Mach Learn (2009) 76: 227–242
DOI 10.1007/s10994-009-5129-3

On structured output training: hard cases and an efficient alternative

Thomas Gärtner · Shankar Vembu

Received: 12 June 2009 / Revised: 12 June 2009 / Accepted: 16 June 2009 / Published online: 23 July 2009

Abstract We consider a class of structured prediction problems for which the assumptions made by state-of-the-art algorithms fail. To deal with exponentially sized output sets, these algorithms assume, for instance, that the best output for a given input can be found efficiently. While this holds for many important real world problems, there are also many relevant and seemingly simple problems where these assumptions do not hold. In this paper, we consider route prediction, which is the problem of finding a cyclic permutation of some points of interest, as an example and show that state-of-the-art approaches cannot guarantee polynomial runtime for this output set. We then present a novel formulation of the learning problem that can be trained efficiently whenever a particular 'super-structure counting' problem can be solved efficiently for the output set. We also list several output sets for which this assumption holds and report experimental results.

Keywords Structured prediction · Combinatorial structures · Kernel methods

1 Introduction

State-of-the-art structured prediction algorithms can be applied using off-the-shelf tools by implementing a joint kernel for inputs and outputs, and an algorithm for inference. The kernel is used for mapping the data to an appropriate feature space, while the inference algorithm is used for successively adding violated constraints to the optimisation problem. While this approach leads to efficient learning algorithms for many important real world problems, there are also many cases in which successively adding violated constraints is infeasible. Our main results are: (i) We show that the usual assumptions do not hold for many

Editors: Aleksander Kołcz, Dunja Mladenić, Wray Buntine, Marko Grobelnik, and John Shawe-Taylor.

T. Gärtner · S. Vembu (✉)
Fraunhofer IAIS, Schloß Birlinghoven, 53754 Sankt Augustin, Germany
e-mail: shankar.vembu@iais.fraunhofer.de

T. Gärtner
e-mail: thomas.gaertner@iais.fraunhofer.de

output sets, even if we restrict the class of structures to those for which a particular counting problem can be solved efficiently; (ii) We devise an alternative optimisation problem that can be solved efficiently whenever this counting problem can be.

As a simple yet relevant problem, we consider the prediction of routes over given points of interest. Solving this problem has many potential applications. For car drivers, prediction of individual routes can be used for intelligent car sharing applications or help optimise a hybrid vehicle's charge/discharge schedule (hybrid fuel economy can be improved by up to 8% if the route is known in advance or predicted accurately—see Froehlich and Krumm 2008 and references therein). Such applications are gaining more importance as more hybrid cars are being used and as the traffic density is increasing nowadays.

Structure of this paper After fixing notation and recalling the basic structured prediction setting, we review the usual training approach (Sect. 2). We then show that the usual assumptions do not hold for route prediction and also for a large class of other problems (Sect. 3). After that, we show how 'super-structures' can be counted efficiently for the case of route prediction and derive an efficient structured output training algorithm under this assumption (Sect. 4). To make this alternative approach applicable to a wider class of problems, we then introduce super-structure counting for a wide class of output sets (Sect. 5). After discussing related work (Sect. 6), we empirically compare our algorithm to special purpose algorithms on multi-label classification and hierarchical classification, and to a general structured output training algorithm on route prediction (Sect. 7).

2 Preliminaries

2.1 Notation

We use $[\![n, m]\!]$ to denote the set of integers $\{n, n+1, \ldots, m\}$, $[\![n]\!]$ to denote the set $[\![1, n]\!]$, and $\langle \cdot, \cdot \rangle$ to denote inner products. Let $\mathcal{X}$ and $\mathcal{Y}$ be the input and output set, respectively, and let $\{(x_i, y_i)\}_{i \in [\![m]\!]} \subseteq \mathcal{X} \times \mathcal{Y}$ be a set of observations. In general, $\mathcal{X}$ is an arbitrary set for which we only assume the presence of an appropriate positive definite kernel function $k_{\mathcal{X}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. The set $\mathcal{Y}$ is parameterised by a finite alphabet Σ. We assume that a polynomial time computable map $\phi : \mathcal{Y} \to \mathbb{R}^d$ is defined and will refer to the inner product under this map by $k_{\mathcal{Y}} : y, y' \mapsto \langle \phi(y), \phi(y') \rangle$. For the joint kernel of inputs and outputs, we consider the tensor product kernel $k[(x, y), (x', y')] = k_{\mathcal{X}}(x, x') k_{\mathcal{Y}}(y, y')$, which has found application in many important domains (Jacob and Vert 2008; Erhan et al. 2006). We refer to the Hilbert spaces corresponding to $k_{\mathcal{X}}, k_{\mathcal{Y}}, k$ as $\mathcal{H}_{\mathcal{X}}, \mathcal{H}_{\mathcal{Y}}, \mathcal{H}$, respectively. In our route prediction problem, each x_i represents a set of features such as an individual person, a day, and a time of the day; Σ^{cyc} is a set containing points of interest; $\mathcal{Y}^{\text{cyc}}$ is the set of cyclic permutations of subsets of Σ containing at least three points (we are interested in cyclic permutations as we assume that each individual starts and ends each route in the same place, e.g., his/her home); $k_{\mathcal{X}}^{\text{cyc}}$ is the linear kernel; and $d^{\text{cyc}} = \Sigma \times \Sigma$. We represent a cyclic permutation by the set of (predecessor, successor)-pairs. For instance, the sequences $\{abc, bca, cab\}$ are equivalent and we use $\{(a, b), (b, c), (c, a)\}$ to represent them. Furthermore, we define $\phi^{\text{cyc}}_{(u,v)}(y) = 1$ if $(u, v) \in y$, i.e., v follows directly after u in y; $\phi^{\text{cyc}}_{(u,v)}(y) = -1$ if $(v, u) \in y$, i.e., u follows directly after v in y; and 0 otherwise.

Throughout this paper, we call an algorithm efficient if it has runtime polynomial in $|\Sigma|$ and the size of the observations.

2.2 Regularised risk minimisation for structured output training

Assuming for simplicity uniform misclassification costs, regularised risk minimisation based approaches aim at minimising an upper bound on the expected error

$$\operatorname*{argmin}_{h\in\mathcal{H}} \lambda\Omega[h] + \sum_{i\in[\![m]\!]} |\{z \in \mathcal{Y} \mid h(x_i, z) > h(x_i, y_i)\}|,$$

where $\lambda \geq 0$ is the regularisation parameter and $\Omega : \mathcal{H} \to \mathbb{R}$ is a convex regularisation function such as $\|\cdot\|^2$.

As the above problem is non-convex, state-of-the-art large margin approaches (Tsochantaridis et al. 2005; Taskar et al. 2005) minimise a convex upper bound; for instance, using hinge loss we obtain $Q(\{(x_i, y_i) \mid i \in [\![m]\!]\}) =$

$$\begin{aligned} &\operatorname*{argmin}_{h\in\mathcal{H}} \lambda\Omega[h] + \sum_{i\in[\![m]\!]} \sum_{z\in\mathcal{Y}\setminus\{y_i\}} \xi_{iz} \\ &\text{subject to} \quad h(x_i, y_i) - h(x_i, z) \geq 1 - \xi_{iz} \quad (\forall i \in [\![m]\!], z \in \mathcal{Y}) \\ &\qquad\qquad \xi_{iz} \geq 0 \quad (\forall i \in [\![m]\!], z \in \mathcal{Y})\,. \end{aligned} \tag{1}$$

In the following, we are also interested in the set of hypotheses potentially occurring as solutions to this optimisation problem and denote it as $\mathcal{H}_{\text{opt}} = \{Q(D) \mid D \subseteq \mathcal{X} \times \mathcal{Y}\}$.

