an input x_t (for instance, a sentence) and an output y_t (for instance, a syntactic analysis of this sentence described as an arborescence on a graph over the words in the sentence [23, 21]). Each edge in this graph is parameterized by a set of F features that, for instance, indicate how likely one word is to be the subject of another. Thus, each training sample is a pair (x_t, y_t) where x_t is a graph parameterized by features on edges, and y_t is the desired output sub-structure (such as a spanning tree, or an arborescence, or a matching depending on the application). The task is to learn a vector (of length F) of parameters $\boldsymbol{\theta}$ such that when edge weights are computed as inner products between the $\boldsymbol{\theta}$ and the edge's features, the output obtained by computing an optimal sub-structure (spanning tree, etc.) is the desired output with some margin.

Algorithm 1 describes the generic online learning framework for structured predcition. It is parameterized by an user-defined loss function $\ell(y_t, \hat{y})$ that specifies the loss incurred by the prediction $\hat{y}$ with respect to the training solution y_t. Algorithm 1 is an adaptation of the Passive-Aggressive MIRA algorithm [24] for structured prediction.

Note that the minimization problem solved for each training sample is exactly δ-inverse optimization where we minimize the perturbations to the feature parameters instead of the edge weights. In this framework, the different inverse optimization problems we considered have applications for different structured predictions. For example, maximum weight arborescences are used to predict the parse tree of a sentence [23, 21], while maximum weight matchings are used for language translation and word alignments [25].

Since we have shown that we can efficiently solve the inverse optimization problems for a variety of combinatorial structures, we can extend the error bounds of the MIRA algorithm [24] to work for learning the corresponding

```
$\boldsymbol{\theta}_1 = \mathbf{0}$
for $t = 1$ to $T$ do
    Obtain training example $x_t, y_t$
    $w \leftarrow$ weight function s.t. $w(e) = \boldsymbol{\theta}_t \cdot \boldsymbol{f_e}$ where $\boldsymbol{f_e}$ is feature vector of edge $e$
    $\hat{y} \leftarrow$ optimal sub-structure for graph $x_t$ under weights $w$
    Suffer loss $\delta_t = \ell(y_t, \hat{y})$
    Update $\boldsymbol{\theta}_{t+1} = \text{argmin}_{\boldsymbol{\theta}'} ||\boldsymbol{\theta}' - \boldsymbol{\theta}_t||_2^2$ such that
        $w' \leftarrow$ weight function s.t. $w'(e) = \boldsymbol{\theta}' \cdot \boldsymbol{f_e}$ where $\boldsymbol{f_e}$ is feature vector of
    edge $e$
        $y_t$ is the $\delta_t$-optimal sub-structure for graph $x_t$ under weights $w'$
end
Return $\boldsymbol{\theta}_{T+1}$
```
Algorithm 1: Generic online learning framework

structured prediction models. In this section, we present both convergence results and loss bounds for our generic online learning framework. The proofs for these bounds closely follow those in Crammer's Ph.D. dissertation [24] and are relegated to Appendix C for clarity and brevity.

The statement of the convergence result depends on a set of dual variables obtained from the optimization problem in the "Update" step of Algorithm 1. This implicitly encodes constraints over all possible outputs; we denote the dual variable for output y on the t^{th} example by α_y^t. We can show that the cumulative sum of these dual variables is bounded by a constant independent of T, which implies convergence of the learning algorithm.

Theorem 1 (Convergence). *Let $\{(x_t, y_t)\}_{t=1}^T$ be a sequence of structured examples. Let $\boldsymbol{\theta}^*$ be any vector that separates the data with a positive margin $\delta^* > 0$. Assume the loss function is upper bounded: $\ell(y_t, \hat{y}) \leq A$. Then the cumulative sum of coefficients is upper bounded by:*

$$\sum_{t=1}^{T} \sum_{y \in \mathcal{Y}^t} \alpha_y^t \leq 2A \left(\frac{||\boldsymbol{\theta}^*||}{\delta^*} \right)^2 . \tag{20}$$

However, it is not enough to show that the algorithm converges: it could converge to a useless solution! We wish to show that in the process of learning it does not make too many errors. In particular, we show that Algorithm 1 incurs a total hinge loss bounded by a constant also independent of T, which implies that at some point it has exactly solved the learning problem.

