Journal of Machine Learning Research 9 (2008) 2803-2846 Submitted 6/08; Published 12/08

Magic Moments for Structured Output Prediction

Elisa Ricci ELISA.RICCI@DIEI.UNIPG.IT
Dept. of Electronic and Information Engineering
University of Perugia
06125 Perugia, Italy

Tijl De Bie TIJL.DEBIE@GMAIL.COM
Dept. of Engineering Mathematics
University of Bristol
Bristol, BS8 1TR, UK

Nello Cristianini NELLO@SUPPORT-VECTOR.NET
Dept. of Engineering Mathematics and Dept. of Computer Science
University of Bristol
Bristol, BS8 1TR, UK

Editor: Michael Collins

Abstract

Most approaches to structured output prediction rely on a hypothesis space of *prediction functions* that compute their output by maximizing a linear *scoring function*. In this paper we present two novel learning algorithms for this hypothesis class, and a statistical analysis of their performance. The methods rely on efficiently computing the first two moments of the scoring function over the output space, and using them to create convex objective functions for training. We report extensive experimental results for sequence alignment, named entity recognition, and RNA secondary structure prediction.

Keywords: structured output prediction, discriminative learning, Z-score, discriminant analysis, PAC bound

1. Introduction

The last few years have seen a growing interest in learning algorithms that operate over structured data: given a set of training input-output pairs, they learn to predict the output corresponding to a previously unseen input, where either the input or the output (or both) are more complex than traditional data types such as vectors.

Examples of such problems abound: learning to align biological sequences, learning to parse strings, learning to translate natural language, learning to find the optimal route in a graph, learning to understand speech, and much more. This problem setting subsumes as a special case the standard regression, binary classification, and multiclass classification problems. In fact in many cases the structured output prediction approach matches practice more closely. However, this broad generality and applicability comes with a number of significant theoretical and practical challenges.

In standard regression the output space is real-valued, and in classification the output space consists of a relatively small unstructured set of labels. In contrast, in structured output prediction the output space is typically massive, containing a rich structure relating the different output values

with each other. Because of this, even the prediction task itself requires a search (or optimization) over the complete output space, which in itself is often nontrivial. A fortiori, the task of learning to predict poses important new challenges in comparison with standard machine learning approaches such as regression and classification.

1.1 Graphical and Grammatical Models for Structured Data

An immediate approach for structured output prediction would be to use a probabilistic model jointly over the input and the output variables. Probabilistic graphical models (PGMs) or stochastic context free grammars (SCFGs) are two examples of techniques that allow one to specify probabilistic models for a variety of inputs and outputs, explicitly encoding the structure that is present. For a given input, the predicted output can then be found as the one that maximizes the a posteriori probability. This way of predicting structured outputs is referred to as maximum a posteriori (MAP) estimation. The learning phase then boils down to modeling the distribution of the joint of input and output data.

However, it is well known that this indirect approach of first modeling the distribution (disregarding the prediction task of interest) and subsequently using MAP estimation for prediction, risks to be suboptimal. Instead a direct discriminative approach is more appropriate, which directly focuses on the prediction task of interest. Such methods, known as discriminative learning algorithms (DLAs), make predictions by optimizing a scoring function over the output space, where this scoring function has not necessarily a probabilistic interpretation.

Recently studied DLAs include maximum entropy Markov models (McCallum et al., 2000), conditional random fields (CRFs) (Lafferty et al., 2001), re-ranking with perceptron (Collins, 2002b), hidden Markov perceptron (HMP) (Collins, 2002a), sequence labeling with boosting (Altun et al., 2003a), maximal margin (MM) algorithms (Altun et al., 2003b; Taskar et al., 2003; Tsochantaridis et al., 2005), Gaussian process models (Altun et al., 2004), and kernel conditional random fields (Lafferty et al., 2004).

Interestingly, both the generative modeling approach and the DLAs mentioned above make use of formally the same hypothesis class of prediction functions. In particular, they all make use of a scoring function that is linear in a set of parameters to score each element of the output space. In the generative approach, this linear function is the log-probability of the joint of the input and output data; in the discriminative approach this can be any linear function. The actual prediction function then selects the output that achieves the highest value of the scoring function (i.e., the highest score). In the generative approach this means that the a posteriori (log)-probability of the output is maximized, such that the MAP estimate is obtained as pointed out above.

1.2 The Contributions of this Paper

In this paper we will adopt the hypothesis space of prediction functions defined as above. The distribution of scores induced by any hypothesis over all possible outputs is a central concept in various approaches, and can be used to compare hypotheses, and hence to train. For example MM approaches (Altun et al., 2003b; Taskar et al., 2003; Tsochantaridis et al., 2005) prescribe to seek hypotheses that make the score of the correct outputs in the training set larger than all incorrect ones (by a certain margin).

We argue that the problem can be better approached by considering the entire distribution of the scores over the output space, and in particular by computing its first two moments. Different

choices of parameters can be assessed by comparing (a function of) those moments. Such an approach would account for all possible output values at once, rather than just the ones with a high score as in the maximum margin approaches. However these moments cannot be computed by brute force enumeration: in all practical cases the output space is far too large to exhaustively traverse it. Nevertheless in this paper we show how the first and second order moments can often be computed efficiently by means of dynamic programming (DP), without explicitly enumerating the output space. We provide specific examples of how these moments can be computed for three types of structured output prediction problems: the sequence alignment problem, sequence labeling, and learning to parse with a context free grammar for RNA secondary structure prediction.

We then present two ways in which these moments can be used to design a convex objective function for a learning algorithm. **The first approach** is the maximization of the Z-score, a common statistical measure of surprise, which is large if the scores of the correct outputs in the training set are significantly different from the scores of all incorrect outputs in the output space. We show that the Z-score is a convex cost function, such that it can be optimized efficiently. **A second approach**—also convex—is reminiscent of Fisher's discriminant analysis (FDA). We call this new algorithm SODA (structured output discriminant analysis) since the optimization criterion is a similar function of the first and second order statistics as in FDA.

We report extensive experimental results for the proposed algorithms applied to three different problem settings: learning to align, sequence labeling, and RNA folding.

Finally we derive learning-theoretic bounds on the performance of these algorithms, showing that the SODA cost function is related to the rank of the correct output among the other outputs and analyzing its statistical stability within the Rademacher framework; additionally, we present a general PAC bound that applies to any algorithm using this hypothesis class.

1.3 Outline of this Paper

The rest of the paper is structured as follows: Section 2 formally introduces the problem of structured output learning and the hypothesis space considered. Section 3 deals with the computation of the first and second order moments of the score distribution through DP. In Section 4 we introduce the two algorithms. In Section 5 we present our experimental results, and in Section 6 we outline learning-theoretical bounds, whose proof is however left for the appendix.

2. Learning to Predict Structured Outputs

We address the general problem of learning a *prediction function* $h : \mathcal{X} \to \mathcal{Y}$, with $\mathcal{Y}$ a potentially highly structured space containing a potentially large number N of elements. The learning is based on a training set of input-output pairs $\mathcal{T} = \{(\mathbf{x}^1, \overline{\mathbf{y}}^1), (\mathbf{x}^2, \overline{\mathbf{y}}^2), \ldots, (\mathbf{x}^\ell, \overline{\mathbf{y}}^\ell)\}$ drawn i.i.d. from some fixed but unknown distribution $P(\mathbf{x}, \mathbf{y})$ over $\mathcal{X} \times \mathcal{Y}$. The inputs and the outputs may be highly structured objects that parameterize sequences, trees or graphs. For example in sequence alignment learning the output variables parameterize the alignment between two sequences, in sequence labeling $\mathbf{y}$ is the label sequence associated to the observed sequence $\mathbf{x}$, and when learning to parse $\mathbf{y}$ represents a parse tree corresponding to a given sequence $\mathbf{x}$.

2.1 Scoring Functions, Prediction Functions, and the Hypothesis Space

As in standard machine learning approaches, we consider learning methods that choose the prediction function from a hypothesis space by minimizing a cost function evaluated on the training data. To establish the type of hypothesis space we will consider, we will rely on the notion *scoring function*, which is a function $s : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that assigns a numerical score $s(\mathbf{x}, \mathbf{y})$ to a pair $(\mathbf{x}, \mathbf{y})$ of input-output variables. Furthermore, we will assume that s is linear in a parameter vector $\theta \in \mathbb{R}^d$:

Definition 1 (Linear scoring function) *A linear scoring function is a function $s_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ defined as:*

$$s_\theta(\boldsymbol{x}, \boldsymbol{y}) = \theta^T \phi(\boldsymbol{x}, \boldsymbol{y}), \tag{1}$$

where the vector $\phi(\boldsymbol{x}, \boldsymbol{y}) = (\phi_1(\boldsymbol{x}, \boldsymbol{y}),\ \phi_2(\boldsymbol{x}, \boldsymbol{y}),\ \ldots,\ \phi_d(\boldsymbol{x}, \boldsymbol{y}))^T$ is defined by a specified set of integer-valued feature functions $\phi_i : \mathcal{X} \times \mathcal{Y} \to [0, C]$ for a fixed upper bound C.

Based on this, we can define prediction functions as considered in this paper as follows:

Definition 2 (Prediction function) *Given a linear scoring function s_θ, we can define a prediction function $h_\theta : \mathcal{X} \to \mathcal{Y}$ as:*

$$h_\theta(\boldsymbol{x}) = \arg\max_{\boldsymbol{y} \in \mathcal{Y}} s_\theta(\boldsymbol{x}, \boldsymbol{y}). \tag{2}$$

This type of prediction function has been used in previous approaches for structured output prediction. For example, when using a discrete-valued PGM to model the joint distribution of the input and output data, the logarithm of the probability distribution is a linear scoring function as defined above. The vector $\phi(\mathbf{x}, \mathbf{y})$ is then the vector of sufficient statistics, and the parameter vector θ corresponds to the logarithms of the clique potentials or conditional probabilities. Then, a MAP estimator corresponds to a prediction function as defined above. Furthermore, note that each feature function ϕ_i that counts a sufficient statistic is either an indicator function, or the sum of an indicator function evaluated on a set of cliques over which the parameter θ_i is reused. Therefore, each of the features must be an integer between 0 and the number of cliques C, as required for linear scoring functions in Definition 1.

Typically, in PGMs the parameters θ would be inferred by Maximum Likelihood. On the contrary in DLAs θ is computed by minimizing criteria that are more directly linked to the prediction performance. Moreover with DLAs richer feature vectors (with features not necessarily associated to clique potentials or conditional probabilities) are allowed to describe more effectively the relation between input and output variables. This means that the score $s_\theta(\mathbf{x}, \mathbf{y})$ looses its interpretation as a log-likelihood function.

In summary, the hypothesis space we consider in this paper is defined as:

$$\mathcal{H} = \{h_\theta : \theta \in \mathbb{R}^d\}. \tag{3}$$

This is a slightly larger hypothesis space as compared to the one considered in PGMs, since the parameters θ are not restricted to represent log probabilities.

While we choose to abandon the probabilistic interpretations, it is often worthwhile to keep the analogy with PGMs in mind: they teach us when the evaluation of the prediction function (2) can be carried out efficiently by means of Viterbi-like algorithms, despite the huge size of the

output space $\mathcal{Y}$. In fact, it is often convenient to define or derive the scoring function starting from a PGM, to ensure that it is easily maximized by a dynamic programming procedure such as the Viterbi algorithm. Subsequently the constraints on the parameters that are meant to guarantee that the scoring function is a log-probability function can be removed, in order to arrive at a hypothesis space of the form (3).

2.2 Ideal Loss Functions

In order to select an appropriate prediction function from the hypothesis space, a cost function needs to be defined. Here we will provide an overview of a few conceptually interesting cost functions, but which are unfortunately hard to optimize. Nevertheless, they can often be approximated as seen from literature, and as we will demonstrate further on.

Consider a *loss function* $\mathcal{L}_\theta$ that maps the input $\mathbf{x}$ and the true training output $\bar{\mathbf{y}}$ to a positive real number $\mathcal{L}_\theta(\mathbf{x},\bar{\mathbf{y}})$, in some way measuring the discrepancy between the prediction $h_\theta(\mathbf{x})$ and $\bar{\mathbf{y}}$. Empirical risk minimization strategies attempt to find the vector $\theta \in \mathbb{R}^d$ such that the *empirical risk*, defined as:

$$\mathcal{R}_\theta(\mathcal{T}) = \frac{1}{\ell}\sum_{i=1}^{\ell} \mathcal{L}_\theta(\mathbf{x}^i,\bar{\mathbf{y}}^i)$$

is minimized, in hopes that this will guarantee that the expected loss $E\{\mathcal{L}_\theta(\mathbf{x},\bar{\mathbf{y}})\}$ is small as well. Often it is beneficial to introduce regularization in order to prevent overfitting to occur, but let us first consider on the empirical risk itself.

Clearly the choice of the loss function is critical, and different choices may be appropriate in different situations. The simplest one is a natural extension of the zero-one loss in binary classification task, defined as:

$$\mathcal{L}_\theta^{ZO}(\mathbf{x},\bar{\mathbf{y}}) \quad = \quad I(h_\theta(\mathbf{x}) \neq \bar{\mathbf{y}})$$

where $I(\cdot)$ is an indicator function. Unfortunately the zero-one loss function is discontinuous and NP-hard to optimize. Therefore algorithms such as CRFs (Lafferty et al., 2001) and MM methods (Altun et al., 2003b) minimize an upper bound on this loss rather than the loss itself, combined with an appropriate regularization term.

However, in structured output prediction, the zero-one loss is quite crude, in the sense that it makes no distinction in the type of mistake that has been made. For example, assume that the outputs $\mathbf{y}$ are sequences of length m, or vectors: $\mathbf{y} = (y_1, y_2, \ldots, y_m)$. In that case, a wrong prediction is likely to be less damaging if it is due to only one or a few incorrectly predicted symbols in the sequence. A better loss function that distinguishes incorrect predictions in this way is the Hamming loss, originally proposed in Taskar et al. (2003) for MM algorithms:

$$\mathcal{L}_\theta^{H}(\mathbf{x},\bar{\mathbf{y}}) \quad = \quad \sum_j I(h_{\theta,j}(\mathbf{x}) \neq \bar{y}_j),$$

counting the number of elements (i.e., the symbols in a sequence, or coordinates in a vector) of the output where a mistake has been made.

However, the Hamming loss is not necessarily a good measure for the severity of an incorrect prediction. For example, certain sentences can be parsed in totally different ways that can all be correct, with a large Hamming distance separating them. Similarly, RNA molecules can have two

totally different stable fold states, both with functional relevance. Therefore, where perfect prediction of the data cannot be achieved using the hypothesis space considered, it could be more useful to measure the fraction of outputs for which the score is ranked higher than for the correct output. This is the main motivation to use what we call the relative ranking (RR) loss:

$$\mathcal{L}_0^{RR}(\mathbf{x},\overline{\mathbf{y}}) = \frac{1}{N}\sum_{j=1}^{N} I(s(\mathbf{x},\overline{\mathbf{y}}) \leq s(\mathbf{x},\mathbf{y}_j)),$$

We refer to this loss as the relative ranking loss, since the rank divided by the total size of the output space N is computed. This loss and related loss functions have been proposed in Freund et al. (1998), Schapire and Singer (1999) and Altun et al. (2003a).

2.3 Playing with Sequences: Labeling, Aligning and Parsing

In order to further clarify the framework of structured output learning we present three typical problems which we will use in the rest of the paper as illustrative examples: sequence labeling learning, sequence alignment learning and parse learning.

2.3.1 SEQUENCE LABELING LEARNING

In sequence labeling tasks a sequence is taken as an input, and the output to be predicted is a sequence that annotates the input sequence, that is, with a symbol corresponding to each symbol in the input sequence. This problem arises in several application such as gene finding or protein structure prediction in computational biology or named entity recognition and part of speech tagging in the natural language processing field. Traditionally a special type of PGM, namely hidden Markov models (HMMs) (Rabiner, 1989), is used in sequence labeling, where the parameters can be learned by maximum likelihood, and subsequently predictions can be made by MAP estimation. In order to derive a DLA for this setting, we will first derive the prediction function corresponding to MAP estimation based on HMMs, and subsequently remove the constraints on the parameters that allow for the probabilistic interpretation of HMMs. Then an appropriate cost function for discrimination can be optimized to select a good parameter setting.

In an HMM (Fig. 1) there is a sequence of observed variables $\mathbf{x} = (x_1, x_2, ..., x_m) \in \mathcal{X}$ which will be the input in the terminology of the paper, along with a sequence of corresponding hidden variables $\mathbf{y} = (y_1, y_2, ..., y_m) \in \mathcal{Y}$, in the present terminology corresponding to the output sequence to be predicted. Each observed symbol x_i is an element of the observed symbol alphabet Σ_x, and the hidden symbols y_i are elements of Σ_y, with $n_o = |\Sigma_x|$ and $n_h = |\Sigma_y|$ the respective alphabet sizes. Therefore the output space is $\mathcal{Y} = {\Sigma_y}^m$, while $\mathcal{X} = {\Sigma_x}^m$. The number of cliques $C = 2m-1$ of the HMM graphical model is equal to the number of edges.

An HMM is defined as a probabilistic model for the joint distribution of the hidden and observed sequence, whereby it is assumed that the probability distribution of each hidden symbol y_k depends solely on the value of the previous symbol in the sequence y_{k-1} (this is the Markov assumption which is quantified by $P(y_k|y_{k-1})$). Furthermore, it is assumed that the probability distribution of the observed symbol x_k depends solely on the value of y_k (quantified by the emission probability distribution $P(x_k|y_k)$). For simplicity, we ignore the probability distribution of the first element of the hidden chain in this exposition. The MAP estimator predicts the hidden sequence $\mathbf{y}$ that is most

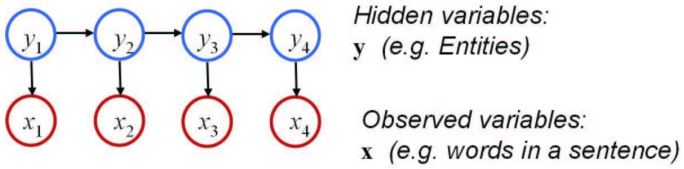


Figure 1: The graph of an HMM with $m = 4$.

likely given the observation sequence $\mathbf{x}$. In formulas:

$$\begin{aligned} h(\mathbf{x}) &= \arg\max_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y}\in\mathcal{Y}} \frac{P(\mathbf{y},\mathbf{x})}{P(\mathbf{x})} = \arg\max_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y})P(\mathbf{x}|\mathbf{y}), \\ &= \arg\max_{\mathbf{y}\in\mathcal{Y}} \prod_{k=2}^{m} P(y_k|y_{k-1}) \prod_{k=1}^{m} P(x_k|y_k), \\ &= \arg\max_{\mathbf{y}\in\mathcal{Y}} \left[\sum_{k=2}^{m} \log P(y_k|y_{k-1}) + \sum_{k=1}^{m} \log P(x_k|y_k) \right], \end{aligned}$$

where we made use of the fact that the argmax of a function is equal to the argmax of its logarithm.