3 Hardness results

In this section, we discuss the assumptions under which state-of-the-art structured output learning algorithms can be applied efficiently and show that these assumption do not hold for several relevant output sets. This is for instance the case for route prediction.

3.1 Motivation

The key challenge in solving the convex optimisation problem (1) is that the number of constraints grows proportional to $|\mathcal{Y}|$ and thus usually exponentially with $|\Sigma|$. To ensure polynomial time complexity, different assumptions need to be made and depending on the nature of Ω, different methods are used that iteratively optimise and add violated constraints. With linear Ω the ellipsoid algorithm can be used if an efficient separation oracle exists (Korte and Vygen 2008), otherwise cutting-plane methods can be employed under similar assumptions. The strongest assumption is the existence of an efficient algorithm for finding the 'best' structure given a hypothesis and an input (Tsochantaridis et al. 2005; Collins 2002). A weaker assumption, which is in fact sufficient for the proof of the main result in Collins (2002), is the existence of an efficient algorithm for finding a 'better' structure than the given one, i.e., an efficient separation oracle. An even weaker assumption is that there is a short certificate for the 'best' structure (Taskar et al. 2005).

In the following, we will first show that these assumptions do not hold if the sub-optimality decision problem for a given $(\mathcal{Y}, \mathcal{H}_{\text{opt}})$ is NP-hard. This decision problem is defined as deciding if for a given $h \in \mathcal{H}_{\text{opt}}$, $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, a $z \in \mathcal{Y}$ exists such that $h(x, z) > h(x, y)$. We then show that for the specific case of route prediction, sub-optimality is indeed NP-complete. Our hardness result gains further significance as we can also show that this case can indeed occur for a specific set of observations. We then turn to a class of

problems for which the output forms particular set systems, show that in this case the above assumptions are polynomially equivalent, and note that they are often hard by relating them to edge-deletion problems.

3.2 Representation in output space

State-of-the-art structured output training algorithms assume (at least) that deciding if an output structure with higher score than a given one exists is in NP. With the definitions given above, we formally define $(\mathcal{Y}, \mathcal{H}_{\text{opt}})$-OPTIMALITY as:

$$h \in \mathcal{H}_{\text{opt}}, x \in \mathcal{X}, y \in \mathcal{Y} \mapsto (\nexists z \in \mathcal{Y} : h(x, z) > h(x, y)) .$$

The strong representer theorem (Schölkopf et al. 2001) holds for all minimisers $h^* \in \mathcal{H}_{\text{opt}}$ of the optimisation problem (1). It shows that $h^* \in \mathbf{span}\{k[(x_i, z), (\cdot, \cdot)] \mid i \in [\![m]\!], z \in \mathcal{Y}\}$. That is,

$$\begin{aligned} h^*(x, y) &= \sum_{i \in [\![m]\!], z \in \mathcal{Y}} \boldsymbol{\alpha}_{iz} k[(x_i, z), (x, y)] = \sum_{i \in [\![m]\!], z \in \mathcal{Y}} \boldsymbol{\alpha}_{iz} k_{\mathcal{X}}(x_i, x) \langle \phi(z), \phi(y) \rangle \\ &= \left\langle \sum_{i \in [\![m]\!], z \in \mathcal{Y}} \boldsymbol{\alpha}_{iz} k_{\mathcal{X}}(x_i, x) \phi(z), \phi(y) \right\rangle . \end{aligned}$$

This motivates us to first consider $(\mathcal{Y}, \phi)$-OPTIMALITY, defined as:

$$w \in \mathbf{span}\{\phi(y) \mid y \in \mathcal{Y}\}, \quad y \in \mathcal{Y} \mapsto (\nexists z \in \mathcal{Y} : \langle w, \phi(z) \rangle > \langle w, \phi(y) \rangle) .$$

To show that $(\mathcal{Y}, \phi)$-OPTIMALITY is not in NP, it suffices to show that $(\mathcal{Y}, \phi)$-SUB-OPTIMALITY is NP-hard. Formally, we define $(\mathcal{Y}, \phi)$-SUB-OPTIMALITY as:

$$w \in \mathbf{span}\{\phi(y) \mid y \in \mathcal{Y}\}, \quad y \in \mathcal{Y} \mapsto (\exists z \in \mathcal{Y} : \langle w, \phi(z) \rangle > \langle w, \phi(y) \rangle) ,$$

and correspondingly $(\mathcal{Y}, \mathcal{H}_{\text{opt}})$-SUB-OPTIMALITY.

3.3 Route prediction—hardness of finding cyclic permutations

We now consider the route prediction problem as an instance for which $(\mathcal{Y}, \mathcal{H}_{\text{opt}})$-SUB-OPTIMALITY is NP-complete and for which thus the assumptions made by state-of-the-art structured prediction algorithms do not hold.

Lemma 1 *With all constants and functions defined as above,* $(\mathcal{Y}^{\text{cyc}}, \phi^{\text{cyc}})$-SUB-OPTIMALITY *is NP-complete.*

Proof We suppose a polynomial time algorithm for $(\mathcal{Y}^{\text{cyc}}, \phi^{\text{cyc}})$-SUB-OPTIMALITY exists and show how it could be used to solve the Hamiltonian path problem for an arbitrary graph $G = (V, E)$. Without loss of generality, we assume that $V \cap [\![2|V| - 1]\!] = \emptyset$. Let $\Sigma = V \cup (V \times V) \cup [\![2|V| - 1]\!]$ and let $y^{\times} = \{(1, 2), (2, 3), \ldots, (2|V| - 2, 2|V| - 1), (2|V| - 1, 1)\}$. With

$$w^{\times} = \phi^{\text{cyc}}(y^{\times}) + \sum_{\{u,v\} \in E} \phi^{\text{cyc}}(\{(u, (u, v)), ((u, v), v), (v, (v, u)), ((v, u), u)\}) ,$$

we have $\langle w^\times, \phi^{\text{cyc}}(y^\times)\rangle = \|\phi^{\text{cyc}}(y^\times)\|^2 = 4|V| - 2$ and

$$\begin{aligned} G \text{ has a Hamiltonian cycle} \quad &\Leftrightarrow \quad \left(\exists z \in \mathcal{Y}^{\text{cyc}} : \langle w^\times, \phi^{\text{cyc}}(z)\rangle > \langle w^\times, \phi^{\text{cyc}}(y^\times)\rangle\right) \\ &\Leftrightarrow \quad (\mathcal{Y}^{\text{cyc}}, \phi^{\text{cyc}})\text{-SUB-OPTIMALITY}(w^\times, y^\times). \end{aligned}$$

Observing that by definition $w^\times \in \mathbf{span}\{\phi^{\text{cyc}}(z) \mid z \in \mathcal{Y}^{\text{cyc}}\}$ completes this part of the proof. It now remains to observe that any $z \in \mathcal{Y}^{\text{cyc}}$ with $\langle w^\times, \phi^{\text{cyc}}(z)\rangle > \langle w^\times, \phi^{\text{cyc}}(y^\times)\rangle$ can serve as a short certificate of sub-optimality. □

Theorem 1 *With all constants and functions defined as above,* $(\mathcal{Y}^{\text{cyc}}, \mathcal{H}^{\text{cyc}}_{\text{opt}})$-SUB-OPTIMALITY *is NP-complete.*

