On Learning Linear Ranking Functions for Beam Search

Yuehua Xu XUYU@EECS.OREGONSTATE.EDU
Alan Fern AFERN@EECS.OREGONSTATE.EDU
School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331 USA

Abstract

Beam search is used to maintain tractability in large search spaces at the expense of completeness and optimality. We study supervised learning of linear ranking functions for controlling beam search. The goal is to learn ranking functions that allow for beam search to perform nearly as well as unconstrained search while gaining computational efficiency. We first study the computational complexity of the learning problem, showing that even for exponentially large search spaces the general consistency problem is in NP. We also identify tractable and intractable subclasses of the learning problem. Next, we analyze the convergence of recently proposed and modified online learning algorithms. We first provide a counter-example to an existing convergence result and then introduce alternative notions of "margin" that do imply convergence. Finally, we study convergence properties for ambiguous training data.

1. Introduction

Heuristic search is a powerful problem solving paradigm, but is often impractical due to memory and time constraints. One common way to attempt to maintain tractability is to use beam search, which maintains a "beam" of the heuristically best b nodes and prunes all other nodes from the search queue. While beam search is not guaranteed to be complete nor optimal, if the heuristic is good enough, good solutions will be found quickly.

This paper studies the problem of learning heuristics, or ranking functions, that allow beam search to quickly find solutions, without seriously sacrificing optimality. We consider this problem for linear ranking functions, where each search node v is associated with a feature vector $f(v)$ and nodes are ranked according to $w \cdot f(v)$ where w is a weight vector. Each training instance corresponds to a search space that is labeled by a set of target solutions, each solution being a (satisficing) path from the initial node to a goal node. Our goal is to learn a weight vector w so that a beam search of a specified beam width always maintains one of the target paths in the beam until reaching a goal node. Ideally, if the training set is representative, w quickly guides beam search to a solution for new problems.

Appearing in *Proceedings of the 24^{th} International Conference on Machine Learning*, Corvallis, OR, 2007. Copyright 2007 by the author(s)/owner(s).

Recent research has demonstrated good empirical results for learning beam search ranking functions. In the area of structured classification, Daume III and Marcu (2005) learned beam search ranking functions for the natural language processing tasks of chunking and joint chunking/tagging. In the area of automated planning, Xu et al. (2007) learned ranking functions to control a state-space search planner that outperformed state-of-the-art planners in a number of benchmark domains.

Motivated by those successes, this paper focuses on the computational complexity of the learning problem and the convergence properties of various learning algorithms. Our complexity results show that even for exponentially large search spaces, the consistency problem (i.e. finding a w that solves all training instances) is in NP. We also identify core tractable and intractable subclasses of the problem. Interestingly, some of these subclasses resemble more traditional "learning to rank" problems e.g. (Agarwal & Roth, 2005), with clear analogies to applications.

Despite the hardness results, Daume III and Marcu (2005) and Xu et al. (2007) obtained good empirical success using Perceptron-style online learning algorithms, motivating our study of their convergence. In this direction, we demonstrate a counter example to a recent convergence result and then introduce stronger notions of "problem margin" that we prove are sufficient to guarantee convergence. Finally, we extend the analysis to ambiguous training data.

2. Problem Setup

We first define breadth-first and best-first beam search, the two paradigms considered in this work. A *search space* is a tuple $\langle I, s(\cdot), f(\cdot), < \rangle$, where I is the initial search node, s is a successor function from search nodes to finite sets of search nodes, f is a feature function from search nodes to m-dimensional real-valued vectors, and $<$ is a total preference ordering on search nodes. We think of f as defining properties of search nodes that are useful for evaluating their relative goodness and $<$ as defining a canonical ordering on nodes, e.g. lexicographic. In this work, we use f to define a linear ranking function $w \cdot f(v)$ on nodes where w is an m-dimensional weight vector, and nodes with larger values are considered to be higher ranked. Since a given w may assign two nodes the same rank, we use $<$ to break ties such that v is ranked higher than v' given $w \cdot f(v') = w \cdot f(v)$ and $v' < v$, arriving at a total rank ordering on search nodes. We denote this total rank ordering as $r(v', v|w, <)$, or just $r(v', v)$ when w and $<$ are clear from context, indicating that v is ranked higher than v'.

Given a search space $S = \langle I, s(\cdot), f(\cdot), < \rangle$, a weight vector w, and a beam width b, *breadth-first beam search* simply conducts breadth-first search, but at each search depth keeps only the b highest ranked nodes according to r. More formally, breadth-first beam search generates a unique *beam trajectory* $(B_0, B_1, \ldots)$ as follows,

- $B_0 = \{I\}$ is the initial beam;
- $C_{j+1} = \mathbf{BreadthExpand}(B_j, s(\cdot)) = \bigcup_{v \in B_j} s(v)$ is the depth $j+1$ *candidate set* of the depth j beam;
- B_{j+1} is the unique set of b highest ranked nodes in C_{j+1} according to the total ordering r.

Note that for any j, $|C_j| \leq cb$ and $|B_j| \leq b$, where c is the maximum number of children of any search node.

Best-first beam search is almost identical except that we replace the function $\mathbf{BreadthExpand}$ with $\mathbf{BestExpand}(B_j, s(\cdot)) = B_j \cup s(v^*) - v^*$, where v^* is the unique highest ranking node in B_j. Thus, instead of expanding all nodes in the beam at each search step, best-first search is more conservative and only expands the single best node. Note that unlike breadth-first search this can result in beams that contain search nodes from different depths of the search space relative to I.