Thus, to fully specify the HMM, one needs to consider all the transition probabilities (denoted t_{ij} for $i, j \in \Sigma_y$ for the transition from symbol i to j), and the emission probabilities (denoted e_{io} for the emission of symbol $o \in \Sigma_x$ by symbol $i \in \Sigma_y$). Using this notation, we can rewrite the prediction function as follows (with $I(\cdot)$ equal to one if the equalities between brackets hold):

$$\begin{aligned} h(\mathbf{x}) = \quad \arg\max_{\mathbf{y}\in\mathcal{Y}} \quad & \sum_{i,j\in\Sigma_y} \log(t_{ij}) \sum_{k=2}^{m} I(y_{k-1} = i, y_k = j) \\ & + \sum_{i\in\Sigma_y, o\in\Sigma_x} \log(e_{io}) \sum_{k=1}^{m} I(y_k = i, x_k = o). \end{aligned}$$

For simplicity of notation let us replace all logarithms of parameters t_{ij} and e_{io} by parameters θ_i summarized in a $d = n_h n_o + n_h^2$ dimensional parameter vector θ. Additionally, let us summarize the corresponding sufficient statistics $\sum_{k=2}^{m} I(y_{k-1} = i, y_k = j)$ and $\sum_{k=1}^{m} I(y_k = i, x_k = o)$ in a corresponding feature vector $\phi(\mathbf{x},\mathbf{y}) = [\phi_1(\mathbf{x},\mathbf{y})\ \phi_2(\mathbf{x},\mathbf{y})\ \ldots\ \phi_d(\mathbf{x},\mathbf{y})]^T$. (Note that these sufficient statistics count the number of occurrences of each specific transition and emission.) Then we can rewrite the prediction function in a linear form as required:

$$h_\theta(\mathbf{x}) = \arg\max_{\mathbf{y}\in\mathcal{Y}} \theta^T \phi(\mathbf{x},\mathbf{y}).$$

This prediction can be evaluated efficiently by means of the Viterbi algorithm. Note that in order to learn the parameters by means of maximum likelihood estimation, constraints are imposed to ensure that they represent log-probabilities. In order to arrive at a DLA that operates in the same setting, it suffices to ignore these constraints, and to minimize an appropriate empirical risk subject to some regularization, as outlined in Section 2.2. Moreover relaxing also the Markov assumption the proposed formulation can be extended to the case of arbitrary features. In general in fact the

vector $\phi(\mathbf{x},\mathbf{y})$ contains not only statistics associated to transition and emission probabilities but also any feature that reflects the properties of the objects represented by the nodes of the HMM. For example in most of the natural language processing tasks, feature vectors also contain information about spelling properties of words. Sometimes also the so-called 'overlapping features' (Lafferty et al., 2001) are employed, which indicate relations between observations and some previous and future labels. Most of DLAs dealing with this task have proceeded in this way (McCallum et al., 2000; Lafferty et al., 2001; Collins, 2002a; Altun et al., 2003a,b; Taskar et al., 2003).

2.3.2 SEQUENCE ALIGNMENT LEARNING

As second case studied, we consider the problem of learning how to align sequences: given as training examples a set of correct pairwise global alignments, find the parameter values that ensure sequences are optimally aligned. This task is also known as inverse parametric sequence alignment problem (IPSAP) and since its introduction in Gusfield et al. (1994), it has been widely studied (Gusfield and Stelling, 1996; Kececioglu and Kim, 2006; Joachims et al., 2005; Pachter and Sturmfels, 2004; Sun et al., 2004).

Consider two strings S_1 and S_2 of lengths n_1 and n_2 respectively. The strings are ordered sequences of symbols $s_i \in \mathcal{S}$, with $\mathcal{S}$ a finite alphabet of size $n_{\mathcal{S}}$. In case of biological applications, for DNA sequences the alphabet contains the symbols associated with nucleotides ($\mathcal{S} = \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}$), while for amino acids sequences the alphabet is $\mathcal{S} = \{\mathtt{A},\mathtt{R},\mathtt{N},\mathtt{D},\mathtt{C},\mathtt{Q},\mathtt{E},\mathtt{G},\mathtt{H},\mathtt{I},\mathtt{L},\mathtt{K},\mathtt{M},\mathtt{F},\mathtt{P},\mathtt{S},\mathtt{T},\mathtt{W},\mathtt{Y},\mathtt{V}\}$.

An alignment of two strings S_1 and S_2 of lengths n_1 and n_2 is defined as a pair of sequences T_1 and T_2 of equal length $n \geq n_1, n_2$ that are obtained by taking S_1 and S_2 respectively and inserting symbols $-$ at various locations in order to arrive at strings of length n. Two symbols in T_1 and T_2 are said to correspond if they occur at the same location in the respective string. If corresponding symbols are equal, this is called a *match*. If they are not equal, this is a *mismatch*. If one of the symbols is a $-$, this is called a *gap*.

With each possible match, mismatch or gap a score is attached. To quantify these scores, three score parameters can be used: one for matches (θ_m), one for mismatches (θ_s), and one for gaps (θ_g). In analogy with the notation in this paper, the pair of given sequences S_1 and S_2 represent the input variable $\mathbf{x}$ while their alignment is the output $\mathbf{y}$. The score of the global alignment is defined as the sum of this score over the length of T_1 and T_2, that is, as a linear function of the alignment parameters:

$$s_\theta(\mathbf{x},\mathbf{y}) = \theta^T \phi(\mathbf{x},\mathbf{y}) = \theta_m m + \theta_s s + \theta_g g$$

where $\phi(\mathbf{x},\mathbf{y}) = [m\, s\, g]^T$ and m, s and g represent the number of matches, mismatches and gaps in the alignment. Fig. 2 depicts a pairwise alignment between two sequences and the associated path in the alignment graph. The number N of all possible alignments between S_1 and S_2 is clearly exponential in the size of the two strings. However, an efficient DP algorithm for computing the alignment with maximal score $\bar{\mathbf{y}}$ is known in literature: the Needleman-Wunsch algorithm (Needleman, 1970).

The scoring models presented above consider a local form of gap penalty: the gap penalty is fixed independently of the other gaps in the alignment. However for biological reasons it is often preferable to consider an affine function for gap penalties, that is to assign different costs if the gap starts (gap opening penalty θ_o) in a given position or if it continues (gap extension penalty θ_e). Then the score of an alignment is:

$$s_\theta(\mathbf{x},\mathbf{y}) = \theta_m m + \theta_s s + \theta_o o + \theta_e e$$

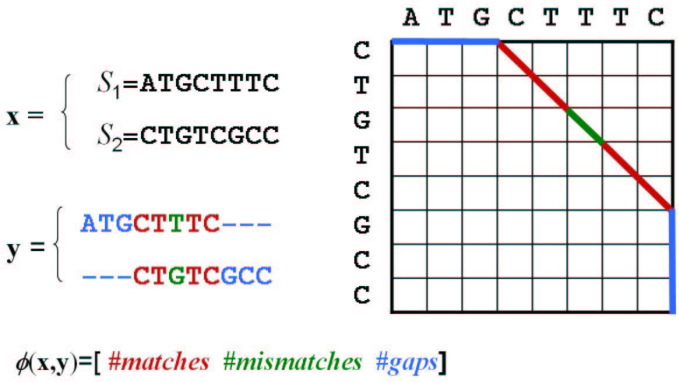


Figure 2: An alignment $\mathbf{y}$ between two sequences S_1 and S_2 can be represented by a path in the alignment graph.

where m, s, o and e represent the number of match, mismatch, gap openings and gap extensions respectively and θ_m, θ_s, θ_o, θ_e are the associated costs. As before we can define the vectors $\theta = [\theta_m\ \theta_s\ \theta_o\ \theta_e]^T$ and $\phi(\mathbf{x},\mathbf{y}) = [m\ s\ o\ e]^T$. Therefore the score is still a linear function of the parameters and the prediction can be computed by a DP algorithm.

More often a different model is considered where a (symmetric) scoring matrix specifies different score values for each possible pair of symbols. In general there are $d = \frac{n_S(n_S+1)}{2}$ different parameters in θ associated with the symbols of the alphabet plus two additional ones corresponding to the gap penalties. This means that to align sequences of amino acids we have 210 parameters to determine plus other 2 parameters for gap opening and gap extension. We denote with z_{jk} the number of pairs where a symbol of T_1 is j and it corresponds to a symbol k in T_2. Again the score is a linear function of the parameters:

$$s_\theta(\mathbf{x},\mathbf{y}) = \sum_{j \geq k} \theta_{jk} z_{jk} + \theta_o o + \theta_e e$$

and the optimal alignment is computed by the Needleman-Wunsch algorithm.

2.3.3 LEARNING TO PARSE

In learning to parse the input $\mathbf{x}$ is given by a sequence, and the output is given by its associated parse tree according to a context free grammar. Usually weighted context-free grammars (WCFGs) (Manning and Schetze, 1999) are used to approach this problem. Learning to parse has been already studied as a particular instantiation of structured output learning, both in natural language processing applications (Tsochantaridis et al., 2005; Taskar et al., 2004) and in computational biology for RNA secondary structure alignment (Sato and Sakakibara, 2005) and prediction (Do et al., 2006). In this paper we consider the latter and we use WCFGs to model the structure of RNA sequences. Two examples of RNA secondary structure for two sequences are shown in Fig. 3.

A WCFG is defined as five tuples $(\Upsilon, \Sigma_x, R, S, \theta)$, where $\Upsilon = \{\Upsilon_1, \ldots, \Upsilon_{|\Upsilon|}\}$ is a set of nonterminals, $\Sigma_x = \{X_1, \ldots, X_{|\Sigma_x|}\}$ is a set of terminals, $R = \{\Upsilon_i \rightarrow \alpha | \Upsilon_i \in \Upsilon, \alpha \in (\Upsilon \cup \Sigma_x)^*\}$ is a set of rules,

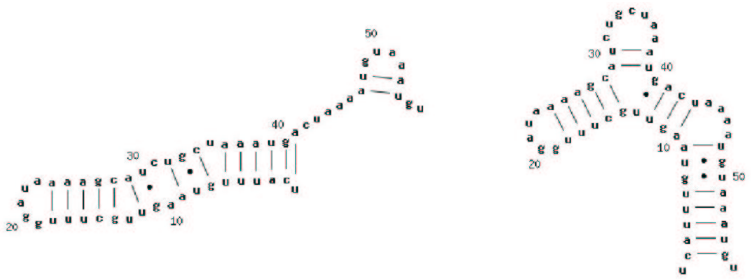


Figure 3: Two examples of RNA secondary structures for two sequences of the Rfam database (Griffiths-Jones et al., 2003).

$S \in \Upsilon$ is the starting symbol, and θ is a set of weights. We use rules of the forms $\Upsilon_i \to X$, $\Upsilon_i \to \Upsilon_j \Upsilon_k$, $\Upsilon_i \to X \Upsilon_j X'$, and $\Upsilon_i \to \Upsilon_{j'}$ ($j' > i$). R is also indexed by an ordering $\{r_1, \ldots, r_{|R|}\}$ and $d = |R|$. Each node in the parse tree $\mathbf{y}$ corresponds to a grammar rule and each weight $\theta_i \in \theta$ is associated with a rule $r_i \in R$. Given a sequence $\mathbf{x}$ and an associated parse tree $\mathbf{y}$ we can define a feature vector $\phi(\mathbf{x}, \mathbf{y})$ which contains a count of the number of occurrences of each of the rules in the parse tree $\mathbf{y}$. Given a parameter vector θ, the prediction function $h_\theta(\mathbf{x})$ is computed by finding the best parse tree. For SCFGs, this can be done efficiently with the Cocke-Younger-Kasami (CYK) algorithm (Younger, 1967).

3. Computing the Moments of the Scoring Function

An interesting corollary of the proposed structured output approach based on linear scoring functions is that certain statistics of the score $s(\mathbf{x}, \mathbf{y})$ can be expressed as function of the parameter vector θ. More specifically given an observed vector $\mathbf{x}$, we can consider the first order moment or mean $M_{1,\theta}(\mathbf{x})$ and the centered second order moment or covariance $M_{2,\theta}(\mathbf{x})$ of the scores along all possible N output variables $\mathbf{y}_j$. It is straightforward to see that $M_{1,\theta}(\mathbf{x})$ is a linear function of θ, that is,

$$\begin{aligned} M_{1,\theta}(\mathbf{x}) &\triangleq \frac{1}{N}\sum_{j=1}^{N} s_\theta(\mathbf{x}, \mathbf{y}_j) \\ &= \theta^T \frac{1}{N}\sum_{j=1}^{N} \phi(\mathbf{x}, \mathbf{y}_j) \\ &= \theta^T \mu \end{aligned}$$

with $\mu = [\mu_1 \ldots \mu_d]^T = \frac{1}{N}\sum_{j=1}^{N} \phi(\mathbf{x}, \mathbf{y}_j)$. Similarly, for the covariance:

$$\begin{aligned} M_{2,\theta}(\mathbf{x}) &\triangleq \frac{1}{N}\sum_{j=1}^{N} (s_\theta(\mathbf{x}, \mathbf{y}_j) - M_{1,\theta}(\mathbf{x}))^2 \\ &= \theta^T \left(\frac{1}{N}\sum_{j=1}^{N} (\phi(\mathbf{x}, \mathbf{y}_j) - \mu)(\phi(\mathbf{x}, \mathbf{y}_j) - \mu)^T \right) \theta \end{aligned}$$

$$= \quad \theta^T C \theta.$$

The matrix C is a matrix with elements:

$$\begin{aligned} c_{pq} &= \frac{1}{N}\sum_{j=1}^{N}(\phi_p(\mathbf{x},\mathbf{y}_j)-\mu_p)(\phi_q(\mathbf{x},\mathbf{y}_j)-\mu_q) \\ &= \frac{1}{N}\sum_{j=1}^{N}\left(\phi_p(\mathbf{x},\mathbf{y}_j)\phi_q(\mathbf{x},\mathbf{y}_j)\right)-\mu_p\mu_q = \nu_{pq}-\mu_p\mu_q \end{aligned} \tag{4}$$

where $1 \leq p,q \leq d$.

3.1 Magic Moments

It should be clear that in practical structured output learning problems the number N of possible output vectors associated to a given input $\mathbf{x}$ can be massive. At first sight, this leaves little hope that the above sums can ever be computed for realistic problems. However, it turns out that the same ideas that allow one to perform inference in PGMs allow one to compute these sums efficiently using DP, be it with somewhat more complicated recursions.

The underlying ideas to derive the recursions for μ are based on the commutativity of the semiring that is used in the Viterbi (or more generally the max-product and related algorithms) in PGMs. In particular, this recursion is used in various forms:

$$E\left\{\sum_{i=1}^{k} a_i\right\} \quad = \quad E\left\{\sum_{i=1}^{k-1} a_i\right\} + E\left\{a_k\right\},$$

where the expectations are jointly over independent random variables a_i. For the recursions for the second order moment (which can be used to compute the centered second order moment as shown in (4)), the following recursive expression is applied in different variations:

$$E\left\{\left(\sum_{i=1}^{k} a_i\right)^2\right\} \quad = \quad E\left\{\left(\sum_{i=1}^{k-1} a_i\right)^2\right\} + 2E\left\{a_k \sum_{i=1}^{k-1} a_i\right\} + E\left\{a_k^2\right\},$$

where again the expectations are jointly over independent random variables a_i. Note that the middle term on the right hand side is computed by previous iterations for the first order moment. For concreteness, we will now consider separately the three illustrative scenarios introduced above.

3.2 Sequence Labeling Learning

Given a fixed input sequence $\mathbf{x}$, we show here for the sequence labeling example that the elements of μ and C can be computed exactly and efficiently by dynamic programming routines.

We first consider the vector μ and construct it in a way that the first $n_h n_o$ elements contain the mean values associated with the emission probabilities and the remaining n_h^2 elements correspond to transition probabilities. Each value of μ can be determined by Algorithm 1.

In the emission part for each element a $n_h \times m$ dynamic programming table μ^e_{pq} is considered. The index p denotes the hidden state ($1 \leq p \leq n_h$) and q refers to the observation ($1 \leq q \leq n_o$). For example the first component of μ corresponds to the DP table μ^e_{11}. In practice each cell of the DP

table correspond to a node of the HMM trellis. At the same time another $n_h \times m$ DP table, denoted by π, is considered and filled in a way that each element $\pi(i,j)$ contains the number of all possible paths in the HMM trellis terminating at position (i,j). Then a recursive relation is considered to compute each element $\mu^e_{pq}(i,j)$, $\forall 1 \leq j \leq m$, $\forall 1 \leq i \leq n_h$. Basically at step (i,j) the mean value $\mu^e_{pq}(i,j)$ is given summing the occurrences of emission probabilities e_{pq} at the previous steps (e.g., $\sum_i \mu^e_{pq}(i,j-1)\pi(i,j-1)$) with the number of paths in the previous steps (if the current observation x_j is q and the current state y_j is p) and dividing this quantity by $\pi(i,j)$.

In a similar way the mean values associated to the transition probabilities are computed. Dynamic programming tables μ^t_{pz}, $1 \leq p,z \leq n_h$ are filled with recursive formulas in Algorithm 4 in appendix E.

Analogously the elements of the covariance matrix C can be obtained. We have five sets of values: variances of emission probabilities (c^e_{pq}, $1 \leq p \leq n_h, 1 \leq q \leq n_o$), variances of transition probabilities (c^t_{pz}, $1 \leq p,z \leq n_h$), covariances of emission probabilities ($c^e_{pqp'q'}$, $1 \leq p,p' \leq n_h, 1 \leq q,q' \leq n_o$), covariances of transition probabilities ($c^t_{pzp'z'}$, $1 \leq p,p',z,z' \leq n_h$) and mixed covariances ($c^{et}_{pqp'z}$, $1 \leq p,p',z \leq n_h, 1 \leq q \leq n_o$). To determine each of them we consider (4) and we compute the values v^e_{pq}, v^t_{pz}, $v^e_{pqp'q'}$, $v^t_{pzp'z'}$ and $v^{et}_{pqp'z}$ since the mean values are already known. This computation is again performed following Algorithm 1 but with recursive relations given in Algorithm 4, in appendix E (the number 5, 11, 12 in Algorithm 4 are meant to indicate the lines of Algorithm 1 where the formulas must be inserted).