Proof For an arbitrary graph $G = (V, E)$, let Σ, $y^\times$, $w^\times$ be defined as above and let $\mathcal{X} = 2^E$ with $k_{\mathcal{X}}(e, e') = |e \cap e'|$. With $m = |E| + 1$, $\{x_i \mid i \in [\![|E|]\!]\} = E$ and $x_m = E$, we choose the training data D as

$$\{(\{\{u, v\}\}, \{(u, (u, v)), ((u, v), v), (v, (v, u)), ((v, u), u)\}) \mid \{u, v\} \in E\} \cup \{(E, y^\times)\},$$

and let $\boldsymbol{\alpha}_{iz} = 1/2$ for $z = y_i$ and $\boldsymbol{\alpha}_{iz} = 0$ otherwise. That is, each edge in the original, undirected graph is represented by a directed cycle of length four. For $i \in [\![|E|]\!]$ we then have

$$h(x_i, y_i) = \left\langle \sum_{i \in [\![m]\!], z \in \mathcal{Y}} \boldsymbol{\alpha}_{iz} k_{\mathcal{X}}(x_i, x)\phi(z), \phi(y_i) \right\rangle = 4,$$

$h(x_i, z) = 3$ if $|z \cap y_i| = 3$, and $h(x_i, z) < 3$ if $|z \cap y_i| < 3$. Thus there are no violated constraints for $i \in [\![|E|]\!]$ and h is indeed optimal on this part of the data. Furthermore,

$$h(x_m, y_m) = \left\langle \sum_{i \in [\![m]\!], z \in \mathcal{Y}} \boldsymbol{\alpha}_{iz} k_{\mathcal{X}}(x_i, x)\phi(z), \phi(y_m) \right\rangle = \langle w^\times, \phi(y^\times)\rangle.$$

Together with Lemma 1 this gives the desired result. □

3.4 Connections to other assumptions

The strongest assumption made in related work is the existence of a polynomial time algorithm for $(\mathcal{Y}, \mathcal{H})$-DECODING (Tsochantaridis et al. 2005; Collins 2002):

$$h \in \mathcal{H}, x \in \mathcal{X} \mapsto \operatorname*{argmax}_{z \in \mathcal{Y}} h(x, z).$$

A weaker assumption which is in fact sufficient for the proof in Collins (2002) is the existence of a polynomial time algorithm for $(\mathcal{Y}, \mathcal{H})$-SEPARATION:

$$h \in \mathcal{H}, x \in \mathcal{X}, y \in \mathcal{Y} \mapsto \begin{cases} z & \text{for some } z \in \mathcal{Y} \text{ with } h(x, z) > h(x, y) \\ \emptyset & \text{otherwise.} \end{cases}$$

An even weaker assumption is that optimality is in NP (Taskar et al. 2005). Formally $(\mathcal{Y}, \mathcal{H})$-OPTIMALITY is

$$h \in \mathcal{H}, x \in \mathcal{X}, y \in \mathcal{Y} \mapsto \nexists z \in \mathcal{Y} : \quad h(x, z) > h(x, y).$$

Proposition 1
$(\mathcal{Y}, \mathcal{H})$-SUB-OPTIMALITY *is NP-complete.*
$\Rightarrow$ $(\mathcal{Y}, \mathcal{H})$-OPTIMALITY *is coNP-complete.*
$\Rightarrow$ $(\mathcal{Y}, \mathcal{H})$-SEPARATION *is coNP-hard.*
$\Rightarrow$ $(\mathcal{Y}, \mathcal{H})$-DECODING *is coNP-hard.*

Proof The result follows immediately by observing that (1) optimality is the complement of sub-optimality, (2) any separation oracle can be used to decide optimality, and (3) any algorithm for decoding can be used as a separation oracle. □

Corollary 1 *Unless* coNP = NP, $(\mathcal{Y}^{\text{cyc}}, \mathcal{H}^{\text{cyc}})$-OPTIMALITY *is not contained in* NP. *Unless* P = coNP, *there is no polynomial time algorithm for*

1. $(\mathcal{Y}^{\text{cyc}}, \mathcal{H}^{\text{cyc}})$-SEPARATION, *and*
2. $(\mathcal{Y}^{\text{cyc}}, \mathcal{H}^{\text{cyc}})$-DECODING.

Proof The result follows immediately from Theorem 1 and Proposition 1. □

Lastly, the decision problem $(\mathcal{Y}, \mathcal{H})$-OPTIMAL-VALUE

$$\beta \in \mathbb{R}, h \in \mathcal{H}, x \in \mathcal{X}, y \in \mathcal{Y} \mapsto (\beta = \max\{h(x, z) \mid z \in \mathcal{Y}\})$$

is also of interest. Following the proof that EXACT-TSP is D^P-complete (Papadimitriou 1994), it can be shown that also $(\mathcal{Y}^{\text{cyc}}, \mathcal{H}^{\text{cyc}})$-OPTIMAL-VALUE is D^P-complete. The significance of this result is that optimality-value can be reduced to separation and decoding, providing even stronger evidence that neither of these problems can be solved in polynomial time.

3.5 Set systems

We now consider set systems $(\Sigma, \mathcal{Y}^\pi)$ with $\mathcal{Y}^\pi = \{y \in 2^\Sigma \mid \pi(y) = 1\}$ where $\pi : 2^\Sigma \to \{\pm 1\}$, and let $\phi^\in : \mathcal{Y} \to \{0, 1\}^\Sigma$ be the indicator function $\phi^\in_e(y) = 1$ if $e \in y$ and 0 otherwise. An independence system is a set system for which $y' \subseteq y \in \mathcal{Y}^\pi$ implies $y' \in \mathcal{Y}^\pi$, i.e., for which π is monotone. In this section, we often need to make the dependence of $\mathcal{Y}$ on the alphabet Σ explicit and therefore denote the parameterised output set as $\mathcal{Y}(\Sigma)$.

We call a set system $(\Sigma, \mathcal{Y}^\pi(\Sigma))$ addable if for each $\ell \in \mathbb{N}$ there is a $\Sigma_\ell \supset \Sigma$, such that $y_\ell = \text{argmax}\{|y| \mid y \in \mathcal{Y}^\pi(\Sigma_\ell \setminus \Sigma)\}$ can be found efficiently and $|y_\ell| = \ell = \max\{|y| \mid y \in \mathcal{Y}^\pi(\Sigma_\ell) \setminus \mathcal{Y}^\pi(\Sigma)\}$. For instance, all independence systems are addable, whereas anti-monotone set systems are not.

Proposition 2 *For addable set systems, if we restrict our setting such that* $w \in \{0, 1\}^d$, *then* $(\mathcal{Y}^\pi, \phi^\in)$-DECODING, $(\mathcal{Y}^\pi, \phi^\in)$-SEPARATION, *and* $(\mathcal{Y}^\pi, \phi^\in)$-OPTIMALITY *are polynomially equivalent.*

Proof It remains to show that a polynomial time algorithm for deciding optimality could be used to solve the decoding problem in polynomial time. First note that $\langle w, \phi(y)\rangle$ can assume only d different values.

