
Inference & Learning with Linear Constraints

Vasin Punyakanok **Dan Roth** **Wen-tau Yih** **Dav Zimak**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{punyakan,danr,yih,davzimak}@uiuc.edu

Abstract

We present a discriminatory learning framework for the problem of assigning globally optimal values to a set of variables with complex and expressive dependencies among them. The problem is modeled as an integer linear program (ILP) where the cost values associated with the variables are represented and trained as linear classifiers. The framework unifies and extends several existing discriminatory approaches; most importantly, it supports more complex dependencies among variables than existing ones. This presentation concentrates on the benefits of the additional expressivity and on comparing different training paradigms – with and without global feedback – in the context of semantic role labeling.

1 Introduction

Making decisions in real world problems often involves assigning values to sets of variables where complex and expressive dependencies can influence, or even dictate, what assignments are possible. This is common in natural language tasks where predicting pos tags of words is governed by the constraint that no three consecutive words are verbs; or that certain verbs must have, somewhere in the sentence, three arguments of specific semantic types. Similar examples exist in scene interpretation tasks where predictions must respect some constraints that could arise from the nature of the data or task specific conditions.

Efficient solutions have been given to problems of this kind under some restrictions. One class of solutions decouples the process of learning estimators to variables' values, from that of the inference process that is applied later to derive a global assignment to the set of variables of interest. A second class of solutions incorporates the dependencies among the variables into the learning process, and directly induces estimators that yield a (good) global assignment. Significant amount of the work in these directions in recent years has focused on developing discriminatory models. Discriminative HMM, Conditional models [9, 8] and a large number of dynamic programming based schemes used in the context of sequential predictions (e.g., Named Entities [13]) fall into the first category. Conditional Random Fields [7], Collins' perceptron scheme [2, 1] and other and discriminative learning with Markov assumptions [12] are in the second.

This paper presents a discriminatory learning framework for the problem of assigning globally optimal values to a set of variables with complex and expressive dependencies among them. The inference process of assigning globally optimal values to mutually dependent variables is formalized as an optimization problem and is solved as an integer linear programming (ILP)

problem, via one of the strong commercially available packages available for this ¹. Cost functions for the variables estimators can be learned via any of the known linear classifiers with strong generalization abilities, such as Winnow or Perceptron.

The framework presented unifies the two approaches mentioned above in that it supports a separate process of learning variables' estimators (classifiers) followed by an inference process, as well as incorporating the constraints among the variables into the training process. Additionally, the inference based training paradigm allows us to train in a conservative manner; namely, train only a subset of the classifiers, determined by propagating information via the constraints. As we show, these two training paradigms have advantages in different situations, quantified in terms of the quality of the variables' estimators. Importantly, unlike previously studied approaches, our inference paradigm is not restricted to allow only sequential constraints among the variables. This is significant, as we show, in allowing learning the variables' estimators (classifiers) in terms of complex features, but also allows us to incorporate expressive constraints on the outcomes of the classifiers, and even include decision time constraints, not known to the system during learning.

This paper points to two experimental studies with this framework and then concentrates on presenting its key ingredients and what distinguishes it from previous, less expressive approaches. Specifically, we show experiments that exhibit the benefit from the additional expressivity, as well as those that shed light on the relative advantages of different training paradigms. We end by showing that this framework can be used to learn in a hierarchical setting, where different level of feedback might be provided to the classifiers.

1.1 Target Applications

We consider two natural language applications to motivate the use of our paradigm. ILP has been used successfully on these to achieve state of the art performance [10, 11].

Semantic role labeling (SRL) is believed to be an important task toward natural language understanding, and has immediate applications in tasks such Information Extraction and Question Answering. The goal is to identify, for each verb in the sentence, all the constituents which fill a semantic role, and determine their argument types, such as *Agent*, *Patient*, *Instrument*, as well as adjuncts such as *Locative*, *Temporal*, *Manner*, etc. For example, given a sentence “I left my pearls to my daughter-in-law in my will”, the goal is to identify different arguments of the verb *left* which yields the output:

[_{A0} I] [_V left] [_{A1} my pearls] [_{A2} to my daughter-in-law] [_{AM-LOC} in my will].