Algorithm 1 Computation of μ^e_{pq} for sequence labeling learning

Input: $\mathbf{x} = (x_1, x_2, ..., x_m)$, p, q.

for $i = 1$ **to** n_h
    $\pi(i,1) := 1$
    if $q = x_1 \wedge p = i$, **then** $\mu^e_{pq}(i,1) := 1$
end
for $j = 2$ **to** m
    for $i = 1$ **to** n_h
        $M := 0$
        $\pi(i,j) := \sum_i \pi(i,j-1)$
        if $q = x_j \wedge p = i$, **then** $M := 1$
        $\mu^e_{pq}(i,j) := \frac{\sum_i (\mu^e_{pq}(i,j-1)+M)\pi(i,j-1)}{\pi(i,j)}$
    end
end

Output: $\frac{\sum_i \mu^e_{pq}(i,m)\pi(i,m)}{\sum_i \pi(i,m)}$

3.2.1 COMPUTATIONAL COST ANALYSIS

At first sight the calculation of μ and C requires running a DP algorithm like Algorithm 1 d times for μ and d^2 times for C. Hence the overall computational cost seems to depend strongly on d. However, most of the DP routines are redundant since many cells of μ and C have the same values. In fact, the following can be shown:

Proposition 3 *The number of dynamic programming routines required to calculate* μ and C *increases linearly with the size of the observation alphabet.*

An outline of proof can be found in appendix A.

Algorithms 1 and 4 assume that the HMM is 'fully connected', that is, transitions are allowed from and to any possible hidden states and every symbol can be emitted in every state. However, this condition is often not satisfied in practical applications. We should point out that their adaptation for such situations is straightforward and involves computing only sums that correspond to allowed paths in the DP table. In this case the number of distinct parameters as well as the computational cost increases with respect to complete models. However this effect may be offset by the fact that each DP becomes less time consuming. Furthermore the mean and the covariance values associated to transition probabilities are independent from observations. To calculate them a closed form expression can be used without the need of running any DP routine.

Moreover usually in most applications the size of the observation alphabet (for example the size of the dictionary in a natural language processing system) is very large while the sequences to be labeled are short. This means that the number of distinct observations in each sequence $\mathbf{x}$ is much lower than n_o. In such cases the number of different values in μ and C scales linearly with it.

We point out that the proposed algorithm can be easily extended to the case of arbitrary features in the vector $\phi(\mathbf{x},\mathbf{y})$ (not only those associated with transition and emission probabilities). To compute μ and C in these situations the derivation of appropriate formulas similar to those of μ^e_{pq}, c^e_{pq} and $c^{et}_{pqp'z}$ is straightforward.

3.2.2 ESTIMATING μ AND C BY RANDOM SAMPLING

Still, the computational cost increases with the number of features since for HMMs that are not 'fully connected', it may occur that the number of different values in the matrix C scales quadratically with the observations alphabet size n_o. However we show that in this case accurate and efficient approximation algorithms can be used to obtain close estimates of the mean and the variance values with a significantly reduced computational cost. This can be achieved by considering a finite subsample of all possible values for the output $\mathbf{y}$, rather than using the DP approaches. This comment holds generally for all learning problems considered in this paper, and we come back to this in the theoretical discussion in 6.1 as well as in the experimental results in 5 to support this claim empirically.

3.3 Sequence Alignment Learning

For the sequence alignment learning task we consider separately the three parameter model, the model with affine gap penalties and the model with substitution matrices.

3.3.1 THE SIMPLEST SCORING SCHEME: MATCH, MISMATCH, GAP

In this model the vector $\mu = [\mu_m\ \mu_s\ \mu_g]^T$ contains the average number of matches, mismatches and gaps computed considering all possible alignments. Its elements can be obtained using Algorithm 2. In a nutshell, the algorithm works as follows. First, a matrix π is filled. Every cell $\pi(i,j)$ contains the number of all possible alignments between two prefixes of the strings S_1 and S_2. In fact each alignment corresponds to a path in the alignment graph associated with the DP matrix. At the same time the DP tables for μ_m, μ_s and μ_g are gradually filled according to appropriate recursive relations.

For example each element $\mu_m(i,j)$ is computed dividing the total number of matches by the number of alignments $\pi(i,j)$. If a match occur in position (i,j) ($M=1$) the total number of matches at step (i,j) is obtained adding to the number of matches in the previous steps ($\mu_m(i,j-1)\pi(i,j-1)$, $\mu_m(i-1,j-1)\pi(i-1,j-1)$ and $\mu_m(i-1,j)\pi(i-1,j)$) $\pi(i-1,j-1)$ times a match. Once the algorithm is terminated, the mean values can be read in the cells $\mu_m(n_1,n_2)$, $\mu_s(n_1,n_2)$ and $\mu_g(n_1,n_2)$.

The covariance matrix C is the 3×3 matrix with elements c_{pq}, $p,q\in\{m,s,g\}$ and it is symmetric ($c_{sg}=c_{gs}$, $c_{mg}=c_{gm}$, $c_{sm}=c_{ms}$). Each value c_{pq} can be obtained considering (4) and computing the associated values v_{pq} with appropriate recursive relations (see Algorithm 2).

3.3.2 AFFINE GAP PENALTIES

As before we can define the vector $\mu=[\mu_m\ \mu_s\ \mu_o\ \mu_e]^T$ and the covariance matrix C as the 4×4 symmetric matrix with elements c_{pq} with $p,q\in\{m,s,o,e\}$. The values of μ and C are computed with DP. In particular μ_m, μ_s, v_{mm}, v_{ms} and v_{ss} are calculated as above, while the other values are obtained with the formulas in Algorithm 5 in appendix E. The terms v_{se} and v_{so} are missing since they can be calculated with the same formulas of v_{me} and v_{mo} simply changing M with $1-M$ and μ_m with μ_s. Note that in some situations for low values of (i,j) some terms are not defined (i.e., $\pi(i,j-3)$ when $j=2$). In such situations they must be ignored in the computation.

3.3.3 EXTENSION TO A GENERAL SCORING MATRIX

The formulas illustrated in the previous paragraphs can be extended to the case of a general substitution matrix with minor modifications. Concerning the mean values, μ_o and μ_e are calculated as before. For the others it is:

$$\mu_{z_{pq}}(i,j):=\frac{\mu_{z_{pq}}(i-1,j)\pi(i-1,j)+\mu_{z_{pq}}(i,j-1)\pi(i,j-1)+(\mu_{z_{pq}}(i-1,j-1)+M)\pi(i-1,j-1)}{\pi(i,j)}$$

where $M=1$ when two corresponding symbols in the alignment are equal to p and q or vice versa with $p,q\in\mathcal{S}$. The matrix C is a symmetric matrix 212×212. The values v_{eo}, v_{ee} and v_{oo} are calculated as above. The derivation of formulas for $v_{z_{pq}z_{p'q'}}$ is straightforward from v_{ms} considering the appropriate values for M and the mean values. The formulas for v_{zo} and v_{ze} follow with minor modification from v_{mo} and v_{me}.

3.4 Learning to Parse

For a given input string $\mathbf{x}$, let μ_p and c_{pq} be the mean of occurrences of rule p and the covariance between the numbers of occurrences of rules p and q, respectively, that is, the elements of μ and C. The following relations hold:

$$\begin{aligned}\mu_p &= \frac{1}{N}\sum_{j=1}^{N}\phi_p(\mathbf{x},\mathbf{y}_j)=\frac{1}{N}\psi_p,\\ nc_{pq} &= \frac{1}{N}\sum_{j=1}^{N}\left(\phi_p(\mathbf{x},\mathbf{y}_j)\phi_q(\mathbf{x},\mathbf{y}_j)\right)-\mu_p\mu_q=\frac{1}{N}\gamma_{pq}-\mu_p\mu_q,\end{aligned}$$

where N is the number of all possible parse trees associated to $\mathbf{x}$, ψ_p is the number of occurrences of the rule p in all the parse tree $\mathbf{y}_j$ given $\mathbf{x}$, and γ_{pq} denotes the cooccurrences of p and q.

To compute C and μ an algorithm based on a bottom-up dynamic programming can be developed. Similarly to sequence labeling three types of recurrence equations must be defined: one to

Algorithm 2 Computation of μ and C with matches, mismatches and gaps.

Input: a pair of sequences S_1 and S_2.

$\pi(0,0) := 1$
$\mu_m(0,0) = \mu_s(0,0) = \mu_g(0,0) := 0$
$v_{mm}(0,0) = v_{ms}(0,0) = v_{ss}(0,0) = v_{sg}(0,0) = v_{mg}(0,0) = v_{gg}(0,0) := 0$
for $i = 1 : n_1$
    $\pi(i,0) := 1$
    $\mu_g(i,0) := \mu_g(i-1,0) + 1$
    $v_{gg}(i,0) := v_{gg}(i-1,0) + 2\mu_g(i-1,0) + 1$
end
for $j = 1 : n_2$
    $\pi(0,j) := 1$
    $\mu_g(0,j) := \mu_g(0,j-1) + 1$
    $v_{gg}(0,j) := v_{gg}(0,j-1) + 2\mu_g(0,j-1) + 1$
end
for $i = 1 : n_1$
    for $j = 1 : n_2$
        $\pi(i,j) := \pi(i-1,j-1) + \pi(i,j-1) + \pi(i-1,j)$
        if $s_1(i) = s_2(j)$ **then** $M := 1$ **else** $M := 0$
        $\mu_m(i,j) := \frac{\mu_m(i-1,j)\pi(i-1,j) + \mu_m(i,j-1)\pi(i,j-1) + (\mu_m(i-1,j-1)+M)\pi(i-1,j-1)}{\pi(i,j)}$
        $\mu_s(i,j) := \frac{\mu_s(i-1,j)\pi(i-1,j) + \mu_s(i,j-1)\pi(i,j-1) + (\mu_s(i-1,j-1)+(1-M))\pi(i-1,j-1)}{\pi(i,j)}$
        $\mu_g(i,j) := \frac{\mu_g(i-1,j)+1)\pi(i-1,j) + (\mu_g(i,j-1)+1)\pi(i,j-1) + \mu_g(i-1,j-1)\pi(i-1,j-1)}{\pi(i,j)}$
        $v_{mm}(i,j) := \frac{1}{\pi(i,j)}(v_{mm}(i-1,j)\pi(i-1,j) + v_{mm}(i,j-1)\pi(i,j-1)$
            $+(v_{mm}(i-1,j-1) + 2M\mu_m(i-1,j-1) + M)\pi(i-1,j-1))$
        $v_{ss}(i,j) := \frac{1}{\pi(i,j)}(v_{ss}(i-1,j)\pi(i-1,j) + v_{ss}(i,j-1)\pi(i,j-1)$
            $+(v_{ss}(i-1,j-1) + 2(1-M)\mu_s(i-1,j-1) + (1-M))\pi(i-1,j-1))$
        $v_{gg}(i,j) := \frac{1}{\pi(i,j)}(v_{gg}(i-1,j) + 2\mu_g(i-1,j) + 1)\pi(i-1,j)$
            $+(v_{gg}(i,j-1) + 2\mu_g(i,j-1) + 1)\pi(i,j-1) + v_{gg}(i-1,j-1)\pi(i-1,j-1))$
        $v_{mg}(i,j) := \frac{1}{\pi(i,j)}(v_{mg}(i-1,j) + \mu_m(i-1,j))\pi(i-1,j) + (v_{mg}(i,j-1)$
            $+\mu_m(i,j-1))\pi(i,j-1) + (v_{mg}(i-1,j-1) + M\mu_g(i-1,j-1))\pi(i-1,j-1))$
        $v_{sg}(i,j) := \frac{1}{\pi(i,j)}(v_{sg}(i-1,j) + \mu_s(i-1,j))\pi(i-1,j) + (v_{sg}(i-1,j-1)$
            $+(1-M)\mu_g(i-1,j-1) + (v_{sg}(i,j-1) + \mu_s(i,j-1))\pi(i,j-1))\pi(i-1,j-1))$
        $v_{ms}(i,j) := \frac{1}{\pi(i,j)}(v_{ms}(i-1,j)\pi(i-1,j) + v_{ms}(i,j-1)\pi(i,j-1)$
            $+(v_{ms}(i-1,j-1) + M\mu_s(i-1,j-1) + (1-M)\mu_m(i-1,j-1))\pi(i-1,j-1))$
    end
end

Output: $\mu_m(n_1,n_2)$, $\mu_s(n_1,n_2)$, $\mu_g(n_1,n_2)$,
    $c_{mm}(n_1,n_2) := v_{mm}(n_1,n_2) - \mu_m(n_1,n_2)^2$,
    $c_{ss}(n_1,n_2) := v_{ss}(n_1,n_2) - \mu_m(n_1,n_2)^2$,
    $c_{gg}(n_1,n_2) := v_{gg}(n_1,n_2) - \mu_m(n_1,n_2)^2$,
    $c_{ms}(n_1,n_2) := v_{ms}(n_1,n_2) - \mu_m(n_1,n_2)\mu_s(n_1,n_2)$,
    $c_{mg}(n_1,n_2) := v_{mg}(n_1,n_2) - \mu_m(n_1,n_2)\mu_g(n_1,n_2)$,
    $c_{sg}(n_1,n_2) := v_{sg}(n_1,n_2) - \mu_s(n_1,n_2)\mu_g(n_1,n_2)$

compute the number of parse trees N, another the number of occurrences ψ of each parameter, and the latter the number of cooccurrences γ of each pair of parameters.

For a given input string $\mathbf{x} = (x_1\ x_2 \ldots\ x_m)$, x_s denotes the s-th symbol of $\mathbf{x}$, and $x_{s|t}$ the substring from the s-th symbol to the t-th symbol. We count the number of possible trees N given $\mathbf{x}$ with a DP algorithm such as the CYK algorithm. We use two types of auxiliary variables, $\pi(s,t,\Upsilon_i)$ and $\pi(s,t,\Upsilon_i,\alpha)$ which are the number of possible parse trees whose root is Υ_i for substring $x_{s|t}$, and the number of possible parse trees whose root is applied to rule $\Upsilon_i \to \alpha$ for substring $x_{s|t}$, where $(\Upsilon_i \to \alpha) \in R$.

Then $\pi(s,t,\Upsilon_i)$ is calculated as follows:

$$\pi(s,t,\Upsilon_i) = \sum_{\alpha:(\Upsilon_i\to\alpha)\in\Upsilon} \pi(s,t,\Upsilon_i,\alpha),$$

where:

$$\pi(s,t,\Upsilon_i,\alpha) \quad = \quad \begin{cases} 1 & \alpha = X \in \Sigma_x,\ s=t, \text{ and } X = x_s, \\ \sum_{r=s}^{t-1} \pi(s,r,\Upsilon_{k_1})\pi(r+1,t,\Upsilon_{k_2}) & \alpha = \Upsilon_{k_1}\Upsilon_{k_2} \text{ and } s<t, \\ \pi(s,t,\Upsilon_k) & \alpha = \Upsilon_k, \\ \pi(s+1,t-1,\Upsilon_k) & \alpha = X\Upsilon_k X',\ X = x_s, \text{ and } X' = x_t, \\ 0 & \text{otherwise.} \end{cases}$$

Upon completion of the recursion, $N = \pi(1,m,S)$ is the number of all possible parse trees given $\mathbf{x}$.

We then count the number of occurrences of each rule in all possible parse trees. $\psi_p(s,t,\Upsilon_i)$ denotes the number of occurrences of rule p in all possible parse trees whose root is Υ_i for $x_{s|t}$. We compute $\psi_p(s,t,\Upsilon_i)$ as follows:

$$\psi_p(s,t,\Upsilon_i) = \sum_{\alpha:(\Upsilon_i\to\alpha)\in\Upsilon} \psi_p(s,t,\Upsilon_i,\alpha),$$

where:

$$\begin{aligned} &\psi_p(s,t,\Upsilon_i,\alpha) \\ &= \begin{cases} 1 & \alpha = X = x_s,\ s=t, \\ & \text{and } p = \Upsilon_i \to X, \\ \sum_{r=s}^{t-1} (\psi_p(s,r,\Upsilon_{k_1})\pi(r+1,t,\Upsilon_{k_2}) & \\ \quad +\pi(s,r,\Upsilon_{k_1})\psi_p(r+1,t,\Upsilon_{k_1})) & \\ \quad +I(p,\Upsilon_i\to\alpha)\pi(s,r,\Upsilon_{k_1})\pi(r+1,t,\Upsilon_{k_2}) & \alpha = \Upsilon_{k_1}\Upsilon_{k_2}, \text{ and } s<t, \\ \psi_p(s,t,\Upsilon_k) + I(p,\Upsilon_i\to\alpha)\pi(s,t,\Upsilon_k) & \alpha = \Upsilon_k, \\ \psi_p(s+1,t-1,\Upsilon_k) + I(p,\Upsilon_i\to\alpha)\pi(s+1,t-1,\Upsilon_k) & \alpha = X\Upsilon_k X' \text{ and } s+1<t, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

with $I(p,\Upsilon_i\to\alpha) = 1$ if $p = (\Upsilon_i \to \alpha)$, otherwise it is $I(p,\Upsilon_i\to\alpha) = 0$. Then, $\psi_p(1,m,S)$ is the number of occurrences of p in all parse trees given $\mathbf{x}$.

We count the number of cooccurrences $\gamma_{pq}(s,t,\Upsilon_i)$ in each pair p and q of rules. $\gamma_{pq}(s,t,\Upsilon_i,\alpha)$ denotes the number of cooccurrences in all possible parse trees whose root is Υ_i for $x_{s|t}$. We calculate $\gamma_{pq}(s,t,\Upsilon_i)$ as follows:

$$\gamma_{pq}(s,t,\Upsilon_i) = \sum_{\alpha:(\Upsilon_i\to\alpha)\in\Upsilon} \gamma_{pq}(s,t,\Upsilon_i,\alpha),$$

where:

$$
\gamma_{pq}(s,t,\Upsilon_i,\alpha)
$$

$$
= \begin{cases}
0 & \alpha = X \text{ and } p \neq q, \\
1 & \alpha = X \text{ and } p = q = (\Upsilon_i \rightarrow \alpha), \\
\sum_{r=s}^{t-1} \Big(\gamma_{pq}(s,r,\Upsilon_{k_1})\pi(r+1,t,\Upsilon_{k_2}) & \\
+\pi(s,r,\Upsilon_{k_1})\gamma_{pq}(r+1,t,\Upsilon_{k_2}) & \\
+\psi_p(s,r,\Upsilon_{k_1})\psi_q(r+1,t,\Upsilon_{k_2}) & \\
+\psi_q(s,r,\Upsilon_{k_1})\psi_p(r+1,t,\Upsilon_{k_2}) & \\
+I(p,\Upsilon_i \rightarrow \alpha) f(p,s,r,t,\Upsilon_{k_1},\Upsilon_{k_2}) & \\
+I(q,\Upsilon_i \rightarrow \alpha) f(q,s,r,t,\Upsilon_{k_1},\Upsilon_{k_2}) & \\
+I(p,\Upsilon_i \rightarrow \alpha) I(q,\Upsilon_i \rightarrow \alpha)\pi(s,r,\Upsilon_{k_1})\pi(r+1,t,\Upsilon_{k_2}) \Big) & \alpha = \Upsilon_{k_1}\Upsilon_{k_2} \text{ and } s < t, \\
\gamma_{pq}(s,t,\Upsilon_k) & \\
+I(p,\Upsilon_i \rightarrow \alpha)\psi_q(s,t,\Upsilon_i) + I(q,\Upsilon_i \rightarrow \alpha)\psi_p(s,t,\Upsilon_i) & \\
+I(p,\Upsilon_i \rightarrow \alpha) I(q,\Upsilon_i \rightarrow \alpha)\pi(s,t,\Upsilon_i) & \alpha = \Upsilon_k, \\
\gamma_{pq}(s+1,t-1,\Upsilon_k) & \\
+I(p,\Upsilon_i \rightarrow \alpha)\psi_q(s+1,t-1,\Upsilon_k) & \\
+I(q,\Upsilon_i \rightarrow \alpha)\psi_p(s+1,t-1,\Upsilon_k) & \\
+I(p,\Upsilon_i \rightarrow \alpha) I(q,\Upsilon_i \rightarrow \alpha)\pi(s+1,t-1,\Upsilon_k) & \alpha = X\Upsilon_k X',\ s+1 < t, \\
 & x_s = X, \text{ and } x_t = X', \\
0 & \text{otherwise}
\end{cases}
$$

with $f(p,s,r,t,\Upsilon_{k_1},\Upsilon_{k_2}) = \psi_p(s,r,\Upsilon_{k_1})\pi(r+1,t,\Upsilon_{k_2}) + \pi(s,r,\Upsilon_{k_1})\psi_p(r+1,t,\Upsilon_{k_2})$. Finally, $\gamma_{pq}(1,m,S)$ is the number of cooccurrences of rules p and q in all parse trees given $\mathbf{x}$.