Beginning with the empty set as the current estimate, if the current estimate is optimal, then we are done. Otherwise, we remove in succession each element of the alphabet that is not contained in the current estimate. If removal of an element makes the current estimate

optimal, we add it back in and mark it as an element of the next estimate. Once all edges have been processed, we have a new estimate Σ_ℓ of size ℓ equal to the number of marked elements. We add Σ_ℓ to the original set system, and repeat this procedure with the new set system and y_ℓ as the new estimate. □

A particularly interesting case is hereditary properties on the edges of a graph $G = (V, E)$ where we let $\Sigma = E \subseteq \{U \subseteq V \mid |U| = 2\}$. A property is called hereditary if $\forall F, F' \subseteq E$ we have $F \subseteq F'$ implies $\pi(F) \geq \pi(F')$, i.e., hereditary properties correspond to independence systems. In this case, sub-optimality corresponds to minimum edge deletion problems, which are NP-hard for several important hereditary graph properties like planar, outerplanar, line-invertible, comparability graph, bipartite graph, and many more (Yannakakis 1978). In this case again optimality is not in NP (unless coNP = NP), and a polynomial time algorithm cannot be expected for either separation or decoding.

4 Counting-based structured output training algorithm

We have thus far shown that state-of-the-art structured output learning algorithms cannot be applied efficiently to predict several relevant output structures. We now propose an optimisation problem that admits efficient algorithms under an orthogonal assumption. This assumption holds, for instance, for the case of route prediction.

4.1 Motivation

The major difficulty in structured output learning is to handle the exponentially many constraints that occur in all state-of-the-art algorithms (1). While successively adding violated constraints is feasible under several assumptions, in the previous section we discussed cases like route prediction where none of these assumptions hold.

In the following, we will first show, considering again cyclic permutations as an example, that counting the number of super-structures can be feasible even if there are exponentially many and even if the assumptions made by state-of-the-art structured output learning algorithms do not hold. In particular, it is possible to efficiently compute

$$|\mathcal{Y}^{\text{cyc}}|, \quad \Phi^{\text{cyc}} = \sum_{y \in \mathcal{Y}^{\text{cyc}}} \phi^{\text{cyc}}(y), \quad \text{and} \quad C^{\text{cyc}} = \sum_{y \in \mathcal{Y}^{\text{cyc}}} \phi^{\text{cyc}}(y) \left[\phi^{\text{cyc}}\right]^\top (y).$$

Following this insight, we aim at a structured output learning algorithm that is efficient whenever super-structure counting is feasible. Indeed, whenever we are minimising a quadratic upper bound on the loss, we can derive an unconstrained optimisation problem accessing the output set only by means of $|\mathcal{Y}^{\text{cyc}}|$, Φ^{cyc} and C^{cyc}. To solve this optimisation problem, we give the gradient as well as the Hessian-vector multiplication.

The remainder of this paper is concerned with output sets for which $|\mathcal{Y}|$, $\sum_{y \in \mathcal{Y}} \phi(y)$ and $\sum_{y \in \mathcal{Y}} \phi(y)\phi^\top(y)$ can be computed efficiently.

4.2 Route prediction—counting cyclic permutations

For a given alphabet Σ, we are now interested in computing $|\mathcal{Y}^{\text{cyc}}|$, the number of cyclic permutations of subsets of Σ. For a subset of size i there are $i!$ permutations of which i

represent the same cyclic permutation. That is, there are $(i-1)!$ cyclic permutations of each subset of size i, and for an alphabet of size $N = |\Sigma|$ there are

$$|\mathcal{Y}^{\text{cyc}}| = \sum_{i=3}^{N} \binom{N}{i} (i-1)!$$

different cyclic permutations of subsets of Σ.

Computing Φ^{cyc} is even simpler. Observe that, for each cyclic permutation containing a (predecessor, successor)-pair (u, v), there is also exactly one cyclic permutation containing (v, u). Hence the sum over each feature is zero and $\Phi^{\text{cyc}} = \mathbf{0}$ (where $\mathbf{0}$ is the vector of all zeros).

It remains to compute C^{cyc}. Each element of this matrix is computed as

$$C^{\text{cyc}}_{(u,v),(u',v')} = \sum_{y \in \mathcal{Y}^{\text{cyc}}} \phi^{\text{cyc}}_{(u,v)}(y) \phi^{\text{cyc}}_{(u',v')}(y).$$

As seen above, for $u = u'$ and $v = v'$ there are as many cyclic permutations for which $\phi^{\text{cyc}}_{(u,v)} = +1$ as there are cyclic permutations for which $\phi^{\text{cyc}}_{(u,v)} = -1$. In both cases, $\phi^{\text{cyc}}_{(u,v)}(y)\phi^{\text{cyc}}_{(u',v')}(y) = +1$, and to compute $C^{\text{cyc}}_{(u,v),(u,v)}$ it suffices to compute the number of cyclic permutations containing (u, v) or (v, u). There are $(i-2)!$ different cyclic permutations of each subset of size i that contain (u, v). We thus have

$$C^{\text{cyc}}_{(u,v),(u,v)} = 2 \sum_{i=3}^{N} \binom{N-2}{i-2} (i-2)!.$$

Similarly, for $u = v'$ and $v = u'$, we have $\phi^{\text{cyc}}_{(u,v)}(y)\phi^{\text{cyc}}_{(u',v')}(y) = -1$ and

$$C^{\text{cyc}}_{(u,v),(v,u)} = -2 \sum_{i=3}^{N} \binom{N-2}{i-2} (i-2)!.$$

For $u \neq u' \neq v \neq v' \neq u$, we observe that there are as many cyclic permutations containing (u, v) and (u', v') as there are cyclic permutations containing (u, v) and (v', u'), and hence in this case $C^{\text{cyc}}_{(u,v),(u',v')} = 0$. Finally, we need to consider $C^{\text{cyc}}_{(u,v),(u,v')}$, $C^{\text{cyc}}_{(u,v),(u',v)}$, $C^{\text{cyc}}_{(u,v),(v,v')}$, and $C^{\text{cyc}}_{(u,v),(u',u)}$. Here we have

$$C^{\text{cyc}}_{(u,v),(u,v')} = C^{\text{cyc}}_{(u,v),(u',v)} = -2 \sum_{i=3}^{N} \binom{N-3}{i-3} (i-3)! \quad \text{and}$$

$$C^{\text{cyc}}_{(u,v),(v,v')} = C^{\text{cyc}}_{(u,v),(u',u)} = +2 \sum_{i=3}^{N} \binom{N-3}{i-3} (i-3)!$$

We are now ready to investigate structured output training algorithms based on these counting formulae.

4.3 Loss function

We consider structured output *ranking*, i.e., given a set of training examples $(x_1, Y_1), \ldots, (x_m, Y_m) \in \mathcal{X} \times 2^{\mathcal{Y}}$, the goal is to learn a scoring function $h : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that, for each

$x_i \in \mathcal{X}$, orders (ranks) $\mathcal{Y}$ in such a way that it assigns a higher score to all $y \in Y_i$ than to all $z \in \mathcal{Y} \setminus Y_i$. We use the following quadratic upper bound on the ranking loss $\mathcal{R}(h, i) =:$

$$\sum_{\substack{y \in Y_i \\ z \in \mathcal{Y} \setminus Y_i}} \left[h(x_i, z) - h(x_i, y) + \frac{1}{2} h^2(x_i, z) - h(x_i, z) h(x_i, y) + \frac{1}{2} h^2(x_i, y) \right]. \tag{2}$$

While other quadratic upper bounds can be used (Suykens and Vandewalle 1999; Suykens 1999), our particular choice of loss function is motivated by being the second order Taylor expansion at zero of the exponential ranking loss (Altun et al. 2002). We now aim at finding

$$h^* = \operatorname*{argmin}_{h \in \mathcal{H}} \lambda \|h\|^2 + \sum_{i \in [\![m]\!]} \mathcal{R}(h, i). \tag{3}$$