The training set $\{\langle S_i, P_i \rangle\}$ for our learning problems consists of pairs where the $S_i = \langle I_i, s_i(\cdot), f_i(\cdot), <_i \rangle$ are search spaces constrained such that each f_i has the same dimension. The P_i are *target search paths* that describe desirable search paths through the space S_i. The goal is to learn ranking functions that can efficiently guide beam search to keep at least one target path in the beam. For example, Xu et al. (2007) considered learning in the AI planning domain where the S_i corresponded to AI planning problems, encoding the state space and available actions, and the P_i encoded optimal or satisficing plans for those problems. The goal was to learn a ranking function that quickly finds at least one of the target solution plans for each problem.

We represent each $P_i = (P_{i,0}, P_{i,1}, \ldots, P_{i,d})$ as a sequence of sets of search nodes where $P_{i,j}$ contains target nodes at depth j. Note that $P_{i,0} = B_{i,0} = \{I_i\}$. We will refer to the maximum size t of any target node set $P_{i,j}$ as the *target width* of P_i. For our hardness results, we need not assume any special properties of the P_i. However, for our convergence results, we will require that the target trajectories be *dead-end free*, which means that for all i and $j < d$, each $v \in P_{i,j}$ has at least one child node $v' \in P_{i,j+1}$.

Intuitively, for a dead-end free training set each P_i represents a layered direct graph with at least one path from each target node to a "goal node" in $P_{i,d}$. Thus, the training set specifies not only a set of goals for each search space but also gives possible solution paths to the goals. For simplicity, we assume that all target solution paths have depth d, but all results easily generalize to non-uniform depths.

For breadth-first beam search we specify a learning problem by giving a training set and a beam width $\langle \{\langle S_i, P_i \rangle\}, b \rangle$. The objective is to find a weight vector w that generates a beam trajectory containing at least one of the target paths for each training instance. More formally, we are interested in the consistency problem:

Definition 1 (Breadth-First Consistency). *Given the input $\langle \{\langle S_i, P_i \rangle\}, b \rangle$ where $P_i = (P_{i,0}, P_{i,1}, \ldots, P_{i,d})$, the breadth-first consistency problem asks us to decide whether there exists a weight vector w such that for each S_i, the corresponding beam trajectory $(B_{i,0}, B_{i,1}, \ldots, B_{i,d})$ using beam width b satisfies $B_{i,j} \cap P_{i,j} \neq \emptyset$ for each j?*

A weight vector that demonstrates a "yes" answer is guaranteed to allow a breath-first beam search of width b to find a goal node in d search steps for all training instances.

Unlike for breadth-first beam search, the length of the beam trajectory required by best-first beam search to reach a goal node can be greater than the depth d of the target paths. This is because best-first beam search, does not necessarily increase the maximum depth of search nodes in the beam at each search step. Thus, in addition to specifying a beam width for the learning problem, we also specify a maximum number of search steps, or horizon, h. The objective is to find a weight vector that allows a best-first beam search to find a goal node within h search steps, while always keeping some node from the target paths in the beam.

Definition 2 (Best-First Consistency). *Given the input $\langle \{\langle S_i, P_i \rangle\}, b, h \rangle$, where $P_i = (P_{i,0}, \ldots, P_{i,d})$, the best-first consistency problem asks us to decide whether there is a weight vector w such that for each S_i there is a beam trajectory $(B_{i,0}, \ldots, B_{i,k})$ of beam width b and length $k \leq h$ such that $B_{i,k}$ contains at least one goal node in $P_{i,d}$ and each $B_{i,j}$ contains at least one node in $\bigcup_j P_{i,j}$?*

3. Computational Complexity

In this section, we study the computational complexity of the above consistency problems. We first focus on breadth-first beam search, and give best-first results at the end. It is important to note that the size of the search spaces will typically be exponential in the encoding size of the learning problem. For example, in planning domains, STRIPS operators compactly encode exponentially large planning problems. We measure complexity in terms of the problem size, not the exponentially larger search space size.

We consider subclasses of breadth-first consistency by placing constraints on the following problem parameters: n - the number of training instances, d - the depth of target solution paths, c - the maximum number of children of any node, t - the maximum target width of any P_i, and b - the beam width. We will restrict our attention to problem classes where the maximum number of children c is polynomial in the problem size, which is true for nearly all search spaces of interest. We also restrict b to be polynomial in the problem size, which again is a reasonable assumption since otherwise each beam-search step could take exponential time. We will also assume that all feature functions can be evaluated in polynomial time in the problem size.

We first show that breadth-first consistency is in NP even for exponentially large search spaces. Given a weight vector w and beam width b, we can generate a unique depth d beam trajectory for each training instance. Here we consider the inverse problem of checking whether a set of hypothesized beam trajectories, one for each training instance, could have been generated by some weight vector. The algorithm *TestTrajectories* in Figure 1 efficiently carries out this check. The idea is to construct a linear program containing constraints on w for generating the trajectories.

Lemma 1. *Given a set of search spaces $\{S_i\}$ and a corresponding set of beam trajectories $\{(B_{i,0},\ldots,B_{i,d})\}$ of width b, the algorithm TestTrajectories (Figure 1) decides in polynomial time whether there exists a weight vector w that can generate $(B_{i,0},\ldots,B_{i,d})$ in S_i for all i.*

Proof. (Sketch) It can be shown that w satisfies the constraints generated by *TestTrajectories* iff for each i, j, $r(v', v| <_i, w)$ leads beam search to generate $B_{i,j+1}$ from $B_{i,j}$. The linear program contains m variables and at most $ndcb^2$ constraints. Since we are assuming that the maximum number of children of a node v is polynomial in the size of the learning problem, the size of the linear program is also polynomial and thus can be solved in polynomial time, e.g. using Karmarkar's algorithm. □

This Lemma shows that we can check certificate beam trajectories efficiently, which implies the following.