In the following section we discuss how we can use the computed first and second order statistics to define a suitable objective function which can be optimized for structured output learning tasks.

4. Moment-based Approaches to Structured Output Prediction

Suppose we have a training set of input-output pairs $\mathcal{T} = \{(\mathbf{x}^1,\overline{\mathbf{y}}^1),(\mathbf{x}^2,\overline{\mathbf{y}}^2),\ldots,(\mathbf{x}^\ell,\overline{\mathbf{y}}^\ell)\}$. The task we consider is to find the parameter values θ such that the optimal given output variables $\overline{\mathbf{y}}^i$ can be reconstructed from $\mathbf{x}^i$, $\forall\ 1 \leq i \leq \ell$. We want to fulfill this task by defining a suitable objective function which is a convex function of the first and second order statistics we presented before. Based on this idea we introduce two possible approaches.

4.1 Training Sets of Size One

To give an intuition of the main idea behind both methods, we first analyze the situation where the training set in made of only one pair $(\mathbf{x},\overline{\mathbf{y}})$. In this situation, both methods are identical to each other.

The idea is to consider the distribution of the scores for all possible $\mathbf{y}$. We then define a measure of separation between the score of the correct training output, and the entire distribution of all scores for all possible outputs. More specifically, the objective function we propose is the difference between the score of the true output and the mean score of the distribution, divided by the square root of the variance as a normalization. Mathematically:

$$
\max_{\theta} \frac{s_\theta(\mathbf{x},\overline{\mathbf{y}}) - M_{1,\theta}(\mathbf{x})}{\sqrt{M_{2,\theta}(\mathbf{x})}} \quad = \quad \max_{\theta} \frac{\theta^T b}{\sqrt{\theta^T C \theta}} \tag{5}
$$

where $b = \phi(\mathbf{x}, \overline{\mathbf{y}}) - \mu$ is the difference between the feature vector associated to the optimal output and the average feature vector μ. Maximizing this objective over θ means that we search for a parameter vector θ that makes the score of the correct output $\overline{\mathbf{y}}$ as different as possible from the mean score, measured in number of standard deviations. This corresponds to a well known quantity in statistics: the Z-score. Given the distribution of all possible scores (i.e., given its mean and its variance), the Z-score of the correct pair $(\mathbf{x}, \overline{\mathbf{y}})$ is defined as the number of standard deviations its score $s(\mathbf{x}, \overline{\mathbf{y}})$ is away from the mean of the distribution.

The Z-score is an interesting measure of separation between the correct output and the bulk of all possible outputs corresponding to a given input. Under normality assumptions, it is directly equivalent to a *p*-value. Hence, maximizing the Z-score can be interpreted as maximizing the significance of the score of the correct pair: the larger the Z-score, the more significant it is, and the fewer other outputs would achieve a larger score. If the normality assumption is too unrealistic, one could still apply a (looser) Chebyshev tail bound to show that the number of scores that exceed the score of a large training output score $s_\theta(\mathbf{x}, \overline{\mathbf{y}})$ is small.

To quantify this connection between the Z-score of a training pair and the rank of its score among all other scores, we would like to introduce an alternative formulation for optimization problem (5).

Proposition 4 *Optimization problem (5) is equivalent to:*

$$\begin{array}{rl} \min_\theta & \frac{1}{N}\sum_{j=1}^N \xi_j^2 \\ s.t. & \theta^T\left(\phi(\boldsymbol{x},\overline{\boldsymbol{y}}) - \phi(\boldsymbol{x},\boldsymbol{y}_j)\right) = 1 + \xi_j \quad \forall j \end{array} \tag{6}$$

in the sense that it is optimized by the same value of θ *or a scalar multiple of it.*

Proof Substituting ξ_j from the constraint in the objective, the objective of optimization problem (6) is equivalent to:

$$\begin{aligned} & \frac{1}{N}\theta^T \sum_{j=1}^N (\phi(\mathbf{x},\overline{\mathbf{y}}) - \phi(\mathbf{x},\mathbf{y}_j))(\phi(\mathbf{x},\overline{\mathbf{y}}) - \phi(\mathbf{x},\mathbf{y}_j))^T\theta - \frac{2}{N}\theta^T \sum_{j=1}^N (\phi(\mathbf{x},\overline{\mathbf{y}}) - \phi(\mathbf{x},\mathbf{y}_j)) + 1 \\ = \;& \frac{1}{N}\theta^T \sum_{j=1}^N (\mu - \phi(\mathbf{x},\mathbf{y}_j))(\mu - \phi(\mathbf{x},\mathbf{y}_j))^T\theta + \theta^T(\phi(\mathbf{x},\overline{\mathbf{y}}) - \mu)(\phi(\mathbf{x},\overline{\mathbf{y}}) - \mu)^T\theta \\ & -2(\phi(\mathbf{x},\overline{\mathbf{y}}) - \mu) + 1 \\ = \;& \theta^T C\theta + (\theta^T b - 1)^2. \end{aligned}$$

Hence, the optimization problem (6) is equivalent to:

$$\min_\theta \quad \theta^T C\theta + (\theta^T b - 1)^2.$$

Now, note that the objective in optimization problem (5) is invariant with respect to scaling of θ. Hence, we can fix the scale arbitrarily, and require $\theta^T b = 1$. The optimization problem then reduces to (using the monotonicity of the square root):

$$\begin{array}{rl} \min_\theta & \theta^T C\theta \\ \text{s.t.} & \theta^T b = 1. \end{array}$$

The optimality conditions of the former are $C\theta + bb^T\theta = b \Leftrightarrow C\theta = (1 - b^T\theta)b$, and the Lagrange optimality conditions of the latter are $C\theta = \lambda b$ with λ a Lagrange multiplier. Hence, both optimality conditions and optimization problems are equivalent in the sense that they are optimized by the

same θ up to a scaling factor. ■

The following interesting theorem now establishes the link between the relative ranking loss $\mathcal{L}_\theta^{RR}$ as defined in Section 2.2 and the above optimization problem.

Theorem 5 (Relative ranking loss upper bound) *Let us denote by $\mathcal{L}_\theta^{RRU}(\boldsymbol{x},\bar{\boldsymbol{y}})$ the value of the objective of optimization problem (6) evaluated on training pair $(\boldsymbol{x},\bar{\boldsymbol{y}})$:*

$$\mathcal{L}_\theta^{RRU}(\boldsymbol{x},\bar{\boldsymbol{y}}) \quad = \quad \frac{1}{N}\sum_{j=1}^{N}\xi_i^2 = \frac{1}{N}\sum_{j=1}^{N}\left(\theta^T\left(\phi(\boldsymbol{x},\bar{\boldsymbol{y}})-\phi(\boldsymbol{x},\boldsymbol{y}_j)\right)-1\right)^2.$$

Then,

$$\mathcal{L}_\theta^{RRU}(\boldsymbol{x},\bar{\boldsymbol{y}}) \geq \mathcal{L}_\theta^{RR}(\boldsymbol{x},\bar{\boldsymbol{y}}).$$

(The RRU in the superscript stands for Relative Ranking Upper bound.)

Proof The rank of $s_\theta(\mathbf{x},\bar{\mathbf{y}})$ among all $s_\theta(\mathbf{x},\mathbf{y}_j)$ for all possible $\mathbf{y}_j$ is given by the number of $\mathbf{y}_j$ for which $\theta^T\left(\phi(\mathbf{x},\bar{\mathbf{y}})-\phi(\mathbf{x},\mathbf{y}_j)\right)\leq 0$. Hence, this is the number of times that $\xi_i \leq -1$ in optimization problem (6), such that the objective is at least as large as the rank divided by N, that is, the relative rank. ■

Additionally, we would like to point out that optimization problem (5) and equivalently (6) is also strongly connected to Fisher's discriminant analysis (FDA). Intuitively, maximizing our objective function corresponds to maximizing the distance between the mean of the distribution of the scores for all possible incorrect pairs and the 'mean' of the 'distribution' of the score for the single correct output, normalized by the sum of the standard deviations (note that one class reduces to one data point so the associated standard deviation is zero). Then (5) is equivalent to performing FDA when one class reduces to a single data point as defined by the correct training label.

4.2 Training Sets of General Sizes

Having introduced the main idea on the special case of a training set of size 1, we now turn back to the general situation where we are interested in computing the optimal parameter vector given a training set $\mathcal{T}$ of ℓ pairs of sequences. We will consider two different generalizations to which we refer as the Z-score based approach, and as structured output discriminant analysis (SODA).

4.3 Z-score Based Algorithm

In the first generalization, we will emphasize the Z-score interpretation. For training sets containing more than one input-output pair, we need to redefine the Z-score for a set of ℓ pairs of sequences $\mathcal{T}=\{(\mathbf{x}^1,\bar{\mathbf{y}}^1),(\mathbf{x}^2,\bar{\mathbf{y}}^2),\ldots,(\mathbf{x}^\ell,\bar{\mathbf{y}}^\ell)\}$. A natural way is to do this based on the global score: the sum of the scores for all sequence pairs in the set. Its mean is the sum of the means for all sequence pairs $(\mathbf{x}^i,\bar{\mathbf{y}}^i)$ separately, and can be summarized by $\bar{b}=\sum_i b_i$. Similarly, for the covariance matrix: $\bar{C}=\sum_i C_i$. Hence, the Z-score definition can naturally be extended to more than one input-output pair by using $\bar{b}$ and $\bar{C}$ instead of b and C in (5). In summary, extending the optimization problem (5) to the general situation of a given training set $\mathcal{T}$, the optimization problem we are interested in is:

$$\max_\theta \quad \frac{\theta^T\bar{b}}{\sqrt{\theta^T\bar{C}\theta}}. \tag{7}$$

The solution of (7) can be computed by simply solving the linear system $\bar{C}\theta = \bar{b}$, where $\bar{C}$ is a symmetric positive definite matrix. If $\bar{C}$ is not symmetric positive definite, regularization can be introduced in a straightforward way (similar as in FDA) by solving $(\bar{C} + \lambda I)\theta = \bar{b}$ instead. This effectively amounts to restricting the norm of θ to small values. Then the optimal parameter vector can be obtained extremely efficiently by using iterative methods such as the conjugate gradient method.

4.3.1 INCORPORATING THE HAMMING DISTANCE

A nice property of this approach is that it can be extended to take into account the Hamming distance between the output vectors. For each pair $(\mathbf{x}, \mathbf{y})$ we consider the score:

$$s(\mathbf{x}, \mathbf{y}) = \theta^T \phi(\mathbf{x}, \mathbf{y}) + \delta_H(\mathbf{y}, \bar{\mathbf{y}}) = \theta'^T \phi'(\mathbf{x}, \mathbf{y})$$

where we have defined the vectors $\theta'^T = \begin{bmatrix} \theta^T & 1 \end{bmatrix}$ and $\phi'(\mathbf{x}, \mathbf{y})^T = \begin{bmatrix} \phi(\mathbf{x}, \mathbf{y})^T & \delta_H(\mathbf{y}, \bar{\mathbf{y}}) \end{bmatrix}$. It is easy to verify that the associated optimization problem has the same form of (7) when the vectors θ' and ϕ' are considered. In practice the covariance matrix C is augmented with one column (and one row, since it is symmetric) containing the covariance values between the loss term and all the other parameters. We refer to this column as c_δ. Analogously the mean vector is augmented by one value (μ_δ) that represents the mean value of the terms $\delta_H(\mathbf{y}, \bar{\mathbf{y}})$ computed along all negative pseudoexamples. When the Hamming distance is adopted the computation of μ_δ and c_δ can be realized with DP algorithms. For example for sequence labeling learning Algorithm 1 is used with recursive relations similar to those in Algorithm 4.

4.3.2 Z-SCORE APPROACH WITH CONSTRAINTS

As a side remark, let us draw a connection with existing MM approaches such as described in Taskar et al. (2003) and in Tsochantaridis et al. (2005).

Their approach to structured output learning is to explicitly search for the parameter values θ such that the optimal hidden variables $\bar{\mathbf{y}}^i$ can be reconstructed from $\mathbf{x}^i$, $\forall 1 \leq i \leq \ell$. In formulas these conditions can be expressed as:

$$\theta^T \phi(\mathbf{x}^i, \bar{\mathbf{y}}^i) \geq \theta^T \phi(\mathbf{x}^i, \mathbf{y}^i_j) \quad \forall 1 \leq i \leq \ell \quad \forall 1 \leq j \leq N_i. \tag{8}$$

This set of constraints defines a convex set in the parameter space and its number is massive, due to the huge size of the output space. To obtain an optimal set of parameters θ that successfully fulfill (8) usually an optimization problem is formulated with these constraints, together with a suitable objective function. In MM approaches for example this objective function is typically chosen to be the squared norm of the parameter vector.

Interestingly, using the Z-score as objective function, we observe that most (and often all) of the constraints (8) are satisfied automatically, which often leads to a satisfactory result with a good generalization performance without considering the constraints explicitly.

However, in the cases where the result of (7) still violates some of the constraints and one wishes to avoid this, one can choose to impose these explicitly. The resulting optimization problem is still convex and it reduces to:

$$\begin{aligned} \min_\theta \quad & \theta^T \bar{C} \theta \\ \text{s.t.,} \quad & \theta^T \bar{b} \geq 1 \\ & \theta^T \phi(\mathbf{x}^i, \bar{\mathbf{y}}^i) \geq \theta^T \phi(\mathbf{x}^i, \mathbf{y}^i_j) \quad \forall 1 \leq i \leq \ell \quad \forall 1 \leq j \leq N_i. \end{aligned} \tag{9}$$

We have developed an incremental algorithm that implements problem (9), shown in Algorithm 3 (see Tsochantaridis et al., 2005, for a similar approach and a more detailed study). First a feasible solution is determined without adding any constraints. Then the following steps are repeated until convergence. For each training example, the most likely hidden variables are determined by a Viterbi-like algorithm. If its score is higher than the given one, the associated constraint is added to the set of constraints of the problem (9) and (9) is solved. The convergence is guaranteed from the convexity of the problem. Each added constraint provides the effect of restricting the feasible region.

Algorithm 3 Iterative algorithm to incorporate the active constraints.

Input: The training set $\mathcal{T} = \{(\mathbf{x}^1, \overline{\mathbf{y}}^1)(\mathbf{x}^2, \overline{\mathbf{y}}^2) \dots (\mathbf{x}^\ell, \overline{\mathbf{y}}^\ell)\}$

$\mathcal{C} := \oslash$
for $i = 1, \dots, \ell$ compute $\underline{b}_i$ and C_i
Compute $\underline{b} := \sum_i b_i$ and $\overline{C} := \sum_i C_i$
Find θ_{opt} solving (7)
Repeat
  $exit := 0$
  for $i = 1, \dots, \ell$
    Compute $\tilde{\mathbf{y}}^i := \arg\max_{\mathbf{y}} \ \theta_{opt}^T \phi(\mathbf{x}^i, \mathbf{y})$
    If $\theta_{opt}^T(\phi(\mathbf{x}^i, \overline{\mathbf{y}}^i) - \phi(\mathbf{x}^i, \tilde{\mathbf{y}}^i)) \leq 0$
      $exit := 1$
      $\mathcal{C} := \mathcal{C} \cup \{\theta^T(\phi(\mathbf{x}^i, \overline{\mathbf{y}}^i) - \phi(\mathbf{x}^i, \tilde{\mathbf{y}}^i)) \geq 0\}$
      Find θ_{opt} solving (7) s.t. $\mathcal{C}$
    end
  end
until $exit = 1$

Output: θ_{opt}

Often, real data sets do not allow a feasible solution θ. A possible way to deal with this problem is by the introduction of slack variables or relaxing the constraints by requiring the inequalities to hold subject to the small possible used-defined tolerance ε (Ricci et al., 2007). However, we argue that in such cases simply optimizing the Z-score as described earlier without adding any constraints may offer a natural and computationally attractive alternative to using soft-margin constraints.

4.3.3 RELATED WORK

It is worth noting that the Z-score has previously been used in the context of sequence alignment, although in previous work it was computed with respect to different distributions. In Doolittle (1981) Z-scores are used to assess the significance of a pairwise alignment between two aminoacid sequences and are computed calculating the mean and the standard deviation values over a random sample taken from a standard database or obtained permuting the given sequence. A high Z-score corresponds to an alignment that is less likely to occur by chance and therefore biologically significant.

To our knowledge, there are no methods to calculate the Z-scores on a set of random sequences in exact way. The only attempt to this aim is due to Booth et al. (2004). They proposed an efficient algorithm that finds the standardized score in the case of permutations of the original sequences but this approach is limited to the ungapped sequences. We have to stress that we consider a much wider range of applications (not only sequence alignment) and a slightly different definition of the Z-score: for example, for sequence alignment for each pair of given sequences the mean and standard deviation are computed over the set of all possible alignments (also with gaps and not only the optimal ones) without any permutations.

4.4 SODA: Structured Output Discriminant Analysis

Another way to extend problem (5) to the general situation of a training set $\mathcal{T}$ is to minimize the empirical risk associated to the upper bound on the relative ranking loss $\mathcal{R}^{RRU}$, defined in the usual way as:

$$\mathcal{R}_\theta^{RRU}(\mathcal{T}) \quad = \quad \sum_{i=1}^{\ell} \mathcal{L}_\theta^{RRU}(\mathbf{x}^i, \overline{\mathbf{y}}^i).$$

This simple summing of the loss for individual data points leaves the connection with FDA more intact, hence the name SODA for structured output discriminant analysis. As usual in empirical risk minimization, the hope is that minimizing the empirical risk will ensure that the expected loss $E_{(\mathbf{x},\bar{\mathbf{y}})}\left\{\mathcal{L}_\theta^{RRU}(\mathbf{x},\overline{\mathbf{y}})\right\}$ (here the relative ranking loss) is small as well, and we will shortly prove that this is the case in 6.1. Filling everything in, the resulting empirical risk minimization problem becomes:

$$\begin{aligned} \min_\theta \quad & \sum_{i=1}^{\ell} \frac{1}{N_i} \sum_{j=1}^{N_i} \xi_{ij}^2 \\ \text{s.t.} \quad & \theta^T(\phi(\mathbf{x}^i, \overline{\mathbf{y}}^i) - \phi(\mathbf{x}^i, \mathbf{y}_j^i)) = 1 + \xi_{ij} \quad \forall i. \end{aligned} \tag{10}$$

This is the optimization problem we solve in SODA. To solve it easily, and to elucidate more clearly the analogy with the Z-score approach, we rewrite it one more time as follows.