4.4 Tensor representation

The standard representer theorem states that there is a minimiser h^* with $h^* \in \mathcal{F} = \mathbf{span}\{k[(x_i, z), (\cdot, \cdot)] \mid i \in [\![m]\!], z \in \mathcal{Y}\}$. It also holds in our case (3): Without loss of generality, we consider $h = f + g$ where $f \in \mathcal{F}$ and $g \in \mathcal{H}$ with $g \perp \mathcal{F}$. Now $h(x_i, z) = \langle h(\cdot, \cdot), k[(x_i, z), (\cdot, \cdot)] \rangle = \langle f(\cdot, \cdot), k[(x_i, z), (\cdot, \cdot)] \rangle + 0 = f(x_i, z)$ and (2) is independent of g. Furthermore $\|h(\cdot, \cdot)\|^2 = \|f(\cdot, \cdot)\|^2 + \|g(\cdot, \cdot)\|^2$ showing that for each $h \in \mathcal{H}$ there is an $f \in \mathcal{F}$ for which the value of the objective function (3) is no larger.

As usually $|\mathcal{Y}|$ grows exponentially with the input of our learning algorithm, it is intractable to optimise over functions in $\mathcal{F}$ directly. Let $e_1, \ldots, e_d$ be the canonical orthonormal bases of $\mathbb{R}^d$. We then have that $\{k_{\mathcal{X}}(x_i, \cdot) \otimes \langle e_l, . \rangle \mid i \in [\![m]\!], l \in [\![d]\!]\}$ spans $\{k_{\mathcal{X}}(x_i, .) \otimes \langle \phi(z), . \rangle \mid i \in [\![m]\!], z \in \mathcal{Y}\}$. Therefore it is sufficient to optimise over only md variables. We hence consider

$$\boldsymbol{\alpha}^* = \operatorname*{argmin}_{\boldsymbol{\alpha} \in \mathbb{R}^{m \times d}} \lambda \|f_{\boldsymbol{\alpha}}\|^2 + \sum_{i \in [\![m]\!]} \mathcal{R}(f_{\boldsymbol{\alpha}}, i) \tag{4}$$

with

$$f_{\boldsymbol{\alpha}}(x, z) = \sum_{i \in [\![m]\!], l \in [\![d]\!]} \boldsymbol{\alpha}_{il} k_{\mathcal{X}}(x_i, x) \langle e_l, \phi(z) \rangle .$$

4.5 Optimisation

Denote $f^i_{\boldsymbol{\alpha}} = \sum_{j \in [\![m]\!], l \in [\![d]\!]} \boldsymbol{\alpha}_{jl} k_{\mathcal{X}}(x_i, x_j) e_l$. Let Y be the matrix $Y \in \mathbb{R}^{m \times d}$ such that $Y_{i\cdot} = \sum_{y \in Y_i} \phi^\top(y)$ and K be the kernel matrix such that $K_{ij} = k_{\mathcal{X}}(x_i, x_j)$. We then have $f^i_{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^\top K_{i\cdot}$ and can now express $\mathcal{R}(f_{\boldsymbol{\alpha}}, i)$ using Φ, C,

$$\sum_{z \in \mathcal{Y}} f(x_i, z) = \left\langle f^i_{\boldsymbol{\alpha}}, \Phi \right\rangle,$$

$$\sum_{z \in \mathcal{Y} \setminus Y_i} f(x_i, z) = \sum_{z \in \mathcal{Y}} f(x_i, z) - \sum_{y \in Y_i} f(x_i, y),$$

$$\sum_{z \in \mathcal{Y}} f^2(x_i, z) = f^i_{\boldsymbol{\alpha}} C f^i_{\boldsymbol{\alpha}}, \quad \text{and}$$

$$\sum_{z\in\mathcal{Y}\setminus Y_i} f^2(x_i,z) = \sum_{z\in\mathcal{Y}} f^2(x_i,z) - \sum_{y\in Y_i} f^2(x_i,y).$$

With $F^i_{\boldsymbol{\alpha}} = Y_{i\cdot} f^i_{\boldsymbol{\alpha}}$ we obtain

$$\mathcal{R}(f_{\boldsymbol{\alpha}}, i) = \frac{1}{2} f^i_{\boldsymbol{\alpha}} C f^i_{\boldsymbol{\alpha}} + \left\langle f^i_{\boldsymbol{\alpha}}, \Phi \right\rangle - 2F^i_{\boldsymbol{\alpha}} - F^i_{\boldsymbol{\alpha}} \left(\left\langle f^i_{\boldsymbol{\alpha}}, \Phi \right\rangle - F^i_{\boldsymbol{\alpha}} \right).$$

Let **tr** denote the trace operator, and let diag be the operator that maps a square matrix to the column vector corresponding to its diagonal as well as a column vector to the corresponding diagonal matrix. We use $\mathbf{1}$ to denote the vector of all ones and $\mathbf{I}$ to denote the identity matrix. Using the canonical orthonormal basis of $\mathbb{R}^d$, we can write the optimisation problem (4) as

$$\begin{aligned}\operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^{d\times m}} \lambda\, \mathbf{tr}\, \boldsymbol{\alpha} K \boldsymbol{\alpha}^\top + \frac{1}{2}\, \mathbf{tr}\, K \boldsymbol{\alpha}^\top C \boldsymbol{\alpha} K + 2\Phi^\top \boldsymbol{\alpha} K \mathbf{1} + \frac{|\mathcal{Y}|}{2} \left\|\operatorname{diag}(Y \boldsymbol{\alpha} K)\right\|^2 \\ - 2|\mathcal{Y}|\, \mathbf{tr}\, Y \boldsymbol{\alpha} K - \Phi^\top \boldsymbol{\alpha} K \operatorname{diag}(Y \boldsymbol{\alpha} K).\end{aligned}$$

We can use iterative methods like Newton conjugate gradient with the gradient

$$2\lambda \boldsymbol{\alpha} K + C \boldsymbol{\alpha} K^2 + 2\Phi \mathbf{1}^\top K - Y^\top \operatorname{diag}(\Phi^\top \boldsymbol{\alpha} K) K - 2|\mathcal{Y}| Y^\top K + (|\mathcal{Y}| Y^\top - \Phi \mathbf{1}^\top)(\mathbf{I} \circ Y \boldsymbol{\alpha} K) K$$

and the product of the Hessian with vector v

$$2\lambda v K + C v K^2 + |\mathcal{Y}| Y^\top (\mathbf{I} \circ Y v K) K - \Phi \operatorname{diag}(Y v K) K - Y^\top \operatorname{diag}(\Phi^\top v K) K.$$

5 Counting super-structures

Having introduced an algorithm that can be trained whenever super-structure counting is feasible, we now show that this assumption not only holds for route prediction but also for a large class of other output sets. For some of these output sets efficient decoding is trivial, whereas for others decoding is infeasible, showing that our super-structure counting assumption is orthogonal to the assumptions made in the structured prediction literature.