Here A0 represents *leaver*, A1 represents *thing left*, A2 represents *benefactor*, AM-LOC is an adjunct indicating the location of the action, and V determines the verb.

We model the problem as a multiclass classification problem that maps constituent candidates to one of 45 different types [6, 1]. However, local multiclass decisions are insufficient. Structural constraints are necessary to ensure, e.g., that no arguments can overlap or embed each other. While it is possible to incorporate these constraints say, with HMM, additional constraints – such as that a sentence can have at most one A0 – may be harder to incorporate into a fixed representation. Linguistic constraints are often more difficult.

Entity and Relation Recognition is the problem of recognizing the *kill (KFJ, Oswald)* relation in the sentence “tJ. V. Oswald was murdered at JFK after his assassin, R. U. KFJ...” This task requires making local decisions of two distinct, but mutually supporting types — named entities and relations. For example, knowing that Oswald and KFJ are *people*, and JFK is a *location* supports identifying that a *kill* relation is described in the sentence. In our model, a collection of local classifiers are learned to identify entities and relations. Then, at decision time, we incorporate global constraints that restrict what entities and relations can coexist, e.g., that the *kill* relation must take a person as the first argument.

¹ E.g., for the Entity-Relation problem mentioned later, the ILP problem has hundreds of variables and constraints; our formulation allows the processing of 20 sentences a second.

Both of applications leverage information that can be learned from local information to aid in finding a coherent and consistent global classification. Even in the case that the local classifiers are naive, it is possible to add enough information from structural and linguistic constraints to infer the correct classification.

2 Using Integer Linear Programming for Inference

Given an assignment $\mathbf{x} \in \mathcal{X}^{n_x}$ to a collection of input variables, $\mathbf{X} = (X_1, \dots, X_{n_x})$, our task involves identifying the optimal assignment $\mathbf{y} \in \mathcal{Y}^{n_y}$ to a collection of the output $\mathbf{Y} = (Y_1, \dots, Y_{n_y})$. In order to do so, we utilize a collection of classifiers that locally output the score of each possible value of each individual output variable y_i based on the given input, and possibly with the knowledge of the assignment to the rest of output variables. Additional constraints among the output variables might also exist. Therefore, an effective inference procedure is needed to infer the globally optimal output using the classifiers and the given constraints. This section explains formally the notion of classifiers and the inference along with the use of integer linear programming (ILP) for the inference.

2.1 Classifications and Inferences

Our scheme utilizes multiclass classifiers for each output variable, rather than a global classifier. We consider the case when the multiclass classifier consists of a set of functions, $\{f_y(\mathbf{x}, t)\}_{y \in \mathcal{Y}}$, where $f_y : \mathcal{X}^{n_x} \times \{1, \dots, n_y\} \rightarrow \mathbb{R}$. Each function represents a *score* of that output variable Y_t takes value y , given \mathbf{x} score $(\mathbf{x}, y, t) = f_y(\mathbf{x}, t)$. More expressive classifiers may depend on the output. Let $\{f_y(\mathbf{x}, \mathbf{y}^{\bar{t}}, t)\}_{y \in \mathcal{Y}}$ be such a set of classifiers, where $\mathbf{y}^{\bar{t}}$ are labels of all output except for y_t , and $f_y : \mathcal{X}^{n_x} \times \mathcal{Y}^{n_y-1} \times \{1, \dots, n_y\} \rightarrow \mathbb{R}$. Then, score $(\mathbf{x}, \mathbf{y}^{\bar{t}}, y, t) = f_y(\mathbf{x}, \mathbf{y}^{\bar{t}}, t)$, represents the score that $Y_t = y$ given labels to (potentially) all of the output variables except for Y_t . Typically, $\mathbf{y}^{\bar{t}}$ is a very restricted subset of \mathbf{y} ; for example, if each variable represent a value in any particular time step in a Markov process, it depends only on the variable at time $t - 1$.