Proposition 6 *Optimization problem (10) is equivalent to:*

$$\max_\theta \frac{\theta^T b^*}{\sqrt{\theta^T C^* \theta}} \tag{11}$$

where we have defined $b^* = \sum_i b_i$ and $C^* = \sum_i (C_i + b_i {b_i}^T)$. Here, by equivalent we mean that the optimal values for θ differ by a constant scaling factor only. It can be solved efficiently by solving the linear system of equations $C^*\theta = b^*$.

Note that this optimization problem has the same shape as (7) and can be solved again with conjugate gradient algorithms.

Proof We can follow exactly the same procedure as in the proof of Theorem 5 to show that optimization problem (11) is equivalent with:

$$\begin{aligned} \min_\theta \sum_{i=1}^{\ell} \theta^T C_i \theta + (\theta^T b_i - 1)^2 \quad & \Leftrightarrow \quad \min_\theta \theta^T \sum_{i=1}^{\ell} (C_i + b_i b_i^T)\theta - 2\theta^T \sum_{i=1}^{\ell} b_i + 1 \\ & \Leftrightarrow \quad \min_\theta \theta^T C^* \theta - 2\theta b^* + 1. \end{aligned}$$

	MM	HMP	CRFs
Z-score	$5.67e^{-11}$	$1.97e^{-7}$	0.016
SODA	$5.12e^{-10}$	$3.13e^{-6}$	0.04

Table 1: p-values for level of noise $p = 0.4$ and for an HMM with $n_h = 2$ and $n_o = 4$.

The optimality conditions is $C^*\theta = b^*$. In a similar way as in the proof of Theorem 5, we can show that the optimality conditions of optimization problem (10) are given by $C^*\theta = \lambda b^*$, leading to the same value for θ after appropriate scaling. ■

5. Experimental Results

In this subsection we provide some experimental results for the three illustrative examples proposed: sequence labeling, sequence alignment and sequence parse learning.

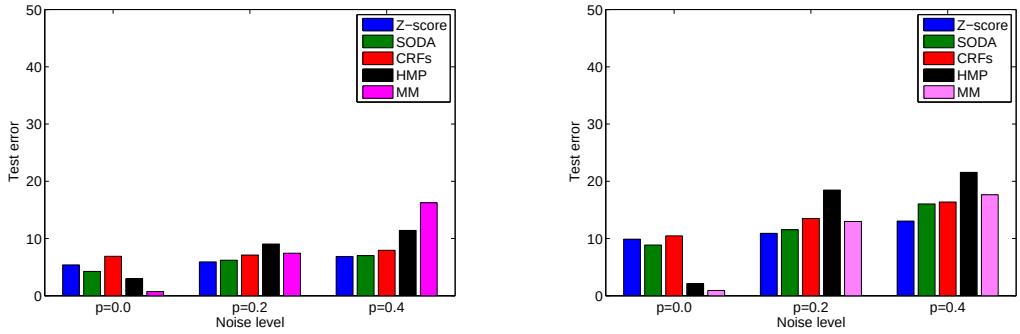
5.1 Sequence Labeling Learning

The first series of experiments, developed in the context of sequence labeling learning, analyzes the behavior of the Z-score based algorithm and of the SODA using both artificial data and sequences of text for named entity recognition. The main aim of this section is to compare our approaches with other existing DLAs on small and medium size data sets.

5.2 Simulation Results

We first present experiments that demonstrate the robustness of our approaches in problems with an increasing degree of noise. We consider two different HMMs, one with $n_h = 2$, $n_o = 4$ and one with $n_h = 3$, $n_o = 5$, with assigned transition and emission probabilities. For these models, we generate hidden and observed sequences of length 100. The training set size is fixed to 20 pairs, while the test set is made up of 100 pairs. Then we add some noise with probabilities $p \in [0,1]$ flipping labels in hidden sequences. More specifically we consider three different scenarios: absence of noise ($p = 0$), moderate level of noise ($p = 0.2$) and noisy data ($p = 0.4$). After learning the parameter, the labeling error (average number of incorrect labels) is measured. We observe the performance of the proposed approaches in comparison with other DLAs such as CRFs, hidden Markov perceptron (HMP) and a MM method with Hamming loss (SVM-struct implementation Tsochantaridis et al., 2005) and linear kernel. The regularization parameters associated to each method are determined based on the performance on a separate validation set of 100 sequences generated together with the training and the test sets. Results are averaged over 1000 training/test samples. In both cases our algorithms outperform other methods for high level of noise, as can be expected (Fig. 4). We also observe slightly better performance of the SODA with respect to the Z-score based algorithm for low p values while with the Z-score a smaller test error is achieved with very noisy data.

To assess the significance of the results obtained comparing our methods with the other DLAs we also run some paired t-tests and compute the associated p-values for both the HMM models and all the levels of noise. Here we only show the p-values obtained by the experiments with high level



(a) (b)

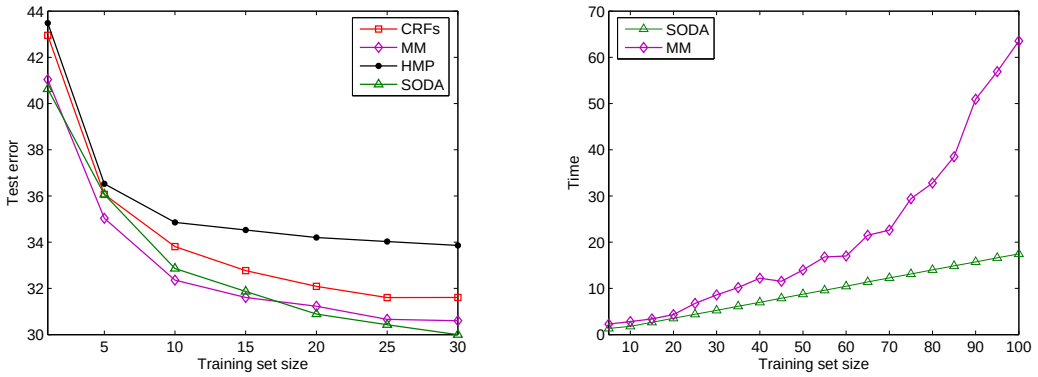
Figure 4: Average number of incorrect labels at varying level of noise for an HMM with (a) $n_h = 2$ and $n_o = 4$ and (b) $n_h = 3$ and $n_o = 5$.

	MM	**HMP**	**CRFs**
Z-score	$11.11e^{-6}$	$3.01e^{-12}$	0.0076
SODA	$8.8e^{-4}$	$5.45e^{-10}$	0.16

Table 2: p-values for level of noise $p = 0.4$ and for an HMM with $n_h = 3$ and $n_o = 5$.

of noise (Tab. 1 and Tab. 2) in order to demonstrate that our approaches significantly outperforms HMP and the MM algorithm in situations where data are noisy. In this scenario SODA and Z-score achieve better performance than CRFs even if in this case the difference of the test error is less evident. On the other hand we also observe that for separable data (absence of noise) the MM algorithm does significantly better than our algorithms (e.g., for the HMM model with $n_h = 2$ and $n_o = 4$ the p-value is $5.04e^{-9}$ for SODA and $1.06e^{-11}$ for the Z-score algorithm). A similar situation occurs also for the HMP (e.g., for the HMM model with $n_h = 2$ and $n_o = 4$ the p-value is $8.65e^{-5}$ for SODA and $4.86e^{-6}$ for the Z-score algorithm). However SODA and Z-score approach still outperform CRFs (e.g., for the HMM model with $n_h = 2$ and $n_o = 4$ the p-value is $3.51e^{-4}$ for SODA and $2.53e^{-3}$ for the Z-score algorithm).

For this series of experiments we also depict some typical learning curves computed for all DLAs considered. We show the curves associated to a HMM model with $n_h = 3$, $n_o = 5$, sequences of length equal to 50 and noise level $p = 0.2$. In this case for the MM algorithm the soft margin parameter C is set equal to 1 and a constant $\varepsilon = 10^{-12}$ specifies the accuracy for constraints to be satisfied. The maximum number of iterations of the averaged perceptron is $T = 100$. CRFs are optimized using a conjugate gradient method. Concerning our approaches we plot results just for the SODA since in this situation (moderate quantity of noise) the learning curves for the Z-score algorithm is almost superimposed to the SODA's one. For the SODA the regularization parameter is $\lambda = 10^{-8}$. The SODA performs better than other methods and among the competing DLAs, the MM approach provides the best performance (Fig. 5.a). Moreover if for the same experiment we also examine the training time we observe (Fig. 5.b) that SODA is definitely faster than the MM algorithm especially for larger data sets.



(a) (b)

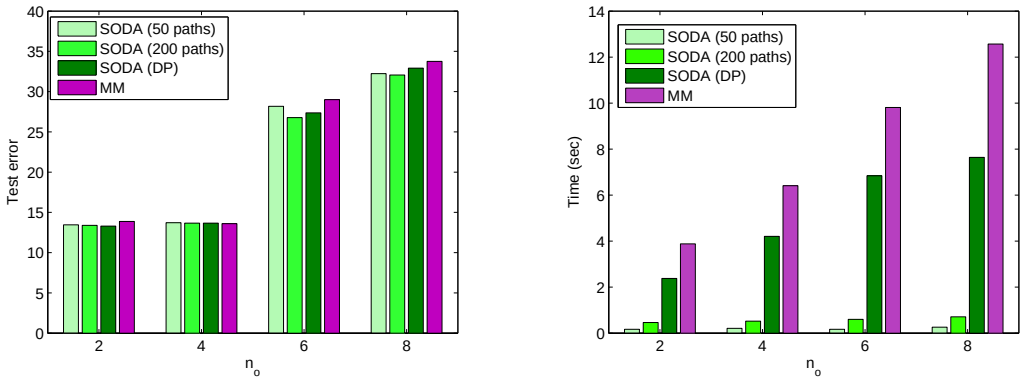
Figure 5: (a) Average number of incorrect labels and (b) computational time as function of the training set size for an HMM with $n_h = 3$ and $n_o = 5$.

A further series of experiments have been conducted to confirm the theoretical results presented in the previous subsection, that is, we want to show that learning with SODA is effectively achieved when mean and covariance matrices are estimated considering just a small subset of incorrect outputs (i.e., incorrect hidden label sequences), taken by random sampling. In fact in the situations where the size of the hidden and the observed space is large and long sequences must be considered, the computation of b^* and C^* with DP can be quite time consuming. Then using random sampling, the computational burden of DP is avoided and the labeling accuracy is still reasonably high, if a sufficient number of possible outputs is sampled. To support this claim we conduct the following experiment. Sequences of length 10 are considered. The training set is fixed to 50 pairs, the test set contains 100 pairs. Sequence pairs are generated with a level of noise $p = 0.2$ obtained by flipping labels. We pick various HMMs: the hidden alphabet size is fixed, $n_h = 3$, while n_o varies. The average labeling error on test set and the time required for computation are reported for SODA with exact matrices, when matrices are computed on a set of 50 and 200 random paths and for the MM method. Results are shown in Fig. 6. While the performance in terms of labeling error are essentially the same for all the algorithms (Fig. 6.a), the computational advantage considering the training time for the sampling approaches is considerable (Fig. 6.b).

5.3 Named Entity Recognition

The second series of experiments have been performed in the context of named entity recognition (NER). In NER phrases in text must be classified as belonging to predefined categories such as persons, organizations, locations, temporal and numerical expressions.

We consider 300 sentences extracted from the Spanish news wire article corpus used for the Special Session of CoNLL2002 on NER. Our subset contains more than 7000 tokens (about 2000 unique) and each sentence has an average length of 30 words. The hidden alphabet is limited to $n_h = 9$ different labels, since the expression types are only persons, organizations, locations and miscellaneous names. Our aim here is not to compete with large scale NER systems but to perform comparison with previous methods so we deliberately choose a small subset and an experimental



(a) (b)

Figure 6: (a) Labeling error on test set and (b) average training time as function of the observation alphabet size n_o.

setup similar to that in Altun et al. (2003b). We perform experiments into different settings: HMM features (the parameters to be determined are the transition and emission probabilities) ($\mathcal{S}_1$) and HMM features of the previous and the next words ($\mathcal{S}_2$). Experiments have been made with a 5-fold cross validation. We compare the performances of our approaches with CRFs, HMP and the MM algorithm with Hamming loss in (Altun et al., 2003b). For the SODA and the Z-score algorithm the regularization parameter is $\lambda = 10^{-8}$. For CRFs we used a public available software (Kudo, 2005) where a quasi-newton optimization technique method is used for optimization. For the MM algorithm a linear kernel is considered, $C = 1$ and $\varepsilon = 0.01$. The number of iterations of the HMP is $T = 200$.

The test errors, reported in Tab. 3, demonstrate the competitiveness of the proposed methods. SODA outperforms all the other approaches for $\mathcal{S}_1$, while it performs slightly worse than the MM algorithm for features $\mathcal{S}_2$. On the other hand for $\mathcal{S}_2$ the best performance is obtained by the Z-score algorithm.

Since the length of feature vectors is large, our approaches are generally slower than MM methods. For very large numbers of parameters, in fact, the time required to compute b^* and C^* may exceed the computation time of competing MM approaches. However, in this case, the sampling strategy can be used to approximate the matrices C^* and b^*. For example in the $\mathcal{S}_1$ setting, the average running time for the SODA is about 9967.47 sec while with SVM-struct the same task is performed in 1043.16 sec. However with the use of approximate matrices computed sampling on 150 random paths and solving the linear system by a conjugate gradient method the computational time is only 656.46 sec.

	Z-score	**SODA**	**MM**	**HMP**	**CRFs**
$\mathcal{S}_1$	11.07	*10.13*	10.97	20.99	11.96
$\mathcal{S}_2$	*7.89*	8.27	8.11	13.78	8.25

Table 3: Classification error on test set on NER (300 sentences).

	Z-score	**SODA**	**MM**	**HMP**	**CRFs**
$\mathcal{S}_1$	9.43	*8.80*	9.35	11.01	9.07
$\mathcal{S}_2$	8.57	8.01	*7.33*	7.83	8.40

Table 4: Classification error on test set on NER (1500 sentences).

To address this problem of scalability of our approach when the number of features is large we also developed a method for solving the linear systems of SODA and of the Z-score approach which is *ad hoc* for problems such as NER where the size of the observation alphabet n_o (i.e., the size of dictionary) tends to be huge while the size of the hidden alphabet n_h (i.e., the number of different labels) is moderate.

The main problem of using our algorithms for tasks with a large number of features is represented by the fact that the matrices C^* and $\bar{C}$ needs to be stored into memory. Moreover solving the corresponding linear systems with conjugate gradient techniques has computational cost $O(d^2)$ which is problematic when d is large. To overcome these difficulties we propose an approach which exploits the sparsity and the redundancy of the covariance matrices to limit the storage requirements and to solve the corresponding linear system with reduced computational cost. This approach is briefly presented in appendix B in the case of sequence labeling and HMM features but an extension of it for other possible configurations of features is possible and quite easy to derive.

Using this method we are able to perform experiments for the NER task on a large subset of the Spanish news wire article corpus of CoNLL2002. We used 1500 sentences which correspond to a dictionary of about 10000 different words. The hidden alphabet is again represented by $n_h = 9$ different labels. The experimental setting is the same of the small data set: we consider the same configuration for features ($\mathcal{S}_1$ and $\mathcal{S}_2$) and we compare the performances of our approaches with CRFs (using CRFs toolkit Kudo, 2005), HMP and the MM algorithm with Hamming loss (SVM-struct implementation Tsochantaridis et al., 2005). Experiments have been made with a 5-fold cross validation procedure and the regularization parameters which provide better performances have been set for all the methods.

From the results, shown in Tab. 4, we can draw similar conclusions that for the small data set: SODA outperforms all the other approaches for $\mathcal{S}_1$, while the MM algorithm provides the smallest test error for $\mathcal{S}_2$. It is somehow surprising that the HMP provides the second best performance for $\mathcal{S}_2$ despite its simplicity. We explain this result considering that with an increased set of features the data tend to be more separable and the HMP tend to outperform our approaches and CRFs.

Note that without having developed an *ad hoc* method such as that described in appendix B we would not have been able to run this second series of experiments on a normal machine since our $d \times d$ covariance matrices are too large to fit into memory (d is about 90000 only for set of features $\mathcal{S}_1$!). For sake of clarity we should also say that sometimes using this method we experienced numerical problems (especially for small regularization parameters) that are probably due to the fact that the approach is based on formulas for matrix inversions. Therefore in the future we plan to develop a better method (e.g., based on updating matrices decompositions such as Cholesky) in order to overcome numerical difficulties.

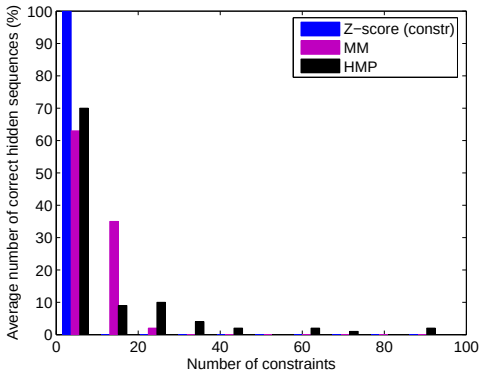


Figure 7: Average number of correctly reconstructed hidden sequences for an HMM with $n_h = 2$ and $n_o = 4$.

5.4 Z-score with Constraints

The last series of experiments shows some results associated with the Z-score approach with constraints (9). We observe experimentally that this approach improves the performance of the unconstrained problem (7) if the noise in the data is limited (i.e., in the feasible or nearly feasible case). For example for the experiments in Fig. 4 when the noise level is $p = 0$ with the constrained Z-score the labeling error is 3.96 and 5.76 respectively for the HMM with $n_h = 2$, $n_o = 4$ and for the HMM with $n_h = 3$, $n_o = 5$ while for the unconstrained problem the error is 5.39 and 9.87 respectively.