5.1 Simple set system cases

We first consider the general ℓ-label prediction problem where $\mathcal{Y} = \{Z \subseteq \Sigma \mid |Z| = \ell\}$ with $\phi : \mathcal{Y} \to \mathbb{R}^\Sigma$ defined as $\phi_i(Z) = 1$ if $i \in Z$ and 0 otherwise. This setting generalises the usual multiclass ($\ell = 1$) and multilabel (by summing over all $\ell \leq |\Sigma|$) settings. For general $\ell \in [\![\Sigma]\!]$ we have (with $|\Sigma| = d$),

$$|\mathcal{Y}| = \binom{d}{\ell}; \qquad \Phi = \mathbf{1}\binom{d-1}{\ell-1}; \qquad C = \mathbf{1}\binom{d-2}{\ell-2} + \mathbf{I}\binom{d-1}{\ell-1}.$$

As special cases we have for multiclass ($\mathcal{Y} = \Sigma$) that $\Phi = \mathbf{1}, C = \mathbf{I}$, and for multilabel ($\mathcal{Y} = 2^\Sigma$) that $\Phi = 2^{|\Sigma|-1}\mathbf{1}, C = 2^{|\Sigma|-2}\mathbf{I} + 2^{|\Sigma|-2}\mathbf{1}$.

For both of these simple cases, exact decoding is very simple. For a given (test) instance x, let $\kappa \in \mathbb{R}^m$ with $\kappa_i = k_{\mathcal{X}}(x_i, x)$. For the multiclass case decoding is $\hat{y} = \operatorname{argmax}_{e\in\Sigma} [\boldsymbol{\alpha}\kappa]_e$. For the multilabel case it is $\hat{y} = \{e \in \Sigma \mid [\boldsymbol{\alpha}\kappa]_e \geq 0\}$. Hence, we could in this case also apply separation based learning algorithms.

5.2 Simple non-set system cases

We now consider poset regression. Let $\mathcal{Y} \subset \Sigma$, $\phi : \mathcal{Y} \to \mathbb{R}^{\Sigma}$ and let $(\Sigma, \succ)$ be a poset. With $\phi_i(z) = 1$ if $z \succ i$ and $\phi_i(z) = 0$ otherwise, we have $\Phi_i = |\{k \in \Sigma \mid k \succ i\}|$ and $C_{ij} = |\{k \in \Sigma \mid k \succ i \wedge k \succ j\}|$. As a special case, we have the ordinal regression problem where $\mathcal{Y} = \Sigma = [\![d]\!]$ (with d ordinal values), $\phi : \mathcal{Y} \to \mathbb{R}^{\Sigma}$ with $\phi_i(z) = 1$ if $z \geq i$ and $\phi_i(z) = 0$ otherwise. In this case $\Phi_i = |\Sigma| - i$ and $C_{ij} = |\Sigma| - \max(i, j)$. Note that hierarchical classification is also a special case of poset regression where $\succ$ forms a directed tree. In both cases, decoding can be done by exhaustively testing only $|\Sigma|$ alternatives.

5.3 Permutations

Let $\mathcal{Y}$ be the set of permutations of Σ and let $\phi : \mathcal{Y} \to \mathbb{R}^{\Sigma \times \Sigma}$. Then $|\mathcal{Y}| = |\Sigma|!$ and with $\phi_{(uv)}(z) = 1$ if $u \succ_z v$, $\phi_{(uv)}(z) = -1$ if $v \succ_z u$, and $\phi_{(uv)}(z) = 0$ otherwise, we have $\Phi = \mathbf{0}$, and

$$C_{(uv)(u'v')} = \begin{cases} -|\Sigma|! & \text{if } u = v' \wedge u' = v \\ +|\Sigma|! & \text{if } u = u' \wedge v = v' \\ \frac{+|\Sigma|!}{3} & \text{if } u = u' \text{ xor } v = v' \\ \frac{-|\Sigma|!}{3} & \text{if } u = v' \text{ xor } v = u' \\ 0 & \text{otherwise} \end{cases}$$

The assumptions made in the literature do not hold in this case as the 'without any cycle of length $\leq \ell$' edge deletion problem is NP-complete (Yannakakis 1978).

5.4 Posets and partial tournaments

Consider $\Sigma = [\![N]\!] \times [\![N]\!]$; $\mathcal{Y} \subseteq 2^{\Sigma}$ such that all $y \in \mathcal{Y}$ form a poset (an acyclic directed graph closed under transitivity) $([\![N]\!], y)$; as well as $\phi : \mathcal{Y} \to \mathbb{R}^{\mathcal{Y}}$ with $\phi_{uv}(z) = 1$ if $(u, v) \in z$, $\phi_{uv}(z) = -1$ if $(v, u) \in z$, and $\phi_{uv}(z) = 0$ otherwise. To the best of our knowledge, no exact way to compute $C_{e,e'}$ is known. However, we can relax $\mathcal{Y}$ to $\tilde{\mathcal{Y}} \subseteq 2^{\Sigma}$ such that all $y \in \mathcal{Y}$ form a partial tournament (a directed graph with no cycles of length two). With $\eta = N(N-1)/2$ we have $|\tilde{\mathcal{Y}}| = 3^{\eta}$, $\Phi = \mathbf{0}$, and

$$\tilde{C}_{(uv),(u'v')} = \begin{cases} -2 \cdot 3^{|\eta|-1} & \text{if } u = v' \wedge u' = v \\ +2 \cdot 3^{|\eta|-1} & \text{if } u = u' \wedge v = v' \\ +2 \cdot 3^{|\eta|-2} & \text{if } u = u' \text{ xor } v = v' \\ -2 \cdot 3^{|\eta|-2} & \text{if } u = v' \text{ xor } v = u' \\ 0 & \text{otherwise} \end{cases}$$

The assumptions made in the literature do not hold in this case as the 'transitive digraph' edge deletion problem is NP-complete (Yannakakis 1978).

5.5 Graph prediction

Consider graphs on a fixed set of vertices, that is $\Sigma = \{U \subseteq [\![N]\!] \mid |U| = 2\}$ and a property $\pi : 2^{\Sigma} \to \Omega$ such as acyclicity, treewidth bounded by a given constant, planarity, outerplanarity bounded by a constant, clique etc. Let $\mathcal{Y}^{\pi}$ and $\phi^{\in}$ be defined as in Sect. 3.5. For

properties like clique, we can compute $\mathcal{Y}$, Φ, C:

$$|\mathcal{Y}^{\text{clique}}| = 2^N, \qquad \Phi^{\text{clique}}_{\{u,v\}} = \sum_{i=2}^{N} \binom{N-2}{i-2}, \qquad C^{\text{clique}}_{e,e'} = \sum_{i=|e\cap e'|}^{N} \binom{N-|e\cap e'|}{i-|e\cap e'|}.$$

For other properties, no way to compute C might be known but we can always relax $\mathcal{Y}$ to $\tilde{\mathcal{Y}} = 2^{\Sigma}$. We then have $\tilde{\Phi} = 2^{|\Sigma|-1}$ and $\tilde{C}_{e,e'} = 2^{|\Sigma|-|e\cup e'|}$.

6 Related work

Discriminative (non-probabilistic) methods for structured prediction were proposed by Collins (2002), Taskar et al. (2005) and Tsochantaridis et al. (2005). To ensure efficient optimisation, these methods make different assumptions and were discussed in Sect. 3.4. Weston et al. (2002) used kernel-PCA on output spaces with arbitrary kernel to obtain a finite dimensional embedding. Then, regularised least-squares regression was used to learn the mapping from the input to the projection of the output onto each individual principal direction. A so-called pre-image problem (Schölkopf et al. 1999) has to be solved for prediction. Cortes et al. (2005) extended this setting and gave an algorithm that computes an exact pre-image for n-gram kernels if one exists. For other cases we are not aware of any guarantees; instead heuristics were used, e.g., for graphs (Bakir et al. 2004). Similar loss functions to the one used in this paper (2) were considered in previous work on structured prediction problems. Altun et al. (2002) introduced the ranking loss for structured prediction and minimised an upper bound given by the exponential loss for discriminative sequence labeling.