Theorem 1. *Breadth-first consistency is in NP.*

```
ExhaustiveAlgorithm ($\{\langle S_i, P_i\rangle\}, b$)
$\Gamma$ = Enumerate($\{\langle S_i, P_i\rangle\}, b$)
// enumerates all possible sets of beam trajectories
for each $y = \{(B_{i,0}\ldots, B_{i,d})\} \in \Gamma$
    if IsConsistent($\{P_i\}, \{(B_{i,0}\ldots, B_{i,d})\}$) then
        if TestTrajectories($\{S_i\}, \{(B_{i,0},\ldots, B_{i,d})\}$) then
            return $w$
return false
```

```
TestTrajectories($\{S_i\}, \{(B_{i,0},\ldots, B_{i,d})\}$)
// $S_i = \langle I_i, s_i(\cdot), f_i(\cdot), <_i\rangle$
construct a linear programming problem $LP$ as below
    the variables are $Z = \{z_1, z_2, \ldots, z_m\}$
    for $i = 1,\ldots,n, j = 1,\ldots,d$
        $C_{i,j}$ = BreadthExpand($B_{i,j-1}, s_i(\cdot)$)
        if $B_{i,j} \subseteq C_{i,j}$ then
            for each $v \in B_{i,j}$ and $v' \in C_{i,j} - B_{i,j}$
                if $v' <_i v$ then
                    add a constraint $Z \cdot f_i(v) \geq Z \cdot f_i(v')$
                else add a constraint $Z \cdot f_i(v) > Z \cdot f_i(v')$
        else return false
$w$ = LPSolver($LP$)// e.g. Karmarkar's algorithm
if $LP$ is solved then
    return $w$
return false
```

Figure 1. The exhaustive algorithm.

Proof. Given a learning problem $\langle\{\langle S_i, P_i\rangle\}, b\rangle$ our certificates correspond to sets of beam trajectories $\{(B_{i,0},\ldots,B_{i,d})\}$ each of size at most $O(ndb)$ which is polynomial in the problem size. The certificate can then be checked in polynomial time to see if for each i, $(B_{i,0},\ldots,B_{i,d})$ contains a target solution path encoded in P_i as required by Definition 1. If it is consistent then according to Lemma 1 we can efficiently decide whether there is a w that can generate $\{(B_{i,0},\ldots,B_{i,d})\}$. □

This result suggests an enumeration-based decision procedure for breadth-first consistency as given in Figure 1. The procedure enumerates sets of beam trajectories, checking whether they contain target paths and if so calls *TestTrajectories.* The following gives us the worst case complexity of this algorithm in terms of the key problem parameters.

Theorem 2. *The algorithm in Figure 1 decides breadth-first consistency and returns a solution weight vector if there is a solution in time $O\left((t + poly(m))\,(cb)^{bdn}\right)$.*

Proof. We first bound the number of certificates. Breadth-first beam search expands nodes in the current beam, resulting in at most cb nodes, from which b nodes are selected for the next beam. Enumerating these possible choices over d levels and n trajectories, one for each training instance, we can bound the number of certificates by $O\left((cb)^{bdn}\right)$. For each certificate the enumeration process checks consistency with the target paths $\{P_i\}$ in time $O(tbdn)$ and then calls *TestTrajectories* which runs in time poly$(m, ndcb^2)$. The total time complexity then is $O\left(\left(tbdn + \text{poly}(m, ndcb^2)\right)(cb)^{bdn}\right) = O\left((t + \text{poly}(m))\,(cb)^{bdn}\right)$. □

Not surprisingly the complexity is exponential in the beam width b, target path depth d, and number of training instances n. However, it is polynomial in the maximum number of children c and the maximum target width t. Thus, breadth-first consistency can be solved in polynomial time for any problem class where b, d, and n are constants. Of course, for most problems of practical interest these constants would be too large to be practical. This leads to the question of whether we can do better than the exhaustive algorithm. For at least one problem class we can,

Theorem 3. *The class of breadth-first consistency problems where $b = 1$ and $t = 1$ is solvable in polynomial time.*

Proof. (Sketch.) Since $t = 1$ each training instance has exactly one target solution path. In addition, for $b = 1$ the only beam trajectory consistent with the target path is the target itself. Thus, we can simply pass the target path for each instance to *TestTrajectories* and return the result of that single call in polynomial time. $\square$

This problem class corresponds to the case where each training instance is labeled by exactly a single solution path and we are asked to find a w that leads a greedy hill-climbing search, or reactive policy, to follow those paths.

Unfortunately, outside of the above problem classes it appears that breadth-first consistency is computationally hard even under strict assumptions. In particular, the following three results show that if any one of b, d, or n are allowed to vary then the consistency problem is hard even when the other problem parameters are small constants.