Moreover, comparing Algorithm 3 with other iterative approaches (HMP Collins 2002b and MM algorithm Tsochantaridis et al. 2005), the use of the Z-score as objective function ensures that the number of iterations is generally much smaller. Then the computational cost is greatly reduced since adding one inequality means running the Viterbi algorithm. To demonstrate this, we perform the following experiment. A pair of observed and hidden sequences of length $m = 100$ is considered. The task is to estimate the values of transition and emission probabilities such that the observed sequences are generated by the hidden one. The number of constraints needed in the training phase to reconstruct the matrices is averaged on 100 experiments. In Fig. 7 the histograms obtained binning the number of constraints needed to reconstruct the original transition and emission probabilities is shown for an HMM with $n_h = 2$ and $n_o = 4$. For sake of comparison the number of constraints needed when learning is performed with the perceptron (Collins, 2002b) and a MM approach (Tsochantaridis et al., 2005) is also provided. As expected, optimizing the Z-score, much less constraints are needed.

5.5 Sequence Alignment Learning

The second series of experiments has been performed in the context of sequence alignment learning. The aim of this section is to compare the performance of our algorithms with a traditional generative approach. Among the proposed methods we present the results associated to SODA since the performance obtained with the Z-score algorithm are nearly identical.

We construct substitution matrices with elements generated randomly but such that the values on the main diagonal are larger than the other. In particular we consider two types of matrices associated respectively with a 3 parameter model (i.e., matches, mismatches and gaps) and a 211

n	**SODA (3)**	**SODA (211)**	**Generative**	**HMP**
1	5.1 ± 1.2	96.12 ± 13.3	98.14 ± 14.5	93.8 ± 12.1
2	2.9 ± 0.8	84.7 ± 7.5	98.01 ± 12.2	83.98 ± 8.6
5	2.32 ± 1.0	74.81 ± 6.2	97.4 ± 7.4	76.13 ± 5.2
10	2.11 ± 0.7	60.08 ± 3.2	92.93 ± 5.2	57.93 ± 2.9
20	2.1 ± 0.5	43.18 ± 2.2	79.13 ± 4.2	42.68 ± 2.1
50	1.87 ± 0.3	35.56 ± 1.4	48.31 ± 2.9	31.92 ± 1.2
100	1.53 ± 0.4	30.84 ± 1.0	32.05 ± 1.5	28.4 ± 0.9
500	0.98 ± 0.3	23.47 ± 0.2	26.11 ± 0.6	21.7 ± 0.4

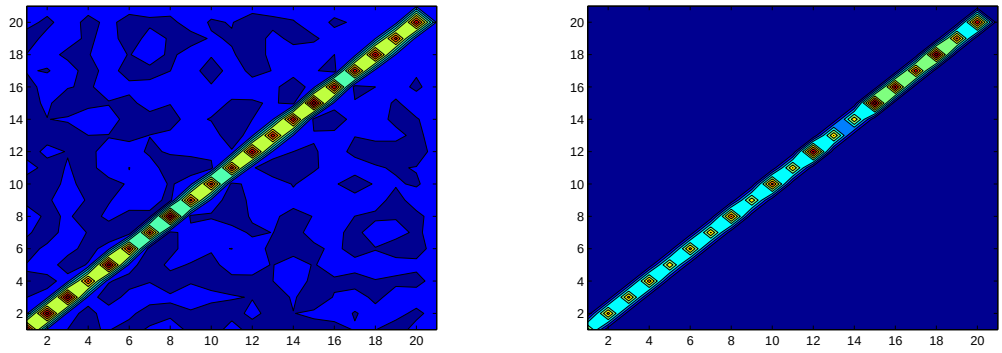
Table 5: Classification error (mean and standard deviation) on test set as function of the training set size n.

parameter models (substitution matrix plus gap penalty). Starting from these matrices we then generate random pairs of sequences of length 10 from a 20 letter alphabet. Pairs are constructed in a way that 50% of symbols between the two sequences are equal. The task we consider is to reconstruct the given matrices starting from training sets of varying size n.

Table 5 shows the results in terms of the test error (number of incorrectly aligned sequences), averaged on 100 runs. A small regularization value $\lambda = 10^{-12}$ is used for SODA. The first two columns of Tab. 5 present the test error for SODA respectively for the 3 and the 211 parameter model. As expected from theory, the convergence to zero error is faster for the 3 parameter model. For the 211 parameter model we also compare SODA with a generative sequence alignment model, where substitution rates between amino acids are computed using Laplace estimates. The gap penalty must be set manually and we choose the value $\theta_g = -0.1$ which guarantees the best performance on the test set. The third column of Tab. 5 shows the associated results: SODA performs better than the generative approach, especially for training set of small size. We also compare the performance of our method with another discriminative approach: the hidden Markov perceptron. In this situation the test error of SODA is slightly larger than that of the HMP. This is in accordance to what we observe in the sequence labeling learning task: when data are linearly separable other discriminative approaches appear more suitable than SODA.

For the SODA algorithm with the 211 parameter model and for a training set with $n = 100$ aligned pairs we also depict the substitution matrix computed by SODA and we compare it with the given one. As one can easily observe, the computed matrix has a similar structure of the correct matrix, having the elements with higher values on the diagonal (Fig. 8).

Note that, in the context of sequence alignment, being the number of parameters limited at most to 211 the training phase is not time consuming even for large training set. In fact the computational cost is dominated by the calculation of mean and covariance matrices which can be greatly sped up by sampling while solving the linear system $C^*\theta = b^*$ is indeed very fast. Here we only consider training sets of size up to 500 pairs of sequences since the advantage in terms of test error for SODA (and in general for all discriminative approaches) with respect to generative approaches is more evident for training sets of small sizes.



(a) (b)

Figure 8: Comparison between a given substitution matrix (a) and the matrix computed with SODA (b) for $n = 100$.

accession	n^o sequences	max. length
RF00032	64	27
RF00260	35	51
RF00436	24	55
RF00164	29	43
RF00480	647	52

Table 6: Summary of the data set of RNA sequences

5.6 Learning to Parse

Lastly, we analyze the RNA secondary structure prediction problem: given an RNA sequence, the task is to predict the basepairs in the sequence. With weighted context-free grammars, this prediction can be accomplished by parsing the RNA sequence. To describe basepairs in RNA sequences, we used the G6 grammar in Dowell and Eddy (2004), which we call $G = \{\Upsilon, \Sigma_x, R, S, \theta\}$, where $\Upsilon = \{S, L, F\}$, $\Sigma_x = \{\mathtt{a}, \mathtt{u}, \mathtt{g}, \mathtt{c}\}$, and $R = \{S \to LS|L, L \to \mathtt{a}F\mathtt{u}|\mathtt{u}F\mathtt{a}|\mathtt{g}F\mathtt{c}|\mathtt{c}F\mathtt{g}|\mathtt{g}F\mathtt{u}|\mathtt{u}F\mathtt{g}|\mathtt{a}|\mathtt{u}|\mathtt{c}|\mathtt{g}, F \to \mathtt{a}F\mathtt{u}|\mathtt{u}F\mathtt{a}|\mathtt{g}F\mathtt{c}|\mathtt{c}F\mathtt{g}|\mathtt{g}F\mathtt{u}|\mathtt{u}F\mathtt{g}|LS\}$. We consider RNA sequences of five families (see Tab. 6) extracted from the Rfam database (Griffiths-Jones et al., 2003). We use sequences including only standard basepairs, that is, a–u, c–g, and g–u.

Results for the experiments conducted with Z-score with constraints, with a generative model, and with the hidden Markov perceptron in a 5-fold cross validation setting are shown. Weights of the grammar are optimized with a training set, and structures associated to sequences in the test set are predicted by the Viterbi algorithm. For the Z-score with constraints, the best results obtained varying the regularization parameter are reported. For the HMP, values ranging from T=100 to T=1000 are used as the number of iterations, and the best result is shown. For the generative model, parameters are estimated with Laplace smoothing.

We measure the performance of the methods in terms of sensitivity and specificity of predicted basepairs. The sensitivity is defined as the number of correctly predicted basepairs over the number

of true basepairs, and the specificity is the number of correctly predicted basepairs over the number of predicted basepairs. In Tab. 7, the values of sensitivity and specificity corresponding to the maximum product, are shown for each algorithm. The Z-score and the HMP have comparable performances and generally outperform the generative approach. For the Z-score approach also the average number of constraints is shown: it is worth noting that only very few constraints are needed for each family, often less than the number of iterations in the HMP by an order of magnitude or more.

6. Statistical Learning Analysis

We present here two learning theory results. The first one is specific for the ranking loss in any algorithm using this hypothesis space, and hence covers the SODA algorithm. The second one applies to any algorithm using the zero-one loss with this hypothesis class, and hence covers most previous approaches.

	Z-score (constr)			**Generative**		**HMP**	
accession	sens.	spec.	n^o constraints	sens.	spec.	sens.	spec.
RF00032	100	95.98	2.0	100	95.53	100	95.59
RF00260	98.77	94.80	6.0	98.97	100	98.57	98.90
RF00436	91.11	90.61	27.6	44.16	53.30	90.27	86.53
RF00164	76.14	73.47	37.8	65.51	62.55	87.06	78.32
RF00480	99.08	89.89	78.2	99.88	86.43	98.83	94.78

Table 7: Prediction on 5-fold cross validation. Average sensitivity and specificity are shown.

6.1 Rademacher Theory for SODA

Here we present a Rademacher bound for the SODA showing that learning based on this upper bound on the relative rank is effectively achieved. For full generality, we want our bound to hold also in the case where the matrices μ and C are estimated by sampling, as we suggested in subsection 3.2.2. We also provide some experimental results for this in the following subsection. Hence, our bound needs to account for finiteness in two ways. First of all for each input-output pair only a limited number n of incorrect outputs may be considered to estimate μ and C; secondly only a finite number ℓ of input-output pairs is given in the training set.

In appendix C, we prove the following theorem. It shows that the empirical expectation of the estimated loss (estimated by computing C and b by random sampling) is a good approximate upper bound for the expected loss. Hereby it is good to keep in mind that this loss itself is an upper bound for the relative ranking loss, such that the Rademacher bound is also a bound on the expectation of the relative ranking loss.

Theorem 7 (Rademacher bound for SODA) *With probability at least* $1-\delta$ *over the joint of the random sample* $\mathcal{T}$ *and the random samples from the output space for each* $(\boldsymbol{x},\bar{\boldsymbol{y}}) \in \mathcal{T}$ *that are taken to approximate the matrices* C^* *and* b^**, the following bound holds for any* θ *with squared norm smaller than* c*:*

$$E_{(\boldsymbol{x},\bar{\boldsymbol{y}})}\left\{\mathcal{L}_\theta^{RRU}(\boldsymbol{x},\bar{\boldsymbol{y}})\right\} \quad \leq \quad \widehat{E}_{(\boldsymbol{x},\bar{\boldsymbol{y}})}\left\{\widehat{\mathcal{L}}_\theta^{RRU}(\boldsymbol{x},\bar{\boldsymbol{y}})\right\}$$

$$+ \quad \widehat{E}_{(\boldsymbol{x},\bar{\boldsymbol{y}})}\left\{\hat{\mathfrak{R}}_{1,(\boldsymbol{x},\bar{\boldsymbol{y}})}\right\} + \hat{\mathfrak{R}}_2$$
$$+ \quad 3M\sqrt{\frac{\log(2\ell/\delta)}{2n}} + 3M\sqrt{\frac{\log(2/\delta)}{2\ell}}.$$

whereby we assume that the number of random samples for each training pair is equal to n.

The Rademacher complexity terms $\hat{\mathfrak{R}}_{1,(\boldsymbol{x},\bar{\boldsymbol{y}})}$ and $\hat{\mathfrak{R}}_2$ decrease with $\frac{1}{\sqrt{n}}$ and $\frac{1}{\sqrt{\ell}}$ respectively, such that the bound becomes tight for increasing n and ℓ, as long as n grows faster than $\log(\ell)$.

For details relating to the exact value of the Rademacher complexity terms, the value of the constant M, and the proofs, we refer to the appendix C.

6.2 PAC Bound

In appendix D, we prove the following theorem that applies to a generic DLA: given a training set of sample pairs $(\mathbf{x}^i, \overline{\mathbf{y}}^i)$, can we learn to predict the output for a previously unseen observation? For example, given a training set of aligned protein sequences, can we learn how to align a previously unseen pair? Or, given a training set of correctly parsed sentences, can we learn how to parse a previously unseen sentence? To be clear, in this section we consider the zero-one loss only, which has been considered most often in previous work on structured output learning.

DLAs directly learn the model parameters such that the accuracy of the prediction is somehow optimized. All these algorithms are in some sense empirical risk minimizers, in that they optimize the prediction performance on a training set. However till now there have been few works trying to address the question whether a small empirical risk guarantees a small expected risk. A first generalization bound has been developed by Collins for the case of the perceptron algorithm (Collins, 2002a) and a capacity bound in terms of covering numbers for the maximum margin approach has been proposed by Taskar et al. (2003). These bounds have subsequently been reconsidered in McAllester (2007) and have been improved in order to achieve consistency for any arbitrary loss. In this paper we answer the learnability question affirmatively from another point of view, independent of the learning approach taken, and we propose a new PAC bound which makes use of a result which bounds the cardinality of the hypothesis space of prediction functions derived from DLAs.

We go back to the original problem of structured output learning. Given is a training set $\mathcal{T} = \{(\mathbf{x}^1, \overline{\mathbf{y}}^1), (\mathbf{x}^2, \overline{\mathbf{y}}^2), \ldots, (\mathbf{x}^\ell, \overline{\mathbf{y}}^\ell)\}$ of observation-output pairs, with observations $\mathbf{x}^i \in \mathcal{X}$ and outputs $\overline{\mathbf{y}}^i \in \mathcal{Y}$ jointly drawn i.i.d. from an unspecified probability distribution $P(\mathbf{x}, \mathbf{y})$. Based on $\mathcal{T}$ we want to infer a prediction function $h_\theta : \mathcal{X} \to \mathcal{Y}$ such that the probability $P(h_\theta(\mathbf{x}) = \overline{\mathbf{y}})$ of an observation-output pair $(\mathbf{x}, \overline{\mathbf{y}})$ with $h_\theta(\mathbf{x}) = \overline{\mathbf{y}}$ is as large as possible. For learnability, the choice of h_θ should be restricted to a limited hypothesis space, and DLAs provide one way to achieve this.

Our hypothesis space $\mathcal{H}$ is the space containing all prediction functions h_θ with $\theta \in \mathbb{R}^d$, defined as:

$$h_\theta(\mathbf{x}) \quad = \quad \arg\max_{\mathbf{y}\in\mathcal{Y}} \; s_\theta(\mathbf{x}, \mathbf{y}) = \arg\max_{\mathbf{y}\in\mathcal{Y}} \; \theta^T \phi(\mathbf{x}, \mathbf{y}),$$

and this for a fixed feature map ϕ with integer features between 0 and C.

For this hypothesis space we can prove (in appendix D) the following theorem.

Theorem 8 (On the PAC-learnability of structured output prediction) *Given a hypothesis space $\mathcal{H}$ of prediction functions h_θ as defined above. Furthermore, consider a training set $\mathcal{T} =$*

$\{(\boldsymbol{x}^1,\bar{\boldsymbol{y}}^1),(\boldsymbol{x}^2,\bar{\boldsymbol{y}}^2),\ldots,(\boldsymbol{x}^\ell,\bar{\boldsymbol{y}}^\ell)\}$ of observation-output pairs, sampled i.i.d. from a fixed but unknown distribution. Then, with probability at least $1-\delta$ over the random sample $\mathcal{T}$, for any $h_\theta \in \mathcal{H}$ for which $h_\theta(\boldsymbol{x}^i)=\bar{\boldsymbol{y}}^i$ for all $(\boldsymbol{x}^i,\bar{\boldsymbol{y}}^i)\in\mathcal{T}$ the expected risk can be bounded as:

$$E_{(\boldsymbol{x},\bar{\boldsymbol{y}})\sim\mathcal{D}}\{h_\theta(\boldsymbol{x})\neq\bar{\boldsymbol{y}}\}\leq\frac{d^2\log(2C)-(d-2)\log(d-2)+d-\log(\delta)}{\ell}.$$

We can thus conclude that learning is guaranteed as soon as $\ell >> d^2\log(2C)$.

This result proves that learning based on DLAs can be achieved effectively, and d and C are the factors that are relevant in determining the learning rate. Importantly this bound holds regardless of the method used to estimate the parameter vector θ. Interestingly, it suggests that the number C (bounded by to the number of cliques for DLAs derived from PGMs) is less important than the number of parameters d.

Note that this bound only holds for the realizable case, such that its practical relevance is limited. Furthermore, unlike the results from McAllester (2007), the bound does not depend on the norm of the weight vector θ, making it loose for small values of $\|\theta\|$. Nevertheless, we believe it is of interest due to the simplicity of its derivation based on results from combinatorics and basic PAC theory.

7. Conclusions

We have presented a formal framework for learning to predict over structured output spaces. The hypothesis space we consider is based on linear scoring functions, much like most previous approaches to this problem.

The distribution of this linear scoring function over all possible outputs contains information that we can use to train the parameters of the learning algorithm. We can compute efficiently the first two moments of this distribution, and we use them to derive convex objective functions for parameter optimization.

In this way, we have derived two new efficient algorithms for structured output prediction that rely on these statistics, both of which can be solved by solving one linear system of equations.

Interestingly, and thanks to the use of the moments, one of the proposed objective functions (SODA) represents a convex upper bound on the relative ranking loss: the fraction of outputs from the output space that rank better than the correct output. Thanks to this property, SODA naturally and adequately deals with the infeasible case where there exists no parameter setting for which the correct given pairs are optimal. We justify this fact theoretically, providing a Rademacher bound, and experimentally, reporting results that are competitive with existing methods, and better than other methods in the infeasible case.

Acknowledgments

We are most grateful to Nobuhisa Ueda since without him the section on SCFG's would not have been possible. We also thank the anonymous reviewers for providing us with their valuable comments. This work was partially supported by NIH grant R33HG00 3070-01, the EU project SMART and the PASCAL network of excellence. The work of Nello Cristianini is partially supported by a Royal Society Wolfson Merit Award.

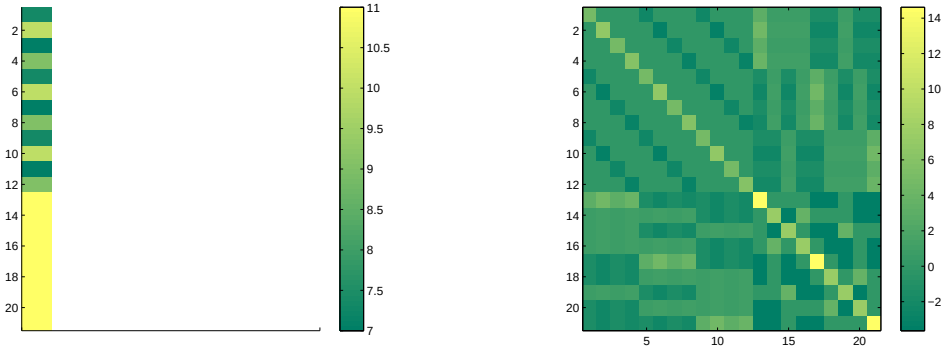
Appendix A. Proof of Proposition 3

The number of DP routines needed to compute μ and C are $7n_o+6$.