If classifiers depend on output variables as well as input, they are called *interacting*, otherwise they are called *noninteracting*. For the case of interacting classifiers, the goal is to find an assignment \mathbf{y} that maximize the total expected score,

$$\text{score}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n_y} \text{score}(\mathbf{x}, \mathbf{y}^{\bar{t}}, t) = \sum_{t=1}^{n_y} f_y(\mathbf{x}, \mathbf{y}^{\bar{t}}, t). \quad (1)$$

Inference is the task of determining an optimal assignment \mathbf{y} given an assignment \mathbf{x} . Typically, we write this as $\hat{\mathbf{y}} = \text{Inference}(\mathbf{x})$, and it is the *global* classification task. Typically, the number of input and output variables, n_x and n_y , are not fixed, but are given for each example. We sometimes write \mathcal{X}^* (and \mathcal{Y}^*) to represent the set of arbitrary, but fixed, sized sets of input (and output) variables. With the classifiers, the inference is to defined as:

$$\hat{\mathbf{y}} = \text{Inference}(\mathbf{x}) = \underset{\mathbf{y}' \in \mathcal{Y}^{n_y}}{\text{argmax}} \sum_{t=1}^{n_y} f_y(\mathbf{x}, \mathbf{y}'^{\bar{t}}, t). \quad (2)$$

2.2 Integer Linear Programming

Fortunately, integer linear programming (ILP) provides a general solution to the inference problem we have. First, we review the basic definition of integer linear programming (ILP). Given a cost vector $\mathbf{p} \in \mathbb{R}^d$, a set of variables, $\mathbf{u} = (u_1, \dots, u_d) \in \{0, 1\}^d$ and a cost matrix $\mathbf{C} \in \mathbb{R}^c \times \mathbb{R}^d$, with c constraints and d (binary) variables, the ILP solution, $\hat{\mathbf{u}}$, is the vector that maximizes the cost function, $\mathbf{p} \cdot \mathbf{u}$,

$$\hat{\mathbf{u}} = \underset{\mathbf{u} \in \{0, 1\}^d}{\text{argmax}} \mathbf{p} \cdot \mathbf{u}, \quad \text{subject to} \quad \mathbf{C}\mathbf{u} \geq 0.$$

Many interesting constraints are linear. In fact any boolean function can be represented as a set of linear constraints since it can be represented as a CNF formula in which each disjunction can be directly translated into a linear constraint.

Typically, \mathbf{u} will represent *indicator* variables to indicate when a condition is held—for example, when $Y_i = y$ or when $Y_i = y$ and $X_j = x$. Inference is the process of finding $\hat{\mathbf{u}}$, since once we know it, there will always be a fixed transformation to $\hat{\mathbf{y}}$. For simplicity, $\mathcal{C}(\{0, 1\}^d)$ and $\mathcal{C}(\mathcal{Y}^{n_y})$ implicitly represent the sets $\{0, 1\}^d$ and \mathcal{Y}^{n_y} , respectively, constrained to $\mathbf{C}\mathbf{u} \geq 0$ for the indicator variables and their corresponding output variables. By incorporating global information into the constraints, ILP can find the *optimal* consistent assignment. In what follows, the values of $f_y(\mathbf{x}, \mathbf{y}^{\bar{t}}, t)$ are cached and there is no direct dependence on \mathbf{y} in the optimization.

$$\hat{\mathbf{I}} = \operatorname{argmax}_{\mathbf{I} \in \{0, 1\}^{|\mathcal{Y}|t}} \sum_{t=1}^{n_y} \sum_{y=1}^{|\mathcal{Y}|} f_y(\mathbf{x}, \mathbf{y}^{\bar{t}}, t) u_{\mathbf{Y}=\mathbf{y}} \quad (3)$$

$$\text{subject to} \quad \sum_{y \in \mathcal{Y}} I_{Y_t=y} = 1, \quad \forall Y_t \in \mathbf{Y} \quad (4)$$

$$\sum_{(Y_t=y_t) \in (\mathbf{Y}=\mathbf{y})} I_{Y_t=y_t} - u_{\mathbf{Y}=\mathbf{y}} \leq n_y - 1, \quad \forall (Y_t = y_t) \in (\mathbf{Y} = \mathbf{y}) \quad (5)$$

$$\sum_{(Y_t=y_t) \in (\mathbf{Y}=\mathbf{y})} I_{Y_t=y_t} - n_y u_{\mathbf{Y}=\mathbf{y}} \geq 0, \quad \forall (Y_t = y_t) \in (\mathbf{Y} = \mathbf{y}) \quad (6)$$