First, we show that the problem class where b is allowed to vary, but $n = 1$, $d = 1$ and $t = 1$ is NP-complete. That is, a single training instance involving a depth one search space is sufficient for hardness. This problem class, resembles more traditional ranking problems and has a nice analogy in the application domain of web-page ranking, where the depth 1 leaves of our search space correspond to possibly relevant web-pages for a particular query. One of those pages is marked as a target page, e.g. the page that a user eventually went to. The learning problem is then to find a weight vector that will cause for the target page to be ranked among the top b pages, where for example b may represent the number of results that can be easily parsed by a user. Our result shows that this problem is NP-complete.

Theorem 4. *The class of breadth-first consistency problems where $n = 1$, $d = 1$, and $t = 1$ is NP-complete.*

Proof. (Sketch.) We reduce from the Minimum Disagreement problem for linear binary classifiers, which is NP-complete (Hoffgen et al., 1995). The problem's input is a training set $S = \{x_1^+, \cdots, x_{r_1}^+, x_1^-, \cdots, x_{r_2}^-\}$ of positive and negative m-dimensional vectors and a positive integer k. We are asked to decide if there is a weight vector w that commits at most k misclassifications, where w classifies a x as positive iff $w \cdot x \geq 0$ and negative otherwise. Given S and k we construct an instance $\langle\langle S_1, P_1\rangle, b\rangle$ of breadth-first consistency as follows. Let $S_1 = \langle I, s(\cdot), f(\cdot), <\rangle$, where $s(I) = \{q_0, q_1, \cdots, q_{r_1+r_2}\}$. For each $i \in \{1, \cdots, r_1\}$, define $f(q_i) = -x_i^+$. For each $i \in \{1, \cdots, r_2\}$,define $f(q_{i+r_1}) = x_i^-$. Define $f(q_0) = 0 \in R^m$, $P_1 = (\{I\}, \{q_0\})$, and $b = k + 1$. Define the total ordering $<$ to be any total ordering in which $q_i < q_0$ for every $i = 1, \ldots, r_1$ and $q_0 < q_i$ for every $i = r_1+1, \ldots, r_1+r_2$.

It can be shown that there exists a linear classifier with at most k misclassifications if and only if there exists a solution to the corresponding breadth-first consistency problem. Intuitively, the target node q_0 represents zero and the q_i represent positive or negative examples. Any w that misclassifies the example of q_i will rank q_i higher than q_0. Thus, if there are more than k misclassification then the target node q_0 will be forced out of the beam. $\square$

The next result shows that if we allow the number of training instances to vary, then the problem remains hard even when the target path depth and beam width are equal to one. This problem class can be viewed as a novel form of multiple-instance learning (Dietterich et al., 1997). Each training instance can be viewed as a bag of m-dimensional vectors, some of which are labeled as positive (i.e. the target nodes). The learning goal is to find a w that for each bag, ranks one of the positive vectors as best.

Theorem 5. *The class of breadth-first consistency problems where $d = 1$, $b = 1$, $c = 6$, and $t = 3$ is NP-complete.*

Proof. (Sketch.) We reduce from 3-SAT (Garey & Johnson, 1979). Let $Q = \{q_{11} \vee q_{12} \vee q_{13}, \ldots, q_{n1} \vee q_{n2} \vee q_{n3}\}$ be the clauses of a 3-SAT instance where the q_{ij} are positive or negative literals over the set of variables $U = \{u_1, \ldots, u_m\}$. We construct a corresponding breadth-first consistency problem $\langle\{\langle S_i, P_i\rangle\}, b = 1\rangle$ such that for each clause $q_{i1} \vee q_{i2} \vee q_{i3}$ there is a single, depth-one training instance with $s_i(I_i) = \{p_{i,1}, \cdots, p_{i,6}\}$, target paths $P_i = (\{I_i\}, \{p_{i,1}, p_{i,2}, p_{i,3}\})$, and preference ordering $<_i$ such that $p_{i,j} <_i p_{i,k}$ for $j = 1, 2, 3$ and $k = 4, 5, 6$. Let $e_k \in \{0, 1\}^m$ denote a vector of zeros except for a 1 in the k'th component. For each $i = 1, \ldots, n$, $j = 1, 2, 3$, if $q_{ij} = u_k$ for some k then $f_i(p_{i,j}) = e_k$ and $f_i(p_{i,j+3}) = -e_k/2$, if $q_{ij} = \neg u_k$ for some k then $f_i(p_{i,j}) = -e_k$ and $f_i(p_{i,j+3}) = e_k/2$. One can show that Q is satisfiable if and only if there exists a solution to the corresponding breadth-first consistency problem. Intuitively, the numeric sign of the weight vector components represent truth values over U and each training instance is constructed so that w ranks a target node as best iff a literal in the clause is true in the corresponding truth assignment. $\square$

Finally, we show that when the depth d is variable the consistency problem remains hard even when $b = n = 1$.

Theorem 6. *The class of breadth-first consistency problems where $n = 1$, $b = 1$, $c = 6$, and $t = 3$ is NP-complete.*

Proof. (Sketch.) Given a consistency problem A with $d = 1$, $b = 1$, $c = 6$ and $t = 3$, we can construct an equivalent problem B with $n = 1$, $b = 1$, $c = 6$, and $t = 3$. This can be done by noting that problem A has multiple depth one instances and that all of these instances must be satisfied to obtain consistency. Problem B, rather can have only a single instance but arbitrary depth. We construct problem B by "stacking" the n training instances from A into a single depth n instance. This can be done in a way such that a w satisfies all of the A instances iff it also satisfies the single B instance. □

b	n	d	c	t	Complexity
poly	*	*	poly	*	NP
K	K	K	poly	*	P
1	*	*	poly	1	P
poly	1	1	poly	1	NP-Complete
1	*	1	6	3	NP-Complete
1	1	*	6	3	NP-Complete

Figure 2. Complexity results for breadth-first consistency. Each row corresponds to a sub-class of the problem and indicates the computational complexity. K indicates a constant value and "poly" indicates that the quantity must be polynomially related to the problem size. * indicates that the quantity is unbounded.