In fact in general in the mean vector μ there are n_o+1 different values. All the elements associated to transition probabilities assume the same values while for emission probability $\mu^e_{pq}=\mu^e_{ef}$, $\forall\, q=f$.

We analyze the structure of the matrix C. It is a symmetric block matrix made basically by three components: the block associated to emission probabilities, that of transition probabilities and that relative to mixed terms. To compute it $6n_o+5$ DP routines are required. In the emission part there are $2n_o$ possible different values since $c^e_{pq}=c^e_{ef}$, $\forall q=f$, $c^e_{pqp'q'}=0$, $\forall q\neq q'$ and $c^e_{pqp'q'}=c^e_{efe'f'}$ $\forall q=q'=f=f'$. In the transition block there are only 5 possible different values. In particular for the second order moments, it holds that $c^t_{pz}=c^t_{eg}$, $\forall p=z=e=g$ and $c^t_{pz}=c^t_{eg}$, $\forall p=e$, $z=g$ and $p\neq z$. For the remaining three values there holds that $c^t_{pzp'z'}=0$, $\forall p\neq p'$, $z\neq z'$, $c^t_{pzp'z'}=c^t_{ege'g'}$, $\forall p=p'$, $z\neq z'$, $e=e'$, $g\neq g'$ and $c^t_{pzp'z'}=c^t_{ege'g'}$, $\forall p\neq p'$, $z=z'$, $e\neq e'$, $g=g'$. The block relative to mixed terms is made of $4n_o$ possible different value. In fact there are n_o values $c^{et}_{pqp'z}$ with $p=p'=z'$, n_o values $c^{et}_{pqp'z}$, with $p=p'$, $p'\neq z'$, n_o values $c^{et}_{pqp'z}$, with $p=z'$, $p'\neq z'$ and n_o values $c^{et}_{pqp'z}$, with $p\neq p'$, $z\neq z'$.

The redundancy in the structure of matrix C and of the vector μ can be observed in Fig. 9 for an HMM with $n_s=3$ and $n_o=4$.



(a) (b)

Figure 9: Mean vector and covariance matrix for an HMM with $n_s=3$ and $n_o=4$.

Appendix B. Solving Linear Systems for Large Feature Spaces

This paragraph provides a brief description of an approach for solving the linear systems $C^*\theta=b^*$ and $\bar{C}\theta=\bar{b}$ avoiding to store the entire matrices C^* and $\bar{C}$. This approach is suited to sequence labeling problems and HMM features and it is particularly effective for problems when n_h, the size of the hidden state alphabet, is small and n_o, the size of the observation alphabet, is large.

We describe the procedure to solve $C^*\theta=b^*$. In fact it subsumes the method for solving $\bar{C}\theta=\bar{b}$. The main idea behind this procedure is that exploiting the structure of C^* we can store just a part of it and compute the optimal parameter vector θ effectively.

The matrix C^* in case of sequence labeling and HMMs features is sparse and redundant. In fact this matrix is given by the sum of two parts: $\bar{C}=\sum_i C_i$ and $BB^T=\sum_i b_i b_i^T$. We first consider the

first part $\bar{C} = \sum_i C_i$. Each C_i is very sparse and has a regular structure as discussed in appendix A. Then also the matrix $\bar{C}$ has the same structure, that is, is a matrix made by four block:

$$\bar{C} = \begin{pmatrix} E & M \\ M^T & T \end{pmatrix}.$$

Here E denotes the block associated to emission probabilities, T that corresponding to transition probabilities and M that relative to mixed terms. We are interested in finding the inverse of the matrix $\bar{C}$ without storing it entirely. Note that in many situations (e.g., sequence labeling problems for text analysis such as NER or POS) the emission part represents the main bottleneck in the computation of the inverse since its size is dependent on n_o (e.g., the size of the dictionary). The size of the transition part instead is usually moderate since it is given by n_h^2 (e.g., the number of different tags). The inverse of $\bar{C}$ can be computed by:

$$\bar{C}^{-1} = \begin{pmatrix} E^{-1} + E^{-1}MP^{-1}M^TE^{-1} & -E^{-1}MP^{-1} \\ -P^{-1}M^TE^{-1} & P^{-1} \end{pmatrix}$$

where $P = T - M^TE^{-1}M$ represent the Schur complement of E. The inverse of the matrix E can be computed easily due to the structure of the matrix E. In fact E is a block matrix:

$$E = \begin{pmatrix} E_d & E_o & E_o & \cdots & E_o \\ E_o & E_d & E_o & \cdots & E_o \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ E_o & \cdots & E_o & E_o & E_d \end{pmatrix}$$

where E_d and E_o are both diagonal matrices. Therefore we can rewrite the matrix E as:

$$\begin{aligned} E = & \begin{pmatrix} E_d - E_o & 0 & 0 & \cdots & 0 \\ 0 & E_d - E_o & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & E_d - E_o \end{pmatrix} + \begin{pmatrix} I \\ I \\ \vdots \\ I \end{pmatrix} E_o \begin{pmatrix} I & I & \cdots & I \end{pmatrix} \\ = & \quad D + H^T E_o H. \end{aligned}$$

Then the inverse can be computed easily considering the formula for the inverse of a sum of matrices:

$$E^{-1} = D^{-1} - D^{-1}H^T(I + E_oHD^{-1}H^T)^{-1}E_oHD^{-1}$$

where D is a diagonal matrix. Due to the special structure of D, H and E, it turns out that the inverse of E is also a block matrix with similar structure of E, that is,

$$E^{-1} = \begin{pmatrix} \bar{E}_d & \bar{E}_o & \bar{E}_o & \cdots & \bar{E}_o \\ \bar{E}_o & \bar{E}_d & \bar{E}_o & \cdots & \bar{E}_o \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \bar{E}_o & \cdots & \bar{E}_o & \bar{E}_o & \bar{E}_d \end{pmatrix}$$

where $\bar{E}_d = (E_d - E_o)^{-1}$ and $\bar{E}_o = \bar{E}_d(I + E_oN_H\bar{E}_d)^{-1}\bar{E}_d$ (N_H is a diagonal matrix with elements on the main diagonal equal to n_h). Then it is not necessary to compute and store the entire matrix E^{-1} but only the small blocks $\bar{E}_d$ and $\bar{E}_o$.

Once the matrix E^{-1} has been obtained then the computation of the Schur complement P and its inverse it is straightforward. This is not a time consuming procedure since its size n_h^2 is typically small and E^{-1} is very sparse. Due to the redundancy and the particular structure of the matrix M we can also compute quite easily all the other terms. In particular the matrix obtained by $E^{-1}MP^{-1}M^TE^{-1}$ is a block matrix made by $n_h \times n_h$ equal blocks. Then it suffices to compute and to store just one of each block.

The inverse of the matrix $\bar{C}$ has then been obtained and we can use it directly to compute the solution of the linear system for the Z-score approach $\theta = \bar{C}^{-1}\bar{b}$. Instead if we want to obtain the optimal parameter vector associated to SODA we must compute the solution of the linear system $(\bar{C}+BB^T)\theta = b^*$. In practice what we need is a method to perform n rank one updates (one for each sample in the training set) of the inverse of the matrix $\bar{C}$ without storing the matrices $\bar{C}^{-1}$ and BB^T entirely. We can use the Sherman-Morrison-Woodbury formula:

$$\bar{C}+BB^T = \bar{C}^{-1} - \bar{C}^{-1}B(I+B^T\bar{C}^{-1}B)B^T\bar{C}^{-1}$$

to calculate the solution of our linear system:

$$\theta = (\bar{C}+BB^T)^{-1}b^* = \bar{C}^{-1}b^* - \bar{C}^{-1}B(I+B^T\bar{C}^{-1}B)B^T\bar{C}^{-1}b^*.$$

In practice we first compute $z = \bar{C}^{-1}b^*$ and use this value to solve the linear system by Cholesky decomposition:

$$(I+B^T\bar{C}^{-1}B)t = B^Tz.$$

Note that the cost of this operation is $O(n^3)$ but it is usually moderate since $n << d$. The computational cost here is dominated by the calculation of z since in theory it requires d^3 multiplications. However in practice this cost is still reasonable since $\bar{C}^{-1}$ tend to be sparse. Once we have computed t we can obtain our solution for SODA simply by $\theta = z - \bar{C}^{-1}Bt$.

Appendix C. Proof of the Rademacher Bound

We consider two types of randomness in our bound: the randomness in choosing the finite sample of training input-output pairs, and the randomness in sampling from the output space for each of the training inputs. Our aim is to provide a learning theory bound for the expected relative rank of the score $s_\theta(\mathbf{x},\bar{\mathbf{y}})$ among all scores $s_\theta(\mathbf{x},\mathbf{y})$ for all $\mathbf{y} \in \mathcal{Y}$.

More exactly, we are interested in bounding $E_{(\mathbf{x},\bar{\mathbf{y}})}\left\{\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}})\right\}$ where the value of the loss $\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}}) = E_{\mathbf{y}}\left\{\left[\theta^T\left(\phi(\mathbf{x},\bar{\mathbf{y}})-\phi(\mathbf{x},\mathbf{y})\right)-1\right]^2\right\}$ is known to be an upper bound on the relative rank of the score of $(\mathbf{x},\bar{\mathbf{y}})$ among all scores of $(\mathbf{x},\mathbf{y})$ for all possible $\mathbf{y}$ (see Theorem 5).

For clarity, let us first consider a fixed training pair $(\mathbf{x},\bar{\mathbf{y}})$. We will derive a Rademacher bound that shows that the loss function $\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}})$ is approximately upper bounded by its empirical estimate $\widehat{\mathcal{L}}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}}) = \widehat{E}_{\mathbf{y}}\left\{\left[\theta^T\left(\phi(\mathbf{x},\bar{\mathbf{y}})-\phi(\mathbf{x},\mathbf{y})\right)-1\right]^2\right\}$, obtained by averaging over a random sample of n values of $\mathbf{y}$. In particular we will show that with a probability of at least $1-\delta_1$ over the random sample of size n:

$$\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}}) \leq \widehat{\mathcal{L}}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}}) + \hat{\mathfrak{R}}_{1,(\mathbf{x},\bar{\mathbf{y}})} + 3M\sqrt{\frac{\log(1/\delta_1)}{2n}},$$

with $\hat{\mathfrak{R}}_{1,(\mathbf{x},\bar{\mathbf{y}})}$ an empirical Rademacher complexity term. The constant M is an upper bound on the value of $\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}})$ valid for all allowable θ, and is a finite number. Such an upper bound can be computed as $M = (C\sqrt{dc}+1)^2$, by considering the constraint $\|\theta\|^2 \leq c$ and the fact that for all d features $0 \leq \phi_i(\mathbf{x},\mathbf{y}) \leq C$.

Second, we will show that the expectation of $\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}})$ over $(\mathbf{x},\bar{\mathbf{y}})$ is approximately upper bounded by its empirical expectation over the training set of size l. We will show that with probability at least $1-\delta_2$ over the training set $\mathcal{T}$ of size ℓ,

$$E_{(\mathbf{x},\bar{\mathbf{y}})}\left\{\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}})\right\} \leq \widehat{E}_{(\mathbf{x},\bar{\mathbf{y}})}\left\{\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}})\right\} + \hat{\mathfrak{R}}_2 + 3M\sqrt{\frac{\log(1/\delta_2)}{2\ell}},$$

with $\hat{\mathfrak{R}}_2(\mathcal{T})$ an empirical Rademacher complexity term, and with the same constant M.

Putting these two partial results together with $\delta_1 = \frac{\delta}{2\ell}$ and $\delta_2 = \frac{\delta}{2}$, we have shown the following theorem:

Theorem 9 (Rademacher bound for SODA) *With probability at least $1-\delta_2-\ell\delta_1 = 1-\delta$ over the joint of the random sample $\mathcal{T}$ and the random samples from the output space for each $(\boldsymbol{x},\bar{\boldsymbol{y}}) \in \mathcal{T}$, the following bound holds for any θ with squared norm smaller than c:*

$$\begin{aligned} E_{(\boldsymbol{x},\bar{\boldsymbol{y}})}\left\{\mathcal{L}_\theta^{RRU}(\boldsymbol{x},\bar{\boldsymbol{y}})\right\} &\leq \widehat{E}_{(\boldsymbol{x},\bar{\boldsymbol{y}})}\left\{\widehat{\mathcal{L}}_\theta^{RRU}(\boldsymbol{x},\bar{\boldsymbol{y}})\right\} \\ &+ \widehat{E}_{(\boldsymbol{x},\bar{\boldsymbol{y}})}\left\{\hat{\mathfrak{R}}_{1,(\boldsymbol{x},\bar{\boldsymbol{y}})}\right\} + \hat{\mathfrak{R}}_2 \\ &+ 3M\sqrt{\frac{\log(2\ell/\delta)}{2n}} + 3M\sqrt{\frac{\log(2/\delta)}{2\ell}}. \end{aligned}$$

The first term on the right hand side of the inequality is the empirical risk, which is minimized on the training set. The next two terms are Rademacher complexity terms, and we will see below that these decrease to zero with increasing ℓ and n. Also the last two terms decrease to zero with increasing ℓ and n, as long as n is chosen to increase faster than $\log(\ell)$.

Both these partial results can be derived by using the generalization error bound in Bartlett and Mendelson (2002, Theorem 2) and applying the McDiarmid's concentration inequality (McDiarmid, 1989). In the following we show how to compute upper bounds on the empirical Rademacher complexities $\hat{\mathfrak{R}}_{1,(\mathbf{x},\bar{\mathbf{y}})}$ and $\hat{\mathfrak{R}}_2$.

C.1 Rademacher Bound for the Relative Rank of a Single Pair

Given a training pair $(\mathbf{x},\bar{\mathbf{y}})$ and a set $\{\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_n\}$ of n randomly sampled values for $\mathbf{y}$ corresponding to $\mathbf{x}$. For notational convenience, let us denote $\varphi_j = \phi(\mathbf{x},\bar{\mathbf{y}}) - \phi(\mathbf{x},\mathbf{y}_j)$. Then we can write the empirical estimate of the loss as

$$\widehat{\mathcal{L}}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}}) = \frac{1}{n}\sum_{j=1}^{n}\left(\theta^T\varphi_j - 1\right)^2.$$

Using this notation, and with σ a vector of length n containing the independently distributed Rademacher variables σ_j being uniformly distributed over 1 and -1, the Rademacher complexity term $\hat{\mathfrak{R}}_{1,(\mathbf{x},\bar{\mathbf{y}})}$ can be written and bounded as:

$$\hat{\mathfrak{R}}_{1,(\mathbf{x},\bar{\mathbf{y}})} = E_\sigma\left\{\max_\theta\left|\frac{2}{n}\sum_{j=1}^{n}\sigma_j\left(\theta^T\varphi_j - 1\right)^2\right|\right\}$$

$$
\begin{aligned}
&= E_\sigma\left\{\max_\theta\left|\frac{2}{n}\sum_{j=1}^{n}\sigma_j\left((\theta^T\varphi_j)^2-2(\theta^T\varphi_j)+1\right)\right|\right\}\\
&\leq E_\sigma\left\{\max_\theta\frac{2}{n}\left(\left|\sum_{j=1}^{n}\sigma_j(\theta^T\varphi_j)^2\right|+\left|\sum_{j=1}^{n}2\sigma_j(\theta^T\varphi_j)\right|+\left|\sum_{j=1}^{n}\sigma_j\right|\right)\right\}. \qquad (12)
\end{aligned}
$$

The first equality is the definition of the empirical Rademacher complexity, and the second equality is a trivial rewriting. The first inequality holds since the absolute value of the sum is smaller than or equal to the sum of absolute values. We now first use the fact that the maximum of a sum of functions is smaller than or equal to the sum of their maxima to show that:

$$
\begin{aligned}
(12) \quad &\leq \frac{2}{n}E_\sigma\left\{\max_\theta\left|\sum_{j=1}^{n}\sigma_j(\theta^T\varphi_j)^2\right|+\max_\theta\left|\sum_{j=1}^{n}2\sigma_j(\theta^T\varphi_j)\right|+\max_\theta\left|\sum_{j=1}^{n}\sigma_j\right|\right\}\\
&= \frac{2}{n}E_\sigma\left\{\sqrt{\max_\theta\left(\sum_{j=1}^{n}\sigma_j(\theta^T\varphi_j)^2\right)^2}\right\}\\
&\quad +\frac{2}{n}E_\sigma\left\{\sqrt{\max_\theta\left(\sum_{j=1}^{n}2\sigma_j(\theta^T\varphi_j)\right)^2}\right\}+\frac{2}{n}E_\sigma\left\{\sqrt{\max_\theta\left(\sum_{j=1}^{n}\sigma_j\right)^2}\right\}. \qquad (13)
\end{aligned}
$$

Here we used the fact that the absolute value is the square root of the square, and that the square root of a positive function is maximized when that function itself is maximized. We proceed by rewriting this expression using bracket notation, $\langle \mathbf{a}, \mathbf{b}\rangle$ denoting the inner product between vectors (or matrices) **a** and **b**. Furthermore, we use the fact that the maximum of a sum (or expectation) is smaller than or equal to the sum (or expectation) of the maxima of the individual sums, to show that:

$$
\begin{aligned}
(13) \quad &\leq \frac{2}{n}\sqrt{E_\sigma\left\{\max_\theta\sum_{j,k=1}^{n}\sigma_j\sigma_k\langle\theta\theta^T,\varphi_j\varphi_j^T\rangle\langle\theta\theta^T,\varphi_k\varphi_k^T\rangle\right\}}\\
&\quad +\frac{2}{n}\sqrt{E_\sigma\left\{\max_\theta\sum_{j,k=1}^{n}4\sigma_j\sigma_k\langle\theta,\varphi_j\rangle\langle\theta,\varphi_k\rangle\right\}}+\frac{2}{n}\sqrt{E_\sigma\left\{\sum_{j,k=1}^{n}\sigma_j\sigma_k\right\}}. \qquad (14)
\end{aligned}
$$

We now invoke the Cauchy-Schwartz inequality, and use the fact that $\|\theta\|^2 \leq c$ and hence $\|\theta\theta^T\|^2 \leq c^2$, to show that:

$$
\begin{aligned}
(14) \quad &\leq \frac{2}{n}\sqrt{E_\sigma\left\{\sum_{j,k=1}^{n}\sigma_j\sigma_k c^2\|\varphi_j\|^2\|\varphi_k\|^2\right\}}\\
&\quad +\frac{2}{n}\sqrt{E_\sigma\left\{\sum_{j,k=1}^{n}4c\sigma_j\sigma_k\|\varphi_j\|\|\varphi_k\|\right\}}+\frac{2}{n}\sqrt{E_\sigma\left\{\sum_{j,k=1}^{n}\sigma_j\sigma_k\right\}}. \qquad (15)
\end{aligned}
$$

Since for $i \neq k$, there holds that $E_\sigma\left\{\sigma_j\sigma_k\right\} = 0$ and $E_\sigma\left\{\sigma_j^2\right\} = 1$, we can finally write that:

$$(15) \quad = \quad \frac{2}{\sqrt{n}}\left(c\sqrt{\frac{1}{n}\sum_{j=1}^{n}\|\varphi_j\|^4} + 2\sqrt{c}\sqrt{\frac{1}{n}\sum_{j=1}^{n}\|\varphi_j\|^2} + 1\right).$$

In summary, we have found the following upper bound on the first empirical Rademacher complexity:

Proposition 10 (Rademacher complexity $\hat{\mathfrak{R}}_{1,(\mathbf{x},\bar{\mathbf{y}})}$) *The Rademacher complexity term $\hat{\mathfrak{R}}_{1,(\boldsymbol{x},\bar{\boldsymbol{y}})}$ can be bounded as:*

$$\hat{\mathfrak{R}}_{1,(\boldsymbol{x},\bar{\boldsymbol{y}})} \leq \frac{2}{\sqrt{n}}\left(c\sqrt{\frac{1}{n}\sum_{j=1}^{n}\|\varphi_j\|^4} + 2\sqrt{c}\sqrt{\frac{1}{n}\sum_{j=1}^{n}\|\varphi_j\|^2} + 1\right),$$

which, given the boundedness of $\|\varphi_j\|$, decreases to zero as n increases to infinity, as required.