Other Constraints...,

Constraint 4 ensures that each variable Y_t takes exactly one label. Constraint 5 and 6 together make sure that indicator variables \mathbf{I} and \mathbf{u} correspond. For any \mathbf{I} that satisfies the constraints, $I_{Y_t=y} = 1$ if and only if $Y_t = y$ in the assignment induced by \mathbf{I} . When using additional constraints on the output, it is possible (and even likely) that the correct prediction of the entire sequence will not agree with the individual classifiers.

2.2.1 Classifiers with Linear Representation

If the classifiers are linear, $f_y(\mathbf{x}, \mathbf{y}^{\bar{t}}, t) = \alpha^y \cdot \Phi^y(\mathbf{x}, \mathbf{y}^{\bar{t}}, t)$, then, the maximum cost is the maximum over a single linear function. In particular, Equation 3 can be rewritten as,

$$\begin{aligned} \max_{\mathbf{I} \in \mathcal{C}(\{0, 1\}^{|\mathcal{Y}|t})} \sum_{t=1}^T \sum_{y=1}^{|\mathcal{Y}|} f_y(\mathbf{x}, \mathbf{y}^{\bar{t}}, t) I_{y,t} &= \max_{\mathbf{y} \in \mathcal{C}(\mathcal{Y}^T)} \sum_{y=1}^{|\mathcal{Y}|} \alpha^y \sum_{t=1}^T \Phi^y(\mathbf{x}, \mathbf{y}^{\bar{t}}, t) I_{y_t=y} \\ &= \max_{\mathbf{y} \in \mathcal{C}(\mathcal{Y}^T)} \sum_{y=1}^{|\mathcal{Y}|} \alpha^y \cdot \Phi^y(\mathbf{x}, \mathbf{y}) = \max_{\mathbf{y} \in \mathcal{C}(\mathcal{Y}^T)} \alpha \cdot \Phi(\mathbf{x}, \mathbf{y}), \end{aligned}$$

where $\Phi^y(\mathbf{x}, \mathbf{y})$ is an accumulation over all output variables of features occurring for class y , α is concatenation of the α^y 's, and $\Phi(\mathbf{x}, \mathbf{y})$ is the concatenation of the $\Phi^y(\mathbf{x}, \mathbf{y})$'s. The global assignment is constrained by defining a function $\mathcal{C}(\cdot)$ to represent the valid assignments.

3 Learning

Given the ILP framework for inference, there are several ways to learn the cost function parameters differing in whether or not the inference process is used during training. As we will see, classifiers can be trained both as stand alone functions to score each output variable separately and as the components of a global scoring function. The only essential difference between the two methods is the use of feedback from the global inference process during training – this difference has motivated recent work, however the ILP framework brings it clearly into view (see Section 3.4).

Learning consists of choosing a function $h : \mathcal{X}^* \rightarrow \mathcal{Y}^*$ from some hypothesis space, \mathcal{H} . Typically, the data is supplied as a set $\mathbf{D}^{\mathcal{X}, \mathcal{Y}} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_m, \mathbf{y}_m)\}$ from a distribution $\mathbb{P}^{\mathcal{X}, \mathcal{Y}}$ over $\mathcal{X}^* \times \mathcal{Y}^*$.