Ricci et al. (2007) proposed to minimise the Z-score of the scoring function h. In their formulation, only linear scoring functions $h(x, y) = \langle w, \phi(x, y)\rangle$ were considered and regularisation was not used. For the special case of string outputs, they gave a polynomial time algorithm for optimising the Z-score. For other structured prediction cases, we are not aware of efficient algorithms for optimising the Z-score. Ricci et al. (2008) extended this setting and designed other structured output training algorithms that can also be trained whenever Φ and C can be computed efficiently. A structured prediction framework that bypasses the decoding problem using search heuristics was proposed by Daumé and Marcu (2005). Recently, Finley and Joachims (2008) analysed the performance of SVM-Struct when exact decoding is intractable. In such cases, SVM-Struct still terminates in polynomial time, but the resulting solution may be suboptimal or incorrect (Finley and Joachims 2008).

7 Empirical evaluation

In this section, we compare our counting-based structured prediction algorithm (CSOP) with general as well as special purpose algorithms on three different problems. For two of these problems efficient decoding is possible, and we show that on these problems our approach is competitive with general as well as with special purpose state-of-the-art algorithms. The last problem is a simulated route prediction task for which we focus on the training routine. That is, the synthetic data is generated from a hidden policy and the learning algorithms aim at reconstructing this policy. In all experiments, we fixed the regularisation parameter of our algorithm $\lambda = |\mathcal{Y}| \cdot \sum_{i\in[\![m]\!]} |Y_i|/m$.

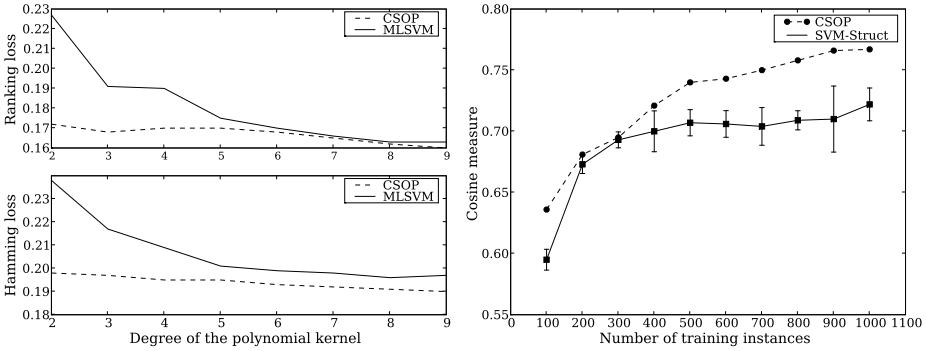

Fig. 1 *Left:* Comparison of multi-label SVM (MLSVM) and our algorithm (CSOP) on multi-label classification. *Right:* Comparison of SVM-Struct and CSOP on dicycle policy estimation

7.1 Multi-label classification

We compared the performance of our algorithm with the kernel method proposed by Elisseeff and Weston (2001) that directly minimises the ranking loss (number of pairwise disagreements). We followed the same experimental set up (dataset and kernels) as described in Elisseeff and Weston (2001). We used the *Yeast* dataset consisting of 1500 training and 917 test genes with 14 labels, and trained our algorithm with polynomial kernel of varying degree (2–9). Figure 1 shows the results for Hamming loss and ranking loss.

7.2 Hierarchical classification

We trained our algorithm on the WIPO-alpha patent dataset (Rousu et al. 2005) consisting of 1352 training and 358 test documents. The number of nodes in the hierarchy is 188 with maximum depth of 3. Each document belongs to exactly one leaf category.The performance of several algorithms (results are taken from Rousu et al. 2005) is shown in Table 1. $\ell_{0/1}$ denotes the zero-one loss in percentage, ℓ_Δ is the average Hamming loss per instance, and ℓ_H is the hierarchical loss (average per instance). The hierarchical loss is based on the intuition that if a mistake is made at node i, then further mistakes made in the subtree rooted at i are unimportant (Cesa-Bianchi et al. 2004). Formally, for any pair of hierarchical labels z and y,

$$\ell_H(z, y) = \sum_{i=1}^{|\Sigma|} I(\phi_i(z) \neq \phi_i(y) \wedge \phi_j(z) = \phi_j(y),\ \ j \in ANC(i)),$$

where $ANC(i)$ is the set of ancestors of i, and $I(p)$ is 1 if p is true and 0 otherwise. In this way the hierarchy or taxonomy of the problem domain is taking into account. SVM denotes an SVM trained for each microlabel independently, H-SVM denotes an SVM trained for each microlabel independently and using only those samples for which the ancestor labels are positive, H-RLS is the hierarchical least-squares algorithm described in Cesa-Bianchi et al. (2004) and H-M^3 is the kernel-based algorithm proposed by Rousu et al. (2005) which uses the maximum margin Markov network framework. The two different versions of this algorithm correspond to using the Hamming loss and the hierarchical loss during training. While the performance on the Hamming loss was comparable to the baseline SVM, our algorithm resulted in best performance on the $\ell_{0/1}$ loss and the hierarchical loss.

Table 1 Comparison of various algorithms on hierarchical classification

	SVM	H-SVM	H-RLS	H-M^3 $-\ell_\Delta$	H-M^3 $-\ell_{\tilde{H}}$	CSOP
$\ell_{0/1}$	87.2	76.2	72.1	70.9	65.0	51.1
ℓ_Δ	1.84	1.74	1.69	1.67	1.73	1.84
ℓ_H	0.053	0.051	0.050	0.050	0.048	0.046

7.3 Route prediction

We experimented with an artificial setting due to the lack of real world data sets on route prediction. We simulate the problem of predicting the cyclic tour of different people. We assume that there is a hidden policy for each person and he/she takes the route that (approximately) maximises the reward of the route. In this setting of dicycle prediction, we can check how well the estimated policy $f^i_{\boldsymbol{\alpha}}$ approximates the hidden policy in the test set ($i \in [\![m+1, m+m']\!]$). The data is constructed as follows: (i) generate M matrices $A^{(j)} \in \mathbb{R}^{\Sigma\times\Sigma}$ uniformly at random with entries in the interval $[-1, 1]$ and $A^{(j)}_{uv} = -A^{(j)}_{vu}$; (ii) generate $(m+m')M$ random numbers uniformly between 0 and 1 to form the inputs $x_i \in \mathbb{R}^M$; (iii) create the output structures $y_i \approx \operatorname{argmax}_{y\in\mathcal{Y}} \sum_{(u,v)\in y, j\in[\![M]\!]} x_{ij} A^{(j)}_{uv}$ for training, that is for all $i \in [\![m]\!]$. On the test set, we evaluated our algorithm by cosine similarity of the learned policy and the true policy:

$$\sum_{i\in[\![m+1,m+m']\!]} \left\langle f^i_{\boldsymbol{\alpha}}, \sum_{j\in[\![M]\!]} x_{ij} A^{(j)} \right\rangle \cdot \left\| f^i_{\boldsymbol{\alpha}} \right\|^{-1} \cdot \left\| \sum_{j\in[\![M]\!]} x_{ij} A^{(j)} \right\|^{-1}$$

Figure 1 shows a plot of the cosine measure on an experiment ($m' = 500$, $M = 15$, $|\Sigma| = 10$) for varying number of training instances. As expected, we see that our algorithm is able to estimate the true policy with increasing accuracy as the number of training instances increases. The plot also shows the performance of SVM-Struct using approximate decoding during training. Approximate decoding is performed by randomly sampling a couple of cyclic permutations and using the best scoring one (the number of repetitions used in our experiments was 25). The results were unstable due to approximate decoding and the results shown on the plot are averages from 5 trials. For SVM-Struct, we experimented with several values of the regularisation parameter and report the best results obtained on the test set—though this procedure gives an advantage to SVM-Struct, our approach still resulted in better performance.