C.2 Rademacher Complexity for the Empirical Expectation of the Loss

Given a randomly sampled training set $\mathcal{T} = \{(\mathbf{x}^1,\bar{\mathbf{y}}^1),(\mathbf{x}^2,\bar{\mathbf{y}}^2),\ldots,(\mathbf{x}^\ell,\bar{\mathbf{y}}^\ell)\}$. The empirical expectation of the loss can be written as:

$$\begin{aligned}\widehat{E}_{(\mathbf{x},\bar{\mathbf{y}})}\left\{\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}})\right\} &= \frac{1}{\ell}\sum_{i=1}^{\ell}\mathcal{L}_\theta^{RRU}(\mathbf{x}^i,\bar{\mathbf{y}}^i)\\ &= \frac{1}{\ell}\sum_{i=1}^{\ell}\left(\frac{1}{N_i}\sum_{j=1}^{N_i}(\theta^T\varphi_j^i - 1)^2\right),\end{aligned}$$

where $\varphi_j^i = \phi(\mathbf{x}^i,\mathbf{y}_j^i) - \phi(\mathbf{x}^i,\bar{\mathbf{y}}^i)$ and N_i is the cardinality of the output space corresponding to $\mathbf{x}^i$.

For notational convenience, let us introduce the matrix Φ^i containing all vectors ${\varphi_j^i}^T$ as its rows. Then we can rewrite the expected loss function in a more compact form as:

$$\begin{aligned}\widehat{E}_{(\mathbf{x},\bar{\mathbf{y}})}\left\{\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}})\right\} &= \frac{1}{\ell}\sum_{i=1}^{\ell}\frac{\|\Phi^i\theta - \mathbf{1}\|^2}{N_i}\\ &= \frac{1}{\ell}\sum_{i=1}^{\ell}\frac{\langle\theta\theta^T,{\Phi^i}^T\Phi^i\rangle - 2\langle\theta,{\Phi^i}^T\mathbf{1}\rangle + \langle\mathbf{1},\mathbf{1}\rangle}{N_i}.\end{aligned}$$

We have rewritten this in a form that contains a term linear in $\theta\theta^T$, a term linear in θ, and a constant term. It is exactly this decomposition of the empirical expectation of the loss that has allowed us to derive a bound on the Rademacher term $\hat{\mathfrak{R}}_{1,(\mathbf{x},\bar{\mathbf{y}})}$, so we can follow the same principles here. We omit the details here, and just state the result:

Proposition 11 (Rademacher complexity $\hat{\mathfrak{R}}_2$) *The Rademacher complexity term $\hat{\mathfrak{R}}_2$ can be bounded as:*

$$\hat{\mathfrak{R}}_2 \leq \frac{2}{\sqrt{\ell}}\left(c\sqrt{\frac{1}{\ell}\sum_{i=1}^{\ell}\left(\frac{\sum_{j,k=1}^{N_i}({\varphi_j^i}^T\varphi_k^i)^2}{N_i^2}\right)} + 2\sqrt{c}\sqrt{\frac{1}{\ell}\sum_{i=1}^{\ell}\left\|\frac{\sum_{j=1}^{N_i}\varphi_j^i}{N_i}\right\|^2} + 1\right),$$

which again decreases to zero as ℓ increases to infinity.

Appendix D. Proof of the PAC Bound

This section contains the proof of the PAC bound stated in subsection 6.2.

D.1 Bounding the Effective Cardinality of the Hypothesis Space

The number of possible functions mapping the input space on the output space is potentially huge: $|\mathcal{Y}|^{|\mathcal{X}|}$ for an observation space of size $|\mathcal{X}|$ and an output space of size $|\mathcal{Y}|$. To make this more concrete, for the HMM prediction problem discussed earlier, this is equal to $\left(n_h^m\right)^{n_o^m} = n_h^{mn_o^m}$, which is doubly exponential in the length of the sequences m.

It would clearly be impossible to achieve learning if we had to consider all of these possible functions mapping observations onto outputs. However, we will show that the hypothesis class of prediction functions defined above contains only a very small subset of these functions. This means that, while the cardinality of functions h_θ is infinite (one such function for each $\theta \in \mathbb{R}^d$), the effective cardinality is low, since many of these functions are equivalent. We will subsequently use this upper bound on the effective cardinality to obtain a PAC bound on the generalization.

To upper bound the effective cardinality of the hypothesis space $\mathcal{H}$, we borrow and reformulate the so-called *few inference functions theorem* by Elizalde (to appear) in the terminology of the present paper:

Theorem 12 (Elizalde) *Let d and C be fixed positive integers. Let $\phi : \mathcal{X} \times \mathcal{Y} \to \{0,1,\ldots,C\}^d$ be a fixed function (called the feature map). Then the hypothesis space $\mathcal{H}$ defined as $\mathcal{H} = \left\{h_\theta | h_\theta(\boldsymbol{x},\boldsymbol{y}) = \arg\max_{\boldsymbol{y}} \theta^T \phi(\boldsymbol{x},\boldsymbol{y}) | \theta \in \mathbb{R}^d\right\}$ has an effective cardinality of at most*

$$K = \frac{2^{d^2-d+1}}{(d-2)!} C^{d(d-1)}.$$

that is, the number of different *prediction functions in $\mathcal{H}$ is at most K.*

D.2 A PAC Bound for Learning Prediction Functions

Based on the effective cardinality of the hypothesis space we can now derive a PAC bound on the expected risk. Let us derive the bound for the case where the empirical risk is equal to zero. Deriving a PAC bound in the case of nonzero empirical risk is a well-known variation (Vapnik, 1998), and we will not discuss it in the current paper.

The probability of classifying all the ℓ training examples correctly with any fixed prediction function is:

$$P(\text{all } \ell \text{ correct } | p) \leq (1-p)^\ell \leq \exp(-\ell p)$$

where p is the expected risk for this prediction function. However in general, the prediction function is chosen from the hypothesis space $\mathcal{H}$. The probability over the sample that *any of K prediction functions* with an expected error rate of at least p faultlessly performs on all training examples is bounded by

$$P(\text{all } \ell \text{ correct, for any prediction function } \in \mathcal{H} | p) \quad \leq \quad K \exp(-\ell p),$$

where K is the effective cardinality of $\mathcal{H}$. Hence, the probability to get a zero training set error, for any of the prediction functions and thus for any of the parameter values, is at most $K \exp(-\ell p)$,

where p is the minimal error probability. Thus, we found an upper bound which holds with confidence at least δ for the expected error rate p as:

$$p \leq \frac{\log(K) - \log(\delta)}{\ell}$$

As we have seen in Theorem 12, the effective cardinality of $\mathcal{H}$ is upper bounded by $K = \frac{2^{d^2-d+1}}{(d-2)!} C^{d(d-1)}$, and thus (using $\log(n!) \leq n\log(n) - n$), we have proven the Theorem 8.

Appendix E. Algorithms

This section contains additional formulas that can be used for moments computation respectively in case of sequence labeling (Algorithm 4) and of sequence alignment (Algorithm 5).

Algorithm 4 Extra formulas for sequence labeling

11: **if** $z = i$ **then** $M := 1$
12: $\mu^t_{pz}(i,j) := \frac{\sum_i \mu^t_{pz}(i,j-1)\pi(i,j-1) + M\pi(p,j-1)}{\pi(i,j)}$

5: **if** $q = x_1 \wedge p = i$ **then** $v^e_{pq}(i,1) := 1$
11: **if** $q = x_j \wedge p = i$ **then** $M := 1$
12: $v^e_{pq}(i,j) := \frac{\sum_i (v^e_{pq}(i,j-1) + 2M\mu^e_{pq}(i,j-1) + M)\pi(i,j-1)}{\pi(i,j)}$

11: **if** $q' = x_j \wedge p' = i$ **then** $M_1 := 1$
if $q = x_j \wedge p = i$ **then** $M_2 := 1$
12: $v^e_{pqp'q'}(i,j) := \frac{\sum_i (v^e_{pqp'q'}(i,j-1) + M_1\mu^e_{pq}(i,j-1) + M_2\mu^e_{p'q'}(i,j-1))\pi(i,j-1)}{\pi(i,j)}$

5: **if** $p = i$ **then** $v^t_{pz}(i,2) = 1$
11: **if** $p = i$ **then** $M := 1$
12: $v^t_{pz}(i,j) := \frac{\sum_i (v^t_{pz}(i,j-1)\pi(i,j-1)) + (2M\mu^t_{pz}(p,j-1) + M)\pi(p,j-1)}{\pi(i,j)}$

11: **if** $p' = j$ **then** $M_1 := 1$
if $p = j$ **then** $M_2 := 1$
12: $v^t_{pzp'z'}(i,j) := \frac{(\sum_i v^t_{pzp'z'}(i,j-1)\pi(i,j-1) + M_1\mu^t_{pz}(p',j-1)\pi(p',j-1) + M_2\mu^t_{p'z'}(p,j-1)\pi(p,j-1))}{\pi(i,j)}$

11: **if** $z' = i$ **then** $M_1 := 1$
if $q = x_j \wedge p = i$ **then** $M_2 := 1$
12: $v^{et}_{pqp'z}(i,j) := \frac{(\sum_i v^{et}_{pqp'z}(i,j-1)\pi(i,j-1) + M_1\mu^e_{pq}(p',j-1)\pi(p',j-1) + M_2\mu^t_{p'z'}(p,j)\pi(p,j))}{\pi(i,j)}$

References

Y. Altun, T. Hofmann, and M. Johnson. Discriminative learning for label sequences via boosting. In *Advances in Neural Information Processing Systems (NIPS)*, pages 977-984, Vancouver, British Columbia, 2003.

Algorithm 5 Extra formulas for affine gap penalties

$$\mu_e(i,j) := \tfrac{1}{\pi(i,j)}(\mu_e(i-1,j)\pi(i-1,j)+\pi(i-2,j)+\mu_e(i,j-1)\pi(i,j-1)$$
$$+\pi(i,j-2)+\mu_e(i-1,j-1)\pi(i-1,j-1))$$

$$\mu_o(i,j) := \tfrac{1}{\pi(i,j)}(\mu_o(i-1,j)\pi(i-1,j)+\pi(i-1,j)-\pi(i-2,j)+$$
$$\mu_o(i,j-1)\pi(i,j-1)+\pi(i,j-1)-\pi(i,j-2)+\mu_o(i-1,j-1)\pi(i-1,j-1))$$

$$\nu_{oo}(i,j) := \tfrac{1}{\pi(i,j)}(\nu_{oo}(i-1,j-1)\pi(i-1,j-1)+\nu_{oo}(i-1,j)\pi(i-1,j)$$
$$+2(\mu_o(i-1,j)\pi(i-1,j)-\mu_o(i-2,j)\pi(i-2,j)-\pi(i-2,j)+\pi(i-3,j))$$
$$+\pi(i-1,j)-\pi(i-2,j)+\nu_{oo}(i,j-1)\pi(i,j-1)+2(\mu_o(i,j-1)\pi(i,j-1)$$
$$-\mu_o(i,j-2)\pi(i,j-2)-\pi(i,j-2)+\pi(i,j-3))+\pi(i,j-1)-\pi(i,j-2)$$

$$\nu_{ee}(i,j) := \tfrac{1}{\pi(i,j)}(\nu_{ee}(i-1,j-1)\pi(i-1,j-1)+\nu_{ee}(i-1,j)\pi(i-1,j)$$
$$+2\mu_e(i-2,j)\pi(i-2,j)+2\pi(i-3,j)+\pi(i-2,j)+\nu_{ee}(i,j-1)\pi(i,j-1)$$
$$+2\mu_e(i,j-2)\pi(i,j-2)+2\pi(i,j-3)+\pi(i,j-2))$$

$$\nu_{mo}(i,j) := \tfrac{1}{\pi(i,j)}((\nu_{mo}(i-1,j)+\mu_m(i-1,j))\pi(i-1,j)-\mu_m(i-2,j)\pi(i-2,j)$$
$$+(\nu_{mo}(i,j-1)+\mu_m(i,j-1))\pi(i,j-1)-\mu_m(i,j-2)\pi(i,j-2)$$
$$+(\nu_{mo}(i-1,j-1)+M\mu_o(i-1,j-1))\pi(i-1,j-1))$$

$$\nu_{me}(i,j) := \tfrac{1}{\pi(i,j)}((\nu_{me}(i-1,j-1)+M\mu_e(i-1,j-1))\pi(i-1,j-1)$$
$$+\nu_{me}(i-1,j)\pi(i-1,j)+\mu_m(i-2,j)\pi(i-2,j)+\nu_{me}(i,j-1)\pi(i,j-1)$$
$$+\mu_e(i,j-2)\pi(i,j-2))$$

$$\nu_{eo}(i,j) := \tfrac{1}{\pi(i,j)}(\nu_{eo}(i-1,j-1)\pi(i-1,j-1)+\nu_{eo}(i,j-1)\pi(i,j-1)$$
$$+\mu_o(i,j-2)\pi(i,j-2)+\pi(i,j-2)-2\pi(i,j-3)+\mu_e(i,j-1)\pi(i,j-1)$$
$$-\mu_e(i,j-2)\pi(i,j-2)+\nu_{eo}(i-1,j)\pi(i-1,j)+\mu_o(i-2,j)\pi(i-2,j)$$
$$+\pi(i-2,j)-2\pi(i-3,j)+\mu_e(i-1,j)\pi(i-1,j)-\mu_e(i-2,j)\pi(i-2,j))$$

Y. Altun, T. Hofmann, and A. J. Smola. Gaussian process classification for segmenting and annotating sequences. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML)*, Banff, Alberta, Canada, 2004.

Y. Altun, I. Tsochantaridis, T. Hofmann. Hidden Markov support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, pages 3-10, Washington, DC, USA, 2003.

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463-482, 2002.

H.S. Booth, J.H. Maindonald, S.R. Wilson, and J.E. Gready. An efficient Z-score algorithm for assessing sequence alignments. *Journal of Computational Biology*, 11(4):616-25, 2004.

M. Collins. 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)*, pages 1-8, 2002.

M. Collins. Ranking algorithms for named-entity extraction: boosting and the voted perceptron. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 489-496, 2002.

C. B. Do, D. A. Woods, and S. Batzoglou. CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics*, 22:e90-8, 2006.

R.F. Doolittle. Similar amino acid sequences: chance or common ancestry. *Science*, 214:149-159, 1981.

R. D. Dowell and S. R. Eddy. Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction. *BMC Bioinformatics*, 5(71):1-14, 2004.

S. Elizalde and K. Woods. Bounds on the number of inference functions of a graphical model. *Statistica Sinica*, 17:1395-1415, 2007.

Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, pages 170-178, Madison, Wisconson, USA, 1998.

S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S.R. Eddy. Rfam: an RNA family database. *Nucleic Acids Research*, 31(1):439-441, 2003.

D. Gusfield, K. Balasubramanian, and D. Naor. Parametric optimization of sequence alignment. *Algorithmica*, 12:312-326, 1994.

D. Gusfield and P. Stelling. Parametric and inverse-parametric sequence alignment with XPARAL. *Methods in Enzymology*, 266:481-494, 1996.

T. Joachims, T. Galor, and R. Elber, Learning to align sequences: a maximum-margin approach, In *New Algorithms for Macromolecular Simulation*, B. Leimkuhler, LNCS Vol. 49, Springer, 2005.

J. Kececioglu and E. Kim, Simple and fast inverse alignment, In *Proceedings of the Tenth ACM Conference on Research in Computational Molecular Biology (RECOMB)*, pages 441-455, Venice, Italy, 2006.

T. Kudo. CRF++: Yet another CRF toolkit, 2005. *[http://crfpp.sourceforge.net]*.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: probabilistic models for segmenting and labeling data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, pages 282-289, Williamstown, MA, USA, 2001.

J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: representation and clique selection. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML)*, pages 64-71, Banff, Alberta, Canada, 2004.

C. D. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, MA, 1999.

D. McAllester. Generalization bounds and consistency for structured labeling in predicting structured data. *Predicting Structured Data*, edited by G. Bakir, T. Hofmann, B. Scholkopf, A. Smola, B. Taskar, and S. V. N. Vishwanathan, MIT Press, 2007.

A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 591-598, Stanford, CA, USA, 2000.

C. McDiarmid. On the method of bounded differences, *London Mathematical Society Lecture Note Series*, 141:148-188, 1989.

S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443-453, 1970.

L. Pachter and B. Sturmfels. Parametric inference for biological sequence analysis. In *Proceedings of the National Academy of Sciences USA*, 101(46):16138-16143, 2004.

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257-286, 1989.

E. Ricci, T. De Bie, and N. Cristianini. Learning to align: a statistical approach. In *Proceedings of the Seventh International Symposium on Intelligent Data Analysis (IDA)*, pages 25-36, Ljubljana, Slovenia, 2007.

K. Sato and Y. Sakakibara. RNA secondary structural alignment with conditional random fields. *Bioinformatics*, 21(Suppl 2):ii237-ii242, 2005.

R. Schapire and Y. Singer. Improved boosting algorithms using confidencerated predictions. *Machine Learning*, 37(3):297-336, 1999.

F. Sun, D. Fernandez-Baca, and W. Yu. Inverse parametric sequence alignment. *Journal of Algorithms*, 53(1):36-54, 2004.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In In *Advances in Neural Information Processing Systems (NIPS)*, Vancouver and Whistler, British Columbia, Canada, 2003.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. Max-margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1-8, Barcelona, Spain, 2004.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables, *Journal of Machine Learning Research*, 6(9):1453-1484, 2005.

V. N. Vapnik. *Statistical Learning Theory*. Wiley & Sons, Inc., 1998.

D. H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 2(10):189-208, 1967.