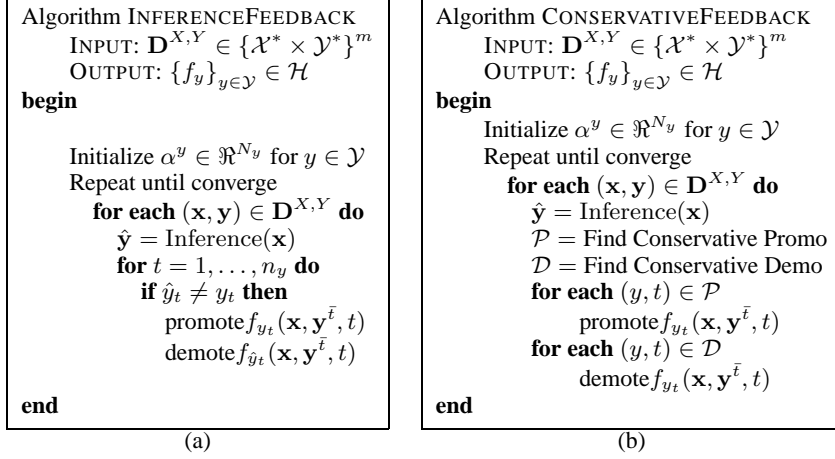


Figure 1: Algorithms for learning with inference feedback.

3.1 Learning Classifiers via Multiclass Learning

Learning stand alone classifiers is perhaps the most straightforward setting. No knowledge of the inference procedure is used. For the sake of presentation and comparison to other methods, the classifiers here are linear functions,

$$f_y(\mathbf{x}, \mathbf{y}^{\bar{t}}, t) = \alpha^y \cdot \Phi^y(\mathbf{x}, \mathbf{y}^{\bar{t}}, t),$$

where $\Phi^y : \mathcal{X}^{n_x} \times \mathcal{Y}^{n_y-1} \times \{1, \dots, n_y\} \rightarrow \{0, 1\}^{N_y}$ is the feature set for class y .

Learning α^y is possible by using simple multiclass learning. Specifically, each example (\mathbf{x}, \mathbf{y}) is transformed into a set of examples, $\{(\Phi^y(\mathbf{x}, \mathbf{y}^{\bar{t}}, t), y_t)\}_{t=1}^T$. Since $y_t \in \mathcal{Y}$, this produces a $|\mathcal{Y}|$ -class multiclass learning problem. Various algorithms can be used here, such as one-versus-rest, where $\alpha^y \cdot \Phi(\mathbf{x}, \mathbf{y}^{\bar{t}}, t) > 0$ if and only if $y = y_t$, or constraint classification, where $\alpha^{y_t} \cdot \Phi(\mathbf{x}, \mathbf{y}^{\bar{t}}, t) > \alpha^y \cdot \Phi(\mathbf{x}, \mathbf{y}^{\bar{t}}, t)$ for all $y \neq y_t$ (see [5] for details and Section 4 for experiments).

3.2 Learning Classifiers via Inference Feedback

We seek to train classifiers so they will produce the correct global classification. To this end, the key difference from Section 3.1 is that here, feedback from the inference process determines which classifiers to train so that together, the classifiers and the inference procedure yield the desired result. As in [2, 1], we train according to a global criteria. The algorithm presented here is an online procedure, where at each step a subset of the classifiers are updated according to inference feedback. Essentially, classifier $f_y(\cdot)$ is promoted only when some variable in the output is misclassified as \hat{y} and its correct label is y —in this case $f_{\hat{y}}(\cdot)$ is demoted also. See figure 1(a) for details. The algorithm is an online procedure, where for each example, we promote classifiers that are under-predicted and demote classifiers that are over-predicted. Any algorithm with appropriate Promote and Demote procedures defined, such as the perceptron and Winnow learning algorithms.