Theorem 2 (Total Loss). *Under the same assumptions as above, assume further that the norm of the examples are bounded by R. Then, the cumulative hinge loss ($\mathcal{H}_{\delta_t}$) suffered by the algorithm over T trials is bounded by:*

$$\sum_{t=1}^{T} \mathcal{H}_{\delta_t}(\theta_t, (x_t, y_t)) \leq 8A \left(\frac{R\,||\boldsymbol{\theta}^*||}{\delta^*} \right)^2 . \tag{21}$$

References

[1] Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the International Conference on Machine Learning (ICML). (2001) 282–289

[2] Punyakanok, V., Roth, D.: The use of classifiers in sequential inference. In: Advances in Neural Information Processing Systems (NIPS). (2001)

[3] Collins, M.: Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). (2002)

[4] Taskar, B., Chatalbashev, V., Koller, D., Guestrin, C.: Learning structured prediction models: A large margin approach. In: Proceedings of the International Conference on Machine Learning (ICML). (2005) 897–904

[5] Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research (JMLR) **6** (Sep 2005) 1453–1484

[6] Edmonds, J.: Optimum branchings. Journal of Research of the National Bureau of Standards **71B** (1967) 233–240

[7] Chu, Y., Liu, T.: On the shortest arborescence of a directed graph. Science Sinica **14** (1965) 1396–1400

[8] Crammer, K., Singer, Y.: Ultraconservative online algorithms for multiclass problems. Journal of Machine Learning Research (JMLR) (2003)

[9] Kivenen, J., Warmuth, M.: Exponentiated gradient versus gradient descent for linear predictors. In: Symposium on the Theory of Computing (STOC). (1995)

[10] Heuberger, C.: Inverse combinatorial optimization: A survey on problems, methods, and results. Journal of Combinatorial Optimization **8**(3) (2004) 329–361

[11] Dell'Amico, M., Maffioli, F., Malucelli, F.: The base-matroid and inverse combinatorial optimization problems. Discrete applied mathematics **128**(2) (2003) 337–353

[12] Mao-Cheng, C., Li, Y.: Inverse matroid intersection problem. Mathematical Methods of Operations Research **45**(2) (1997) 235–243

[13] McAllester, D., Collins, M., Pereira, F.: Case-factor diagrams for structured probabilistic modeling. In: Proceedings of the Converence on Uncertainty in Artificial Intelligence (UAI). (2004)

[14] Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. Journal of Machine Learning Research (JMLR) (2006)

[15] McDonald, R., Crammer, K., Pereira, F.: Online large-margin training of dependency parsers. In: Proceedings of the Conference of the Association for Computational Linguistics (ACL). (2005)

[16] Chiang, D.: Hope and fear for discriminative training of statistical translation models. Journal of Machine Learning Research (2012)

[17] Schrijver, A.: Combinatorial optimization: polyhedra and efficiency. Volume 24. Springer Verlag (2003)

[18] Gurobi Optimization, I.: Gurobi optimizer reference manual (2012)

[19] Fujishige, S.: A primal approach to the independent assignment problem. Journal of the Operations Research Society of Japan **20**(1) (1977) 1–15

[20] Frank, A.: A weighted matroid intersection algorithm. Journal of Algorithms **2**(4) (1981) 328–336

[21] McDonald, R., Pereira, F., Ribarov, K., Hajic, J.: Non-projective dependency parsing using spanning tree algorithms. In: Proceedings of the Joint Conference

on Human Language Technology Conference and Empirical Methods in Natural Language Processing (HLT/EMNLP). (2005)

[22] Zhiquan, H., Zhenhong, L.: A strongly polynomial algorithm for the inverse shortest arborescence problem. Discrete applied mathematics **82**(1) (1998) 135–154

[23] Zeman, D.: A statistical approach to parsing of czech. Prague Bulletin of Mathematical Linguistics **69** (1998) 29œ37

[24] Crammer, K.: Online Learning of Complex Categorical Problems. PhD thesis, Hebrew University of Jerusalem (2004)

[25] Taskar, B., Lacoste-Julien, S., Klein, D.: A discriminative matching approach to word alignment. In: Proceedings of EMNLP 2005. (2005)

[26] Nivre, J., Hall, J., J.Nilsson, Chanev, A., Eryigit, G., Kubler, S., Marinov, S., Marsi, E.: The conll 2007 shared task on dependency parsing. In: Proceedings of CoNLL-2007. (2007)