Figure 2 summarizes our main complexity results from this section for breadth-first consistency. For best-first beam search, most of these results can be carried over. Recall that for best-first consistency the problem specifies a search bound h in addition to a beam width. Using a similar approach as above we can show that best-first consistency is in NP assuming that h is polynomial in the problem size, which is a reasonable assumption. Similarly, one can extend the polynomial time result for fixed b, n, and d. The remaining results in the table can be directly transferred to best-first search, since in each case either $b = 1$ or $d = 1$ and best-first beam search is essentially equivalent to breadth-first beam search in these cases.

4. Convergence of Online Updates

Above we identified a limited set of tractable problem classes and saw that even very restricted classes remain NP-hard. We also saw that some of these hard classes had interesting application relevance. Thus, it is desirable to consider efficient learning mechanisms that work well in practice. Below we describe two such algorithms.

4.1. Online Perceptron Updates

Figure 3 gives the LaSO-BR algorithm for learning ranking functions for breadth-first beam search. It is an extension of the algorithm in Xu et al. (2007) to multiple target paths and resembles the *learning as search optimization (LaSO)* algorithm by Daume III and Marcu (2005) for best-first search. LaSO-BR iterates through each training instances $\langle S_i, P_i \rangle$, for each one conducting a beam search. After generating the depth j beam for the ith training instance, if none of the target nodes in $P_{i,j}$ are on the beam then a search error is said to have occurred. In this case, we perform a Perceptron-style weight update,

$$w = w + \alpha \cdot \left(\frac{\sum_{v^* \in P_{i,j} \cap C} f(v^*)}{|P_{i,j} \cap C|} - \frac{\sum_{v \in B} f(v)}{b} \right)$$

where $0 < \alpha \leq 1$ is a learning rate parameter, B is the current beam and C is the candidate set from which B was generated (i.e. the beam expansion of the previous beam). For simplicity we assume f is a feature function for all training instances. Intuitively this update rule moves the weights in a direction that increases the rank of the target nodes that appeared in C, and decreases the rank of non-target nodes in the beam. Ideally, this will cause at least one of the target nodes to become preferred enough to remain on the beam next time through the search. After the weight update, the beam is set to the set of target nodes in C and the search continues. Note that the processing of each instance is guaranteed to terminate in d search steps.

```
LaSO-BR ($\{\langle S_i, P_i \rangle\}, b$)
$w \leftarrow 0$
repeat until $w$ is unchanged or a large number of iterations
    for every $i$
        Update-BR($S_i, P_i, b, w$)
return $w$

Update-BR ($S_i, P_i, b, w$)
// $S_i = \langle I_i, s_i(\cdot), f(\cdot), <_i \rangle$ and $P_i = (P_{i,0}, \ldots, P_{i,d})$
$B \leftarrow \{I_i\}$ // initial beam
for $j = 1, \ldots, d$
    $C \leftarrow$ BreadthExpand($B, s_i(\cdot)$)
    for every $v \in C$
        $H(v) \leftarrow w \cdot f(v)$ // compute heuristic value of $v$
    Order $C$ according to $H$ and the total ordering $<_i$
    $B \leftarrow$ the first $b$ nodes in $C$
    if $B \cap P_{i,j} = \emptyset$ then
        $w \leftarrow w + \alpha \cdot \left( \frac{\sum_{v^* \in P_{i,j} \cap C} f(v^*)}{|P_{i,j} \cap C|} - \frac{\sum_{v \in B} f(v)}{b} \right)$
        $B \leftarrow P_{i,j} \cap C$
return
```

Figure 3. Online algorithm for breadth-first beam search.

Figure 4 gives the LaSO-BST algorithm for learning in best-first beam search, which is a slight modification of the original LaSO algorithm. The main difference is in the weight update equation, a change that appears necessary for our convergence analysis. The process is similar to LaSO-BR except that a best-first beam search is conducted, which means that termination for each training instance is not guaranteed to be within d steps. Rather, the number of search steps for a single training instance remains unbounded without further assumptions, which we will address later in this section.

```
LaSO-BST ($\{\langle S_i, P_i \rangle\}, b$)
$w \leftarrow 0$
repeat until $w$ is unchanged or a large number of iterations
    for every $i$
        Update-BST($S_i, P_i, b, w$)
return $w$

Update-BST ($S_i, P_i, b, w$)
// $S_i = \langle I_i, s_i(\cdot), f(\cdot), <_i \rangle$ and $P_i = (P_{i,0}, \ldots, P_{i,d})$
$B \leftarrow \{I_i\}$ // initial beam
$\bar{P} = P_{i,0} \cup P_{i,2} \cup \ldots \cup P_{i,d}$
while $B \cap P_{i,d} = \emptyset$
    $C \leftarrow$ BestExpand($B, s_i(\cdot)$)
    for every $v \in C$
        $H(v) \leftarrow w \cdot f(v)$ // compute heuristic value of $v$
    Order $C$ according to $H$ and the total ordering $<_i$
    $B \leftarrow$ the first $b$ nodes in $C$
    if $B \cap \bar{P} = \emptyset$ then
        $w \leftarrow w + \alpha \cdot \left( \frac{\sum_{v^* \in \bar{P} \cap C} f(v^*)}{|\bar{P} \cap C|} - \frac{\sum_{v \in B} f(v)}{b} \right)$
        $B \leftarrow \bar{P} \cap C$
return
```

Figure 4. Online algorithm for best-first beam search.