3.3 Conservative Learning via Inference Feedback

It is possible to use the ILP formulation that take the constraints into account in determining which functions to update. Specifically, the constraints convey information about dependencies among the indicator variables (and thus among the classifier functions). For example, constraint $I_{y_1=1} + I_{y_2=1} = 1$ implies that one of y_1 and y_2 must be 1 and the other, 0. Therefore, if the inference process concludes that $(y_1, y_2) = (0, 1)$, but the correct prediction indicates that $(y_1, y_2) = (1, 0)$, there are several options to update. First, one can follow the procedure in

Section 3.2 and demote $f_{Y=0}(\mathbf{x}, \mathbf{y}^1, 1)$ and $f_{Y=1}(\mathbf{x}, \mathbf{y}^2, 2)$ and promote $f_{Y=1}(\mathbf{x}, \mathbf{y}^1, 1)$ and $f_{Y=0}(\mathbf{x}, \mathbf{y}^2, 2)$. On the other hand, if we notice that if $f_{Y=1}(\mathbf{x}, \mathbf{y}^1, 1)$ were a large enough value, then it would force $I_{y_1=1} = 1$ and $I_{y_2=1} = 0$. Therefore, yielding it unnecessary to demote $f_{Y=1}(\mathbf{x}, \mathbf{y}^2, 2)$. In some sense, there are several options of which functions to promote and demote. A *conservative* strategy is one that updates the fewest number of classifiers by finding the minimum size set, $S = \mathcal{P} \cup \mathcal{D} \subseteq \mathbf{Y}$, of variables from \mathbf{Y} to promote ($\in \mathcal{P}$) and demote ($\in \mathcal{D}$).

In practice, we used a greedy approximation algorithm that incrementally grows S . Due to lack of space, we give only a high level description. Let \hat{S} be the set of variables that were misclassified. At each iteration we perform two operations. First, a variable y_t is removed from \hat{S} at random and added to S . Then, the variables in S , along with the correctly classified variables are used to discover dependent variables from the constraint set in the ILP. Namely, any variable that changes value as a result of S being correctly predicted is considered a dependent variable. In the example above, $y_2 = 1$ is dependent on $y_1 = 1$. Any dependent variables are removed from \hat{S} without being added to S since their value will change as a result of updating y_t . Then these two steps are repeated until there are no more variables in \hat{S} . See Figure 1(b) for details.

3.4 Related work

As previously mentioned, training structures by global feedback has been studied in [2, 12, 1], where discriminant training is used to learn a linear global representations corresponding to probabilistic models. Specifically, $\text{score}(\mathbf{x}, \mathbf{y}) = \alpha \cdot \Phi(\mathbf{x}, \mathbf{y})$, where $\Phi(\mathbf{x}, \mathbf{y})$ is a global feature vector and $\alpha \in \mathbb{R}^{N_\phi}$. Inference in this model is the process of finding the assignment, $\hat{\mathbf{y}} \in \mathcal{Y}^*$ with the highest score,

$$\hat{\mathbf{y}} = \text{Inference}(\mathbf{x}) = \underset{\mathbf{y}' \in \mathcal{Y}^{n_y}}{\text{argmax}} \alpha \cdot \Phi(\mathbf{x}, \mathbf{y}'). \quad (7)$$

In these works, inference strictly depends on the assumptions made about the underlying structure. It is impossible to add additional constraints if they would violate these assumptions. By posing inference as an ILP, we open the door to a wide range of underlying structures to hopefully realize their full potential. The resulting optimization function is,

$$\hat{\mathbf{u}} = \underset{\mathbf{u} \in \{0,1\}^N}{\text{argmax}} \sum_{i=1}^N \sum_{t=1}^{n_y} \alpha_{i,t} u_{i,t},$$

with the appropriate constraints defined to ensure that the assignment \mathbf{y} is valid.

In [2], the existence of classifiers for each variable is not assumed. Rather than training a collection of functions, a single classifier is trained using the output of the global prediction. A variant of the perceptron algorithm is used to find a weight vector α such that $\mathbf{y} = \underset{\mathbf{y}' \in \mathcal{Y}^{n_y}}{\text{argmax}} \alpha \cdot \Phi(\mathbf{x}, \mathbf{y}')$ for all examples $(\mathbf{x}, \mathbf{y}) \in \mathbf{D}^{X,Y}$. They show convergence if the data is separable.