[27] Callison-Burch, C., Bannard, C.: Paraphrasing with bilingual parallel corpora. In: Proceedings of 43rd Annual Meeting of the Association for Computational Linguistics. (2005)

[28] Joachims, T., Finley, T., Yu, C.N.: Cutting-plane training of structural svms. Machine Learning Journal (2009)

[29] Papandreou, G., Yuille, A.: Perturb-and-MAP random fields: Using discrete optimization to learn and sample from energy models. In: International Conference on Computer Vison (ICCV). (2011)

[30] Tarlow, D., Adams, R., , Zemel, R.: Randomized optimum models for structured prediction. In: Proceedings of the Workshop on Artificial Intelligence and Statistics (AI-Stats). (2012)

A Minimum Cost Maximum Flow

Certain problems in structured prediction can be solved using inverse minimum cost maximum flow problems. Consider, for example, the task of assigning reviewers to papers. Suppose we would like to have R reviewers per paper, and at most P papers assigned per reviewer. Such a scenario can be easily modelled as a generalization of bipartite matching with non-unit supply and demands using a minimum cost maximum flow problem. In order to learn the weights of such an instance (suitability of a reviewer for a paper), we can use structured prediction methods that require solving inverse minimum cost maximum flow problems. Formally,

Definition 6 (Inverse Min Cost Max Flow Problem). *Given a directed graph $G = (V, A)$, a capacity function on edges $c : E \to \mathbb{R}^+$, a cost function $w : E \to \mathbb{R}$, and a feasible maximum flow $f : E \to \mathbb{R}^+$, the inverse minimum cost maximum flow problem is to find a new cost function w' such that $||w - w'||_2$ (L_2 norm) is minimized and f is δ-optimal under w'.*

Using Lemma 5, we can easily formulate the inverse min cost max flow problem as a convex optimization problem if we can characterize a minimum cost flow in terms of small cycles in an auxiliary graph. Indeed, the following lemma is well-known.

Lemma 7. *Given an instance of a minimum cost maximum flow problem $G = (V, E, c, w)$, and a maximum flow f, f is δ-optimal if and only if the residual graph G_f has no cycle having weight less than δ.*

B Shortest Path Trees

Suppose we are given a directed graph $G = (V, E)$ with a weight function w on edges, a subtree T_{spt} rooted at r, and a margin δ. The inverse shortest path tree problem is then to minimally modify the edge weights so that T_{spt} becomes the δ-optimal shortest path tree, i.e., for every vertex $v (\neq r)$ in G, the path prescribed by T_{spt} is the δ-optimal shortest path from r to v.

To solve this problem we define variables d_v representing distance labels for each vertex and generate the following quadratic program.

$$\min_{w'} \sum_{e \in E} (w'(e) - w(e))^2 \qquad \textbf{subj. to:} \tag{22}$$

$$d_r = 0 \tag{23}$$

$$d_a + w'(a, b) = d_b \qquad \forall e = (a, b) \in T_{spt} \tag{24}$$

$$d_a + w'(a, b) \geq d_b + \delta \qquad \forall e = (a, b) \notin T_{spt} \tag{25}$$

In other words, our claim is that we can require the distance labels to be the length of the unique path in the shortest path tree. For every other edge $e = (a, b)$, we would like the path to b coming via a to be longer by at least δ. We now prove that these conditions are necessary and sufficient.

Proof. **Sufficient:** Suppose we find a solution to the convex program. The length of the path from r to v via T_{spt} is exactly d_v due to the equality constraints. Consider any other path $P'(r,v) = [r = v_0, v_1, \ldots, v_k = v]$ from r to v. The length of this path is exactly $\sum_{i=0}^{k-1} w'(v_i, v_{i+1}) \geq \sum_{i=0}^{k-1}(d_{v_{i+1}} - d_{v_i} + \delta')$ where $\delta' = 0$ if the edge (v_i, v_{i+1}) belongs to T_{spt} and δ otherwise. Adding, we see that this is at least $d_v + \delta$ since at least one edge on P' does not belong to T_{spt}.