4.2. Previous Result and Counter Example

Adjusting to our terminology, Daume III and Marcu (2005) define a training set to be *linear separable* iff there is a weight vector that solves the corresponding consistency problem. Also for linearly separable data they defined a notion of margin of a weight vector, which we refer to here as the *search margin*. Intuitively, the search margin of w is the maximal value by which all target nodes can be downweighted while still maintaining separability of the data.

Definition 3 (Search Margin)**.** *The search margin of a weight vector w for a linearly separable training set is defined as $\gamma = min_{\{(v^*,v)\}}(w \cdot f(v^*) - w \cdot f(v))$, where the set $\{(v^*, v)\}$ contains any pair where v^* is a target node and v is a non-target node that was compared during the beam search guided by w.*

Daume III and Marcu (2005) state that the existence of a w with positive search margin implies convergence of the original LaSO. However, we have found that for subtle reasons a positive search margin is not sufficient to guarantee convergence for LaSO, LaSO-BR, or LaSO-BST. Intuitively, this is because w may "by chance" prune parts of the search space that otherwise would have lead to ranking errors later in the search. For such coincidental solutions, the learning algorithms are unable to reliably derive a training signal that drives them to a solution.

Counter Example 1. *We give a training set for which the existence of a weight vector with positive search margin does not guarantee convergence to a solution weight vector for LaSO-BR or LaSO-BST. Consider a problem with a single training instance with search space shown in Figure 5a, preference ordering $C < B < F < E < D < H < G$, and single target path $P = (\{A\}, \{B\}, \{E\})$. For $b = 2$ the weight vector $w = [\gamma, \gamma]$ solves the problem and has search margin γ. However, tracing through LaSO-BR and LaSO-BST (or the original LaSO) starting with $w' = 0$, shows that the algorithm converges to $w' = 0$ which is not a solution for this problem.*

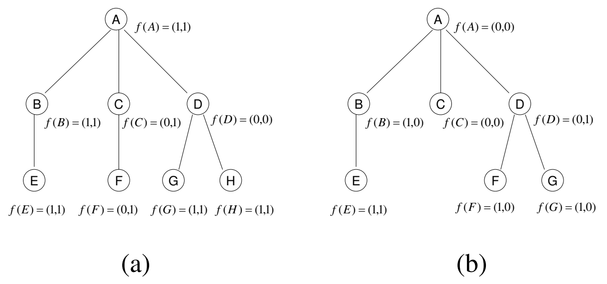

Figure 5. Counter-examples (a) Search margin (b) Level margin

4.3. Convergence Under Stronger Notions of Margin

Given that linear separability (i.e. a positive search margin) is not sufficient to guarantee convergence we consider a stronger notion of margin, the *level margin*, which measures by how much the target nodes are ranked above (or below) other non-target nodes at the same search level.

Definition 4 (Level Margin)**.** *The level margin of a weight vector w for a training set is defined as $\gamma = min_{\{(v^*,v)\}}(w \cdot f(v^*) - w \cdot f(v))$, where the set $\{(v^*, v)\}$ contains any pair such that v^* is a target node at some depth j and v can be reached in j search steps from the initial search node—i.e v^* and v are at the same level.*

For breadth-first beam search, a positive level margin for w implies a positive search margin, but not necessarily vice versa, showing that level margin is a strictly stronger notion of separability. Let R be a constant such that for all training instances, for all nodes v and v', $\|f(v) - f(v')\| \leq R$.

Theorem 7. *Given a dead-end free training set such that there exists a weight vector w with level margin $\gamma > 0$ and $\|w\| = 1$, LaSO-BR will converge with a consistent weight vector after making no more than $(R/\gamma)^2$ weight updates.*

Proof. (Sketch.) First note that the dead-end free property of the training data can be used to show that unless the current weight vector is a solution it will eventually trigger a "meaningful" weight update (one where the previous candidate set contains target nodes).

Let w^k be the weights just before the k'th mistake is made on training instance $\langle S_i, P_i \rangle$, when $B \cap P_{i,j}$ is empty, where B is the depth j beam and C is the candidate set from which B was selected. The occurrence of the mistake indicates that, $\forall v^* \in P_{i,j} \cap C, v \in B, w^k \cdot f(v^*) \leq w^k \cdot f(v)$, which lets us derive an upper bound for $\|w^{k+1}\|^2$. Denoting $C' = P_{i,j} \cap C$ we get,

$$\begin{aligned}\|w^{k+1}\|^2 &= \|w^k + \frac{\sum_{v^* \in C'} f(v^*)}{|C'|} - \frac{\sum_{v \in B} f(v)}{b}\|^2 \\ &= \|w^k\|^2 + \|\frac{\sum_{v^* \in C'} f(v^*)}{|C'|} - \frac{\sum_{v \in B} f(v)}{b}\|^2 \\ &\quad + 2w^k \cdot (\frac{\sum_{v^* \in C'} f(v^*)}{|C'|} - \frac{\sum_{v \in B} f(v)}{b}) \\ &\le \|w^k\|^2 + R^2\end{aligned}$$

where the inequality follows because $w^k \cdot (f(v^*) - f(v)) \le 0$ for all $v^* \in C', v \in B$ and our assumptions on R. By induction we can get $\|w^{k+1}\|^2 \le kR^2$. We now derive a lower bound for $w \cdot w^{k+1}$.