3.5 Justification of Learning Algorithms

We have presented a suite of learning algorithms in the ILP framework. here, we briefly discuss why these algorithms work. If it is possible to learn independent classifiers that can correctly represent each of the sub-tasks, then a high performing classifier minimizes expected error of the independent variables. Combining classifiers via a global inference procedure simply searches for the assignment that maximizes the sum of expected errors, and therefore is the optimal strategy if the classifiers accurately represent probabilities [9]. However, in this work, we make no assumption that accurate classification is possible at the local level. Rather, the learning via inference feedback allows for a theoretically justifiable learning algorithm. In the case when perceptron is used, a mistake bound was proven in [2]. The intuition behind why a global classifier could be learned when no consistent local classifiers exist is that the feedback provides a more conservative updating strategy than the local learning methods, and thus is more careful to

	full data set			≥ 5 arguments		
	Precision	Recall	$F_{\beta=1}$	Precision	Recall	$F_{\beta=1}$
w/o Inference	86.95	87.24	87.10	81.26	81.75	81.50
w/ Inference	88.03	88.23	88.13	83.20	83.50	83.35
Relative gain	8.28%	7.76%	7.98%	10.35%	9.59%	10.00%

Table 1: Performance improvement by applying inference: the model used is classification plus inference; the learning algorithm is SNoW with Winnow update rule and a thick separator [3, 4].

make the “right” updates. The final learning algorithm makes even more conservative updates by analyzing the ILP constraints to make the absolute minimal change in the classification function.

This intuition is clear in our experimental results, where learning via inference feedback works best when the learning problem is made more difficult.

4 Experiments

Our paradigm has been successfully applied on several problems, such as simultaneous entity/relation recognition [11] and semantic role labeling [10]. In this section, we show benefits of incorporating general constraints in inference. Furthermore, we present experimental results seeking the answer to an interesting question: when is it better to train with inference feedback.

The task we have for the experiments is *Semantic Role Labeling (SRL)*, which has already been described in Sec. 1.1. We took the data provided in the CoNLL-2004 shared task [1]. For the purposes of this paper, we restrict our focus to sentences that have greater than five arguments, so as to increase the need of global inference. In addition, to simplify the problem, we assume the boundaries of the constituents are given – the task is mainly to assign the argument types.

We first demonstrate the importance of applying inference in this task. Table 1 shows the difference in precision, recall, and $F_{\beta=1}$ with and without inference. When the full data set is used, the relative gain in $F_{\beta=1}$ is 7.98%. When the data consists of only sentences with more than five arguments, the task becomes more difficult and the constraints play a more important role. As a result, the gain is increased to 10.00%. The SRL task is made more difficult by reducing the number of features. The experiments confirm the intuition in Section 3.5 that result from the justification of how inference affects learning. Here, we show that inference feedback during training improves classification only when the learning task is difficult. Local learning done without feedback using constraint classification (**CC-NF-I**) and one-versus-all training (**OVA-NF-I** and **OVA-NF-NI**) outperform the global feedback based method (**I-F**) when the problem is easy. However, with fewer features, the local learning methods can not learn well, and the global feedback improves the performance. The ILP framework makes it possible to make this comparison using the *same* learning algorithm, thus allowing an insightful and new analysis of global feedback in learning. Figure 2 shows the results using Winnow and Perceptron.

5 Discussion

We presented a discriminatory learning framework for of assigning globally optimal values to a set of variables with complex and expressive dependencies among them, discussed learning paradigms and showed the benefits of additional expressivity provided by the framework. To show the generality of the framework, consider the following example, of modeling *Hierarchical Classification*. Often, classification is performed in stages. In hierarchical classification, given input \mathbf{x} , one first computes $\hat{\mathbf{z}}_1 = h_1(\mathbf{x})$, then $\hat{\mathbf{z}}_2 = h_2(\mathbf{x}, \hat{\mathbf{z}}_1)$ and so on. Once the set of intermediate variables is known, then the output can be computed by $\hat{\mathbf{y}} = h_{l+1}(\mathbf{x}, \hat{\mathbf{z}}^1, \dots, \hat{\mathbf{z}}^l)$, where there are l such intermediate levels. To place hierarchical classification in the ILP framework, it is assumed that the transition from level j to level $j + 1$ can be computed by an ILP formulation. Specifically, if $h_j(\cdot)$ can be computed via ILP using the cost function \mathbf{p}_j and constraints \mathbf{C}_j , then