Necessary: Let w' be a feasible weight function, i.e., T_{spt} is δ-optimal under w'. Let d_v denote the distance of vertex v from r under weights w'. Since T_{spt} forms a shortest path tree, its easy to see that constraints (23) and (24) are satisfied. Now suppose that an edge $e' = (a,b) \notin T_{spt}$ does not satisfy constraint (25). Then we have $d_b + \delta > d_a + w'(a,b)$. Since there is a path from r to a of length d_a, we now have a path to b of length shorter than $d_b + \delta$ which violates δ-optimality. Hence, we have a contradiction.

C Proofs of Learning Theory Results

Theorem 1 (Convergence). *Let $\{(x_t, y_t)\}_{t=1}^{T}$ be a sequence of structured examples. Let $\boldsymbol{\theta}^*$ be any vector that separates the data with a positive margin $\delta^* > 0$. Assume the loss function is upper bounded: $\ell(y_t, \hat{y}) \leq A$. Then the cumulative sum of coefficients is upper bounded by:*

$$\sum_{t=1}^{T} \sum_{y \in \mathcal{Y}^t} \alpha_y^t \leq 2A \left(\frac{||\boldsymbol{\theta}^*||}{\delta^*} \right)^2. \tag{20}$$

Proof. Let $\mathcal{L}^t(\boldsymbol{\alpha})$ denote the lagrangian dual of the optimization problem solved in the update step of Algorithm 1. We have

$$\mathcal{L}^t(\boldsymbol{\alpha}) = -\frac{1}{2} || \sum_{y \in \mathcal{Y}^t} \alpha_y^t \phi_{\Delta y}^t ||^2 + \sum_{y \in \mathcal{Y}^t} \alpha_y^t (\delta_t - \theta_t \phi_{\Delta y}^t) \tag{26}$$

Here $\phi_{\Delta_y}^t = \phi(x_t, y_t) - \phi(x_t, y)$ is shorthand for the difference in feature vectors, while δ_t is the specified margin which is taken to be the current loss incurred. Define $\Delta_t = ||\boldsymbol{\theta}_t - \boldsymbol{\theta}^*||^2 - ||\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}^*||^2$. We will establish a bound on the cumulative sum of the dual coeffecients by bounding the sum of Δ_ts above and below.
Upper bounding:

$$\sum_{t=1}^{T} \Delta_t = ||\boldsymbol{\theta}_1 - \boldsymbol{\theta}^*||^2 - ||\boldsymbol{\theta}_{T+1} - \boldsymbol{\theta}^*||^2 \tag{27}$$

$$= ||\boldsymbol{\theta}^*||^2 - ||\boldsymbol{\theta}_{T+1} - \boldsymbol{\theta}^*||^2 \tag{28}$$

$$\leq ||\boldsymbol{\theta}^*||^2 \tag{29}$$

Equation 28 is obtained by substituting $\boldsymbol{\theta}_1 = 0$

Lower bounding:

$$\begin{aligned}
\Delta_t &= ||\boldsymbol{\theta}_t - \boldsymbol{\theta}^*||^2 - ||\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}^*||^2 && (30)\\
&= ||\boldsymbol{\theta}_t - \boldsymbol{\theta}^*||^2 - \left|\left|\boldsymbol{\theta}_t + \sum_y \alpha_y^t \boldsymbol{\phi}_{\Delta y}^t - \boldsymbol{\theta}^*\right|\right|^2 && (31)\\
&= ||\boldsymbol{\theta}_t - \boldsymbol{\theta}^*||^2 - \Bigg[||\boldsymbol{\theta}_t - \boldsymbol{\theta}^*||^2 + \left|\left|\sum_y \alpha_y^t \boldsymbol{\phi}_{\Delta y}^t\right|\right|^2 \\
&\qquad + 2(\boldsymbol{\theta}_t - \boldsymbol{\theta}^*) \cdot \sum_y \alpha_y^t \boldsymbol{\phi}_{\Delta y}^t \Bigg] && (32)\\
&= -\left|\left|\sum_y \alpha_y^t \boldsymbol{\phi}_{\Delta y}^t\right|\right|^2 - 2\boldsymbol{\theta}_t \cdot \sum_y \alpha_y^t \boldsymbol{\phi}_{\Delta y}^t + 2\boldsymbol{\theta}^* \cdot \sum_y \alpha_y^t \boldsymbol{\phi}_{\Delta y}^t && (33)
\end{aligned}$$