$$\begin{aligned}w \cdot w^{k+1} &= w \cdot w^k + w \cdot (\frac{\sum_{v^* \in C'} f(v^*)}{|C'|} - \frac{\sum_{v \in B} f(v)}{b}) \\ &= w \cdot w^k + \frac{\sum_{v^* \in C'} w \cdot f(v^*)}{|C'|} - \frac{\sum_{v \in B} w \cdot f(v)}{b} \\ &\ge w \cdot w^k + \gamma\end{aligned}$$

This inequality follows from the definition of the level margin. By induction we get that $w \cdot w^{k+1} \ge k\gamma$. Combining this result with the above upper bound on $\|w^{k+1}\|$ and the fact that $\|w\| = 1$ we get that $\sqrt{k}R \ge \|w\|\|w^{k+1}\| \ge w \cdot w^{k+1} \ge k\gamma$, which implies the mistake bound. □

LaSO-BST and LaSO do not have such a guarantee since their beams can contain nodes from multiple levels.

Counter Example 2. *We give a training set for which the existence of a w with positive level margin does not guarantee convergence for LaSO-BST. Consider a single training example with the search space in Figure 5b, single target path $P = (\{A\}, \{B\}, \{E\})$, and preference ordering $C < B < E < F < G < D$. The weight vector $w = [2\gamma, \gamma]$ has a level margin of γ. However, if we follow LaSO-BST (or the original LaSO) started with the weight vector $w' = 0$ and $b = 2$, the algorithm will converges to $w' = 0$ which is not a solution for this problem.*

To guarantee convergence of LaSO-BST, we require an even stronger notion of margin, *global margin*, which measures the rank difference between any target node and any non-target node, regardless of search space level.

Definition 5 (Global Margin)**.** *The global margin of a weight vector w for a training set is defined as $\gamma = min_{\{(v^*,v)\}}(w \cdot f(v^*) - w \cdot f(v))$, where the set $\{(v^*, v)\}$ contains any pair such that v^* is any target node and v is any non-target node in the search space.*

Note that if w has a positive global margin then it has a positive level margin. The converse is not necessarily true. The global margin is similar to the common definitions of margin used to characterize the convergence of linear Perceptron classifiers (Novikoff, 1962).

To ensure convergence of LaSO-BST we also assume that the search spaces are all finite trees. This avoids the possibility of infinite best-first beam trajectories that never terminate at a goal node. Tree structures are quite common in practice and it is often easy to transform a finite search space into a tree. The applications in (Daume III & Marcu, 2005) and (Xu et al., 2007) only involved finite trees.

Theorem 8. *Given a dead-end free training set of finite tree search spaces such that there exists a weight vector w with global margin $\gamma > 0$ and $\|w\| = 1$, LaSO-BST will converge with a consistent weight vector after making no more than $(R/\gamma)^2$ weight updates.*

The proof is similar to that of Theorem 7 except that the derivation of the lower bound makes use of the global margin and we must verify that the restriction to finite tree search spaces guarantees that each iteration of LaSO-BST will terminate with a goal node being reached. We were unable to show convergence for the original LaSO algorithm even under the assumptions of this theorem.

In summary, we introduced three different notions of margin here: search margin, level margin, and global margin. Both algorithms converge for a positive global margin, which implies a positive search margin and a positive level margin. For LaSO-BR, but not LaSO-BST, convergence is guaranteed for a positive level margin, which implies a positive search margin. Finally, a positive search margin corresponds exactly to linear separability, but is not enough to guarantee convergence for either algorithm. This is in contrast to results for classifier learning, where linear separability implies convergence of Perceptron updates.

5. Convergence for Ambiguous Training Data

Here we study convergence for linearly inseparable training data. Inseparability is often the result of training-data ambiguity, in the sense that many "good" solution paths are not included as target paths. For example, this is common in AI planning where there can be many (nearly) optimal solutions, many of which are inherently identical (e.g. differing in the orderings of unrelated actions). It is usually impractical to include all solutions in the training data, which can make it infeasible to learn a ranking function that strictly prefers the target paths over the inherently identical paths not included as targets. In these situations, the above notions of margin will all be negative. Here we consider the notion of *beam margin* that allows for some amount of ambiguity, or inseparability. The following generalizes the analysis in (Xu et al., 2007) to our setting where multiple, rather than single, target paths are allowed per search space.

For each instance $\langle S_i, P_i \rangle$, where $S_i = \langle I_i, s_i(\cdot), f(\cdot), <_i \rangle$ and $P_i = \{P_{i,1}, P_{i,2}, \ldots, P_{i,d_i}\}$, let D_{ij} be the set of nodes that can be reached in j search steps from I_i. That is, D_{ij}

is the set of all possible non-target nodes that could be in beam $B_{i,j}$. A beam margin is a triple (b', δ_1, δ_2) where b' is a non-negative integer, and $\delta_1, \delta_2 \geq 0$.