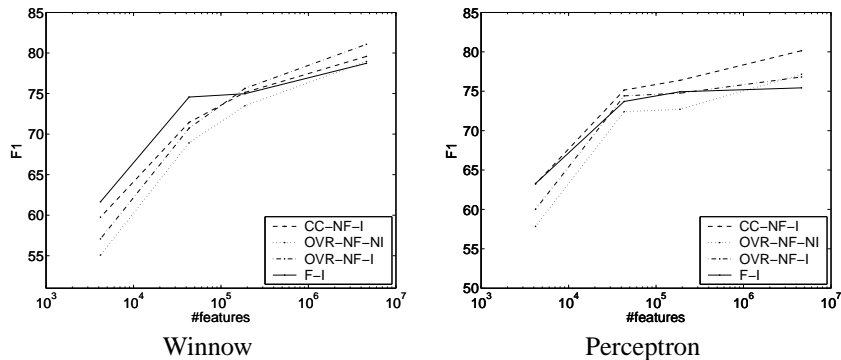


Figure 2: Multiclass learning vs. learning via inference feedback

$\hat{y} = \operatorname{argmax}_{\mathbf{y}} \max_{\mathbf{z}_1, \dots, \mathbf{z}_l} \sum_{j=1}^{l+1} \Gamma_j \mathbf{p}_j \mathbf{u}_j$ subject to, $\mathbf{C}_1 \mathbf{u}_1 \geq 0, \mathbf{C}_2 \mathbf{u}_2 \geq 0, \dots, \mathbf{C}_{l+1} \mathbf{u}_{l+1} \geq 0$, where Γ_j is a constant such that, given $\hat{\mathbf{z}}_1, \dots, \hat{\mathbf{z}}_{j-1}$, any two assignments to $\mathbf{z}_j, \dots, \mathbf{z}_l, \mathbf{y}$ will yield a large change in the cost function (say $> \gamma_j$) if and only if some variable in \mathbf{z}_j changes value. Learning in hierarchical models is a special case of the feedback based learning paradigm presented above. Moreover, the conservative paradigm, where some of the updates are dictated by the constraints, allows a range of update policies, e.g., similar to [1].

Future work in this paradigm will focus both on developing theoretical understanding and justifications, as on further large scale experiments with it.

References

- [1] X. Carreras and L. Màrquez. Online learning via global feedback for phrase recognition. In *Advances in Neural Information Processing Systems 15*, 2003.
- [2] M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP 2002*, 2002.
- [3] A. Grove and D. Roth. Linear concepts and hidden variables. *Machine Learning*, 42(1/2):123–141.
- [4] T. Hang, F. Damerau, , and D. Johnson. Text chunking based on a generalization of winnow. *Journal of Machine Learning Research*, 2:615–637, 2002.
- [5] S. Har-Peled, D. Roth, and D. Zimak. Constraint classification: A new approach to multiclass classification and ranking. In *Advances in Neural Information Processing Systems 15*, 2003.
- [6] P. Kingsbury and M. Palmer. From Treebank to PropBank. In *Proceedings of LREC*, 2002.
- [7] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML-01*, pages 282–289, 2001.
- [8] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of ICML-00*, Stanford, CA, 2000.
- [9] V. Punyakanok and D. Roth. The use of classifiers in sequential inference. In *Advances in Neural Information Processing Systems 13*, 2001.
- [10] V. Punyakanok, D. Roth, W. Yih, and D. Zimak. Semantic role labeling via integer linear programming inference. In *Proceedings of COLING-04*, 2004.
- [11] D. Roth and W. Yih. A linear programming formulation for global inference in natural language tasks. In *Proceedings of CoNLL-2004*, pages 1–8, 2004.
- [12] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems 16*, 2004.
- [13] E. Tjong Kim Sang and F. De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proc. of CoNLL-2003*, pages 142–147, 2003.