8 Conclusions

We investigated the assumptions underlying state-of-the-art structured output training algorithms for several output sets. We showed that these assumptions do not hold for the relevant problem of route prediction and for several other problems related to edge deletion. In particular, we showed that there is no polynomial time algorithm for finding violated constraints in these cases (unless P = coNP) and that deciding if there is no violated constrained has no short certificate (unless NP = coNP).

Despite these hardness results, we showed that for route prediction efficient formulae for 'super-structure' counting can be derived and proposed an alternative structured output training algorithm based on these counting formulae. By deriving 'super-structure' counting

formulae for various combinatorial structures, we showed that our approach subsumes many machine learning problems including multi-class, multi-label and hierarchical classification. Furthermore, it can be used for training complex combinatorial output sets for which the assumptions made in the literature do not hold.

For prediction, inference can naturally not be avoided. While in this work we focused mainly on training algorithms, approximation algorithms for decoding set systems and independence systems can be found in Gärtner and Vembu (2008). We note that it is non-trivial to design approximation algorithms for the decoding problem of the combinatorial structures considered in this paper. Indeed, there are hardness of approximation results for the maximum acyclic subgraph problem (Guruswami et al. 2008) and the problem of finding longest directed cycles (Björklund et al. 2004). An interesting future direction would to explore techniques like simulated annealing (Kalai and Vempala 2006) to perform approximate inference for the prediction step.

We empirically compared our algorithm with state-of-the-art general and special purpose algorithms on different structures. For multi-label and hierarchical classification, decoding is trivial and we showed that predictions from our approach are competitive or better than the state-of-the-art. For route prediction, decoding is hard and we focused on the training part of the algorithms, i.e., on synthetic data we compared the policy estimated by our approach to the policy estimated by SVM-Struct using approximate decoding.

While in this work we minimised a quadratic loss function, we note that it is possible to train a probabilistic discriminative model by using the negative log-likelihood as the loss function. The log-partition function and its gradient can then be approximated using sampling algorithms for combinatorial structures (Jerrum and Sinclair 1996; Randall 2003). Some results in this direction can be found in the recent work of Vembu et al. (2009).

References

Altun, Y., Hofmann, T., & Johnson, M. (2002). Discriminative learning for label sequences via boosting. In *Advances in neural information processing systems* (Vol. 15).

Bakir, G., Zien, A., & Tsuda, K. (2004). Learning to find graph pre-images. In *Proceedings of the annual symposium of the German association for pattern recognition (DAGM)*.

Björklund, A., Husfeldt, T., & Khanna, S. (2004). Approximating longest directed paths and cycles. In *Proceedings of the international colloquium on automata, languages and programming*.

Cesa-Bianchi, N., Gentile, C., Tironi, A., & Zaniboni, L. (2004). Incremental algorithms for hierarchical classification. In *Advances in Neural Information Processing Systems* (Vol. 17).

Collins, M. (2002). 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*.

Cortes, C., Mohri, M., & Weston, J. (2005). A general regression technique for learning transductions. In *Proceedings of the international conference on machine learning*.

Daumé, H. III, & Marcu, D. (2005). Learning as search optimization: approximate large margin methods for structured prediction. In *Proceedings of the international conference on machine learning*.

Elisseeff, A., & Weston, J. (2001). A kernel method for multi-labelled classification. In *Advances in neural information processing systems* (Vol. 14).

Erhan, D., L'heureux, P. J., Yue, S. Y., & Bengio, Y. (2006). Collaborative filtering on a family of biological targets. *Journal of Chemical Information and Modeling*, *46*(2), 626–635.

Finley, T., & Joachims, T. (2008). Training structural SVMs when exact inference is intractable. In *Proceedings of the international conference on machine learning*.

Froehlich, J., & Krumm, J. (2008). Route prediction from trip observations. In *Society of automotive engineers world congress*.

Gärtner, T., & Vembu, S. (2008). Learning to predict combinatorial structures. Unpublished manuscript.

Guruswami, V., Manokaran, R., & Raghavendra, P. (2008). Beating the random ordering is hard: Inapproximability of maximum acyclic subgraph. In *Proceedings of the annual IEEE symposium on foundations of computer science*.

Jacob, L., & Vert, J. P. (2008). Protein-ligand interaction prediction: an improved chemogenomics approach. *Bioinformatics*, *24*(19), 2149–2156.

Jerrum, M., & Sinclair, A. (1996). The Markov chain Monte Carlo method: an approach to approximate counting and integration. In D. S. Hochbaum (Ed.), *Approximation algorithms for NP-hard problems* (pp. 482–520). Boston: PWS-Kent.

Kalai, A. T., & Vempala, S. (2006). Simulated annealing for convex optimization. *Mathematics of Operations Research*, *31*(2), 253–266.

Korte, B., & Vygen, J. (2008). *Algorithms and combinatorics: Vol. 21. Combinatorial optimization: theory and algorithms*. Berlin: Springer.

Papadimitriou, C. H. (1994). *Computational complexity*. Addison-Wesley: Reading

Randall, D. (2003). Mixing. In *Proceedings of the annual IEEE symposium on foundations of computer sciences*.

Ricci, E., Bie, T. D., & Cristianini, N. (2007). Discriminative sequence labeling by Z-score optimization. In *Proceedings of the European conference on machine learning*.

Ricci, E., Bie, T. D., & Cristianini, N. (2008). Magic moments for structured output prediction. *Journal of Machine Learning Research*, *9*, 2803–2846.

Rousu, J., Saunders, C., Szedmák, S., & Shawe-Taylor, J. (2005). Learning hierarchical multi-category text classification models. In *Proceedings of the international conference on machine learning*.

Schölkopf, B., Mika, S., Burges, C. J. C., Knirsch, P., Müller, K. R., Rätsch, G., & Smola, A. J. (1999). Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, *10*(5), 1000–1017.

Schölkopf, B., Herbrich, R., & Smola, A. J. (2001). A generalized representer theorem. In *Proceedings of the annual conference on learning theory*.

Suykens, J. (1999). Multiclass least squares support vector machines. In *Proceedings of the international joint conference on neural networks*.

Suykens, J., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters*, *9*(3), 293–300.

Taskar, B., Chatalbashev, V., Koller, D., & Guestrin, C. (2005). Learning structured prediction models: A large margin approach. In *Proceedings of the international conference on machine learning*.

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

Vembu, S., Gärtner, T., & Boley, M. (2009). Probabilistic structured predictors. In *Proceedings of the annual conference on uncertainty in artificial intelligence*.

Weston, J., Chapelle, O., Elisseeff, A., Schölkopf, B., & Vapnik, V. (2002). Kernel dependency estimation. In *Advances in neural information processing systems* (Vol. 15).

Yannakakis, M. (1978). Node- and edge-deletion NP-complete problems. In *Proceedings of the annual ACM symposium on theory of computing*.