Substituting for $\mathcal{L}^t(\boldsymbol{\alpha})$ from Eq. (26), we get

$$\Delta_t = 2\left[\mathcal{L}^t(\boldsymbol{\alpha}) + \sum_y \boldsymbol{\theta}^* \cdot \alpha_y^t \boldsymbol{\phi}_{\Delta y}^t - \sum_y \alpha_y^t \delta_t\right] \tag{34}$$

As $\boldsymbol{\alpha} = 0$ is dual feasible and $\mathcal{L}^t(0) = 0$, we have $\mathcal{L}^t(\boldsymbol{\alpha}) \geq 0$

$$\begin{aligned}
&\geq 2\left[\sum_y \boldsymbol{\theta}^* \cdot \alpha_y^t \boldsymbol{\phi}_{\Delta y}^t - \sum_y \alpha_y^t \delta_t\right] && (35)\\
&\geq 2\left[\sum_y \alpha_y^t \delta^* - \sum_y \alpha_y^t \delta_t\right] && (36)\\
&\geq 2\sum_y \alpha_y^t (\delta^* - \delta_t) && (37)\\
&\geq 2(\delta^* - A)\sum_y \alpha_y^t && (38)
\end{aligned}$$

In the last step, we used the bound on the instantaneous loss. In the previous, we used the assumption on the margin achieved by $\boldsymbol{\theta}^*$. The rest is algebra.

Summing over all t we get

$$\sum_{t=1}^{T} \Delta_t \geq 2(\delta^* - A)\sum_{t=1}^{T}\sum_y \alpha_y^t \tag{39}$$

Combining the bounds in Equations (29) and (39)

$$2(\delta^* - A)\sum_t \sum_y \alpha_y^t \leq ||\boldsymbol{\theta}^*||^2 \tag{40}$$

Now, fix $c = \frac{2A}{\delta^*}$ and scale $\boldsymbol{\theta}^*$ and δ^* by c. Rearrange to get the desired bound.

Lemma 8. *Under the same assumptions as before, and writing $\boldsymbol{\Phi}^t_{\Delta\circledcirc}$ to denote a $|\mathcal{Y}^t| \times F$ matrix whose rows are the feature vectors for all possible outputs, and where p and q are dual (i.e., $\frac{1}{p} + \frac{1}{q} = 1$), the optimal dual variables α_y^t satisfy:*

$$\delta_t - \boldsymbol{\theta}_t \cdot \boldsymbol{\phi}^t_{\Delta_y} \leq \left|\left|\boldsymbol{\alpha}^t\right|\right|_p \left|\left|\boldsymbol{\Phi}^t_{\Delta\circledcirc}\boldsymbol{\phi}^t_{\Delta y}\right|\right|_q \tag{41}$$

Proof. By the enforced δ_t - optimality conditions, we know that for all t and $y \in \mathcal{Y}^t$ -

$$\boldsymbol{\theta}_{t+1} \cdot \boldsymbol{\phi}^t_{\Delta_y} \geq \delta_t \tag{42}$$

Substituting for $\boldsymbol{\theta}_{t+1}$ in terms of $\boldsymbol{\theta}_t$ using the dual optimality conditions

$$(\boldsymbol{\theta}_t + \sum_{z \in \mathcal{Y}^t} \alpha_z^t \boldsymbol{\phi}^t_{\Delta_z}) \cdot \boldsymbol{\phi}^t_{\Delta y} \leq \delta_t \tag{43}$$

$$\delta_t - \boldsymbol{\theta}_t \cdot \boldsymbol{\phi}^t_{\Delta_y} \leq \sum_{z \in \mathcal{Y}^t} \alpha_z^t \boldsymbol{\phi}^t_{\Delta z} \cdot \boldsymbol{\phi}^t_{\Delta_y} \tag{44}$$

$$= \boldsymbol{\alpha}^t \cdot \boldsymbol{\Phi}^t_{\Delta\circledcirc}\boldsymbol{\phi}^t_{\Delta_y} \tag{45}$$

$$\leq \left|\left|\boldsymbol{\alpha}^t\right|\right|_p \times \left|\left|\boldsymbol{\Phi}^t_{\Delta\circledcirc}\boldsymbol{\phi}^t_{\Delta_y}\right|\right|_q \tag{46}$$

The first equality is rewriting things in terms of the matrix $\boldsymbol{\Phi}$, the final step is Hölder's inequality.