Definition 6 (Beam Margin)**.** *A weight vector w has beam margin (b', δ_1, δ_2) on a training set $\{\langle S_i, P_i \rangle\}$, if for each i, j there is a set $D'_{ij} \subseteq D_{ij}$ such that $|D'_{ij}| \leq b'$ and*

$$\forall v^* \in P_{i,j}, v \in D_{ij} - D'_{ij}, \;\; w \cdot f(v^*) - w \cdot f(v) \geq \delta_1 \textit{ and,}$$
$$\forall v^* \in P_{i,j}, v \in D'_{ij}, \;\; \delta_1 > w \cdot f(v^*) - w \cdot f(v) \geq -\delta_2$$

A weight vector w has beam margin (b', δ_1, δ_2) if at each search depth it ranks the target nodes better than most other nodes (those in $D_{ij} - D'_{ij}$) by a margin of at least δ_1, and ranks at most b' nodes (those in D'_{ij}) better than the target nodes by a margin no greater than δ_2. Whenever this condition is satisfied we are guaranteed that a beam search of width $b > b'$ guided by w will solve all of the training problems. The case where $b' = 0$ corresponds to the level margin, where the data is separable. By considering $b' > 0$ we can consider cases where there is no "dominating" weight vector that ranks all targets better than all non-targets at the same level. For large enough beam widths, dependent on the beam margin, we can show convergence of LaSO-BR.

Theorem 9. *Given a dead-end free training set such that there exists a weight vector w with beam margin (b', δ_1, δ_2) and $\|w\| = 1$, then for any beam width $b > (1 + \delta_2/\delta_1)\, b' = b^*$, LaSO-BR will converge with a consistent weight vector after making no more than $(R/\delta_1)^2 \left(1 - b^* b^{-1}\right)^{-2}$ weight updates.*

Proof. Let w^k be the weights before the k'th mistake is made. Then $w^1 = 0$. Suppose the k'th mistake is made when $B \cap P_{i,j} = \emptyset$ where B is the beam generated at depth j for the ith training instance. We can derive the upper bound of $\|w^{k+1}\|^2 \leq kR^2$ as the proof in Theorem 7.

Next we derive a lower bound on $w \cdot w^{k+1}$. Denote by $B' \subseteq B$ the set of nodes in the beam such that $\delta_1 > w \cdot (f(v^*) - f(v)) \geq -\delta_2$ and let $C' = P_{i,j} \cap C$. By the definition of beam margin, we have $|B'| < b'$.

$$\begin{aligned} w \cdot w^{k+1} &= w \cdot w^k + w \cdot \left(\frac{\sum_{v* \in C'} f(v^*)}{|C'|} - \frac{\sum_{v \in B} f(v)}{b}\right) \\ &= w \cdot w^k + w \cdot \sum_{v \in B - B'} \frac{\frac{\sum_{v* \in C'} f(v^*)}{|C'|} - f(v)}{b} \\ &\quad + w \cdot \sum_{v \in B'} \frac{\frac{\sum_{v* \in C'} f(v^*)}{|C'|} - f(v)}{b} \\ &\geq w \cdot w^k + \frac{(b - b')\delta_1}{b} - \frac{b'\delta_2}{b} \end{aligned}$$

By induction, we get that $w \cdot w^{k+1} \geq k \frac{(b-b')\delta_1 - b'\delta_2}{b}$. Combining this result with the above upper bound on $\|w^{k+1}\|$ and the fact that $\|w\| = 1$ we get that $1 \geq \frac{w \cdot w^{k+1}}{\|w\| \|w^{k+1}\|} \geq \sqrt{k} \frac{\delta_1(b-b') - \delta_2 b'}{bR}$. The mistake bound follows by noting that $b > b^*$ and algebra. $\square$

When there is a positive level margin (i.e. $b' = 0$), the mistake bound reduces to $(R/\delta_1)^2$, which does not depend on the beam width and matches the result for separable data. This is also the behavior when $b >> b'$. When $\delta_1 = \delta_2$ and we use the minimum beam width allowed by the theorem $b = 2b' + 1$, the bound is $((2b' + 1)R/\delta_1)^2$, which is a factor of $(2b' + 1)^2$ larger than when $b >> b'$. This shows that as we increase b (i.e. the amount of search), the mistake bound decreases, suggesting that learning becomes easier. This agrees with the intuition that the more computation we put into search the less we need to learn. Indeed, in the case of exhaustive search no learning is needed at all.

6. Summary

We studied the computational complexity of learning ranking functions for beam search and the convergence of online Perceptron updates. The results identified core tractable and intractable subclasses and clarified convergence issues. We also considered convergence in the case of ambiguous training data, giving a result that highlights the trade-off between the amount of allowed search and the difficulty of the resulting learning problem.

Acknowledgments

We thank the reviewers and Hal Daume III for useful comments/discussions. This work was supported by NSF grant IIS-0307592 and DARPA contract FA8750-05-2-0249.

References

Agarwal, S., & Roth, D. (2005). Learnability of bipartite ranking functions. *COLT-05.*

Daume III, H., & Marcu, D. (2005). Learning as search optimization: Approximate large margin methods for structured prediction. *ICML-05.*

Dietterich, T., Lathrop, R., & Lozano-Perez, T. (1997). Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, *89*, 31–71.

Garey, M. R., & Johnson, D. S. (Eds.). (1979). *Computers and intractability: A guide to the theory of np-completeness.* New York: W. H. Freeman and Company.

Hoffgen, K.-U., Simon, H.-U., & Horn, K. S. V. (1995). Robust trainability of single neurons. *Journal of Computer and System Sciences*, *50*, 114–125.

Novikoff, A. (1962). On convergence proofs on perceptrons. *Symposium on the Mathematical Theory of Automata* (pp. 615–622).

Xu, Y., Fern, A., & Yoon, S. (2007). Discriminative learning of beam-search heuristics for planning. *IJCAI-07.*