Theorem 2 (Total Loss). *Under the same assumptions as above, assume further that the norm of the examples are bounded by R. Then, the cumulative hinge loss ($\mathcal{H}_{\delta_t}$) suffered by the algorithm over T trials is bounded by:*

$$\sum_{t=1}^{T} \mathcal{H}_{\delta_t}(\theta_t, (x_t, y_t)) \leq 8A\left(\frac{R\,||\boldsymbol{\theta}^*||}{\delta^*}\right)^2. \tag{21}$$

Proof. On a round t with non-zero hinge loss, take the y in Eq (46) that has maximal hinge loss (numerator). Take $p = 1$ and $q = \infty$. Then:

$$\sum_t H_{\delta_t}(\boldsymbol{\theta}_t, (x_t, y_t)) \leq \sum_t (\delta_t - \boldsymbol{\theta}_t \cdot \phi^t_{\Delta_y}) \tag{47}$$

$$\leq \sum_t \left|\left|\boldsymbol{\alpha}^t\right|\right|_1 \left|\left|\boldsymbol{\Phi}^t_{\Delta\circledcirc}\boldsymbol{\phi}^t_{\Delta y}\right|\right|_\infty \tag{48}$$

$$= \sum_t (\sum_z \left|\alpha_z^t\right|)(\max_z \boldsymbol{\phi}^t_{\Delta z} \cdot \boldsymbol{\phi}^t_{\Delta y}) \tag{49}$$

Using Hölder's inequality again with $p = q = 2$

$$\leq \sum_t (\sum_z |\alpha_z^t|)(\max_z ||\boldsymbol{\phi}_{\Delta z}^t||_2 \, ||\boldsymbol{\phi}_{\Delta y}^t||_2) \tag{50}$$

By assumption the norm is bounded,

$$\leq \sum_t (\sum_z |\alpha_z^t|)(\max_z (2R)(2R)) \tag{51}$$

$$\leq 4R^2 \sum_t \sum_z |\alpha_z^t| \tag{52}$$

Bounding the cumulative sum using Theorem 1, we get

$$\leq 4R^2 2A \left(\frac{||\boldsymbol{\theta}^*||}{\delta^*} \right)^2 \tag{53}$$

$$= 8A \left(\frac{R \, ||\boldsymbol{\theta}^*||}{\delta^*} \right)^2 \tag{54}$$

D Experimental Analysis

We perform preliminary experiments to demonstrate the efficacy of the online learning framework. We consider two structured prediction tasks: dependency parsing and word alignment for language translation. In dependency parsing, the input is a sentence, and the goal is to find its dependence parse, i.e. evaluate how words in a sentence relate to one-another, forming a tree starting with an empty root node. Figure 4 shows an example of a dependency parse tree of a sentence. The input sentence is considered as a complete graph with a vertex for each word and each edge parameterized by features. The task now is to learn feature weights so that the maximum weight spanning tree in the graph corresponds to the parse tree of the sentence. In word alignment for language translation, we are given two equivalent sentences in two languages and the task is to identify corresponding words. The input instance in this case is considered to be a complete bipartite graph, and the output would be an assignment (matching). Once again given features on edges, the task is to learn feature weights so that the maximum weight perfect matching in the graph would correspond to the correct word alignment.

D.1 Maximum Spanning Trees

Although dependency parsing is better modelled by directed arborescences, for the sake of simplicity we consider only spanning trees in directed graphs in our experiments. For these experiments we used the CoNLL shared task English treebank [26] in order to predict undirected dependency arcs in English sentences. Each word only depends on one word, but can have many dependents.

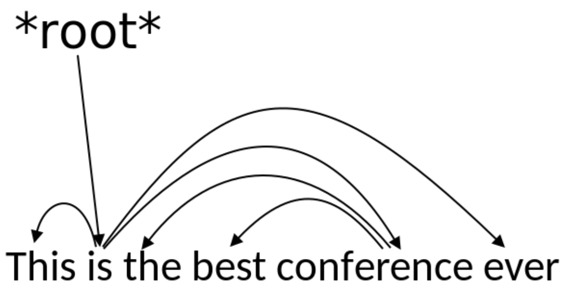

Fig. 4. Example dependency parse tree. The tree describes the relations between head words and their dependents in the sentence

We use a 1500-sentence subset of the training data (36k words) and the test data consists of 3800 sentences (90k words). We train for undirected unlabeled dependencies and evaluate in the same manner. We use standard features: words, word suffixes, position, edge length and predicted part of speech tags.

An averaged structured perceptron baseline obtains an accuracy of 82.7% on the test data. One-best MIRA achieves an accuracy of 84.2%, which is very close to the performance of a structured SVM trained by stochastic gradient descent (accuracy of 84.4%). Our approach achieves a significant improvement on this of 85.1%.

Algorithm	Averaged θ
SVM	84.4
Perceptron	82.7
1-Best MIRA	84.2
Our Algorithm	**85.1**

Table 1. Accuracy results for undirected dependency parsing over the various baselines and our algorithm.

D.2 Bipartite Matching

We are considering German/English alignment at the word level. Given two equivalent sentences in their respective languages, we want to choose an alignment that best fits the "equality" of the respective words in each sentence.

Our data is comprised of 217 manually word-aligned sentences from [27], with many-to-many matchings possible. Since we are restricted to one-to-one alignment (matching) ,we enforce this restriction here by pruning extra edges until we have 1-to-1 matchings only.

The graph structure is as follows: Given a German sentence of length n and an English sentence of length m, we construct a complete bipartite graph

$G = (X_m, Y_n, E)$, where X_m is the German sentence of length m. Y_n is an English sentence of length n. E are the possible alignments for the two sentences, which are currently fully connected. Further, to ensure that the induced matching will be perfect(as required from our problem definition), we add n extra vertices(referred to as dummy vertices) to the German sentence and m dummy vertices to the English sentence, resulting in $G' = (X_m \cup X'_n, Y_n \cup Y'_m, E')$.

Next we will describe the structure of E' in the graph G'. Each vertex of the original G is fully connected with every other vertex in G as before. Each dummy vertex $x'_i \in X'_n$ is fully connected to each dummy node $y'_j \in Y'_m, \forall j$. In addition, each dummy vertex $x'_i \in X'_n$ is also connected to its single, respective real word vertex $y_i \in Y_n$, and similarly Y'_m to X_m. If an alignment for a particular word in X_m or Y_n is not present in M_g due to the nature of word alignment being somewhat sparse, we designate the truth edge to be the one connected to its corresponding dummy vertex and add this to $M_g \subset E$, giving us $M'_g \subset E'$. This will allow us a perfect matching for any two sentences we are given, fitting this particular problem's framework.

Each edge of the graph G' corresponds to a set of feature values for the corresponding words. These include features such as their Dice Coefficient (computed from Europarl corpus), relative word positioning in the sentence, string match without vowels, and others detailed in [25]. We also created slack features and weights, one for each viable edge in the graph. For example, the feature "4_10" means "this edge connects node 4 to node 10". These features are very small valued (the result being that making these features important is expensive), and are used to ensure feasibility during training time for each example presented. The edges corresponding to dummy vertices will *only* have these slack features, which will make certain this perfect matching problem is feasible. After each example is learned, we immediately forget these slack weights.

Algorithm	Averaged θ
Dice=1	36.40
Perceptron	40.52
1-Best MIRA	13.76
Struct SVM	30.42
Our Algorithm	**44.00**

Table 2. Accuracy results for 1-to-1 matching with 10 passes over data with 80-20 train-test split

Results One simple baseline we are testing against is the use of only DICE values. This is being used as a sanity-check for other algorithms' performance. This simple baseline obtained an accuracy of 36.4%. Averaged MIRA performs quite poorly on this dataset with these features, at 13.76%. In our experiments we are only comparing our algorithm with 1-best MIRA, which is unfair as it only constructs the constraints based on the single best matching based on our old weight vector w. Limited to optimizing against one matching appears to be insufficient to learn anything of value. Again however, our algorithm supersedes any k chosen for MIRA, as we ensure our matching beats every single other possible matching. Averaged Perceptron performs better on this dataset, at 40.52%. The variance from run to run was very small compared to the other baselines. Our algorithm obtains an accuracy of 44.0%. This failure of 1-Best MIRA is analogous to a corresponding failure of structured SVMs optimized with cutting plane(30.42%) [5, 28]: when the problem is "hard" (in the sense that $\boldsymbol{\theta}^*$ has high loss), this approach appears to perform quite poorly in practice [29, 30].