Learning Structured Outputs via Kernel Dependency Estimation and Stochastic Grammars

Fabrizio Costa[1], Andrea Passerini[1], and Paolo Frasconi[1]

Dipartimento di Sistemi e Informatica, Università degli Studi di Firenze

Abstract. We focus on graph-valued outputs in supervised learning and propose a novel solution to the pre-image problem in the kernel dependency estimation framework. Output structures are generated by a stochastic grammar and the output feature space is directly associated with the set of productions for the grammar. The regression estimation step learns to map input examples into a feature vector that counts the number of applications of each production rule. A max-propagation algorithm finally builds the predicted output according to the normalized counts. We test our method on a ambiguous context free grammar (CFG) parse tree reconstruction problem. We show on an artificial dataset that mimics the prepositional attachment problem how learning the number of applications of each production rule on a per example base allows CFG parser to better tackle ambiguity issues.

1 Introduction

A structured output prediction problem can be formulated as a supervised learning problem in which the output or target space is not restricted in any way and can be any set (e.g. a set of graphs or sequences). In this way, several predictions can be made collectively on the same input instance, still maintaining that instances are sampled independently. Formally, we denote by $\mathcal{X}$ and $\mathcal{Y}$ the input and output spaces, respectively. Examples are in the form $\mathcal{D} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ with $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ and are sampled identically and independently from a fixed and unknown distribution p. The cost associated with prediction errors is measured by a loss function $V : \mathcal{Y} \times \mathcal{Y} \mapsto R$. Learning consists of finding a function $f : \mathcal{X} \mapsto \mathcal{Y}$ in a given hypothesis space such that the average loss $V(y, f(x))$ is minimized.

The above setting can be useful in many real world application domains. Structural bioinformatics offers interesting examples since there are several prediction tasks where the target is about a relation between two or more residues in a protein chain. Problems of this kind include prediction of disulfide bridges, metal binding sites, beta sheet partners and contact maps. Correlation often plays a major role linking the prediction at different residues in the same chain. It is relatively safe, however, to assume that different protein chains are sampled independently. Thus, these structural bioinformatics problems fit the above

framework choosing $\mathcal{X}$ to be a set of sequences and $\mathcal{Y}$ a set of graphs (whose vertices are the elements of the input sequence). Computational linguistics offers other interesting examples of sequential translation problems like POS-tagging and named entity recognition.

Typically, structured output prediction involves searching the output space $\mathcal{Y}$, where the search can be guided by a scoring function associated with the input x and a candidate output y.

$$f(x) = \arg\max_{y \in \mathcal{Y}} S(x, y). \tag{1}$$

Exhaustive search in spaces of graphs is generally intractable and heuristic methods are necessary. Search in graph space can be implemented starting in some initial state and iteratively moving in $\mathcal{Y}$ by applying some modification operator to the current structure. In [2, 6], parse tree construction from input sequences is formulated in the context of incremental dynamic grammars and recursive neural networks were employed to learn the best operator to be applied at each search step. In some special cases it is possible to define "unimodal" functions $S(x, y)$ for which hill climbing can be shown to be complete in solving Eq. 1. In [12], a unimodal function was proposed for scoring candidate protein contact maps and a recursive neural network was trained in regression mode to predict the value $S(x, y)$ to be plugged in Eq. 1.

Weston et al. [13] and Tsochantaridis et al. [10] have proposed kernel-based formulations of the problem in Eq. 1. Here we follow-up the kernel dependency estimation (KDE) framework introduced in [13] and further specialized to the case of sequential transductions in [1].

2 Kernel Dependency Estimation

In KDE [13, 1], both the input and the output portions of the data are mapped into their feature spaces, denoted as $\mathcal{F}_\mathcal{X}$ and $\mathcal{F}_\mathcal{Y}$, respectively. As usual, these mappings can be implicitly defined via two kernel functions:

$$\kappa(x, x') = \langle \phi(x), \phi(x') \rangle \tag{2}$$

$$\lambda(y, y') = \langle \psi(y), \psi(y') \rangle \tag{3}$$

where $\phi : \mathcal{X} \mapsto \mathcal{F}_\mathcal{X}$ and $\psi : \mathcal{Y} \mapsto \mathcal{F}_\mathcal{Y}$ are the input and the output feature mapping, respectively. Then every function $f : \mathcal{X} \mapsto \mathcal{Y}$ in the original domain, can be mapped to a corresponding function $F : \mathcal{F}_\mathcal{X} \mapsto \mathcal{F}_\mathcal{Y}$ in the transformed space defined as $F(\phi(x)) = \psi(f(x))$. KDE consists of two separate steps: feature estimation and pre-image calculation, as detailed below.

Feature estimation problem. This step consists of predicting the image of the target $\psi(y)$ given the input x. We need the assumption that $\mathcal{F}_\mathcal{Y}$ has finite dimension n_o (or alternatively we could apply kernel PCA to reduce its dimensionality to a finite integer).

The feature estimation problem can be conveniently represented as a vector-output regression problem where one learns a function $g : \mathcal{X} \mapsto \mathcal{F}_{\mathcal{Y}}$ from examples $\{(x_i, \psi(y_i))\}$. It can be shown that using kernel ridge regression, the n_o regression problems can be solved using a single matrix inversion, obtaining the general solution

$$g(x) = \sum_{i=1}^{m} c_i \kappa(x, x_i) \tag{4}$$

being $c_i \in \boldsymbol{R}^{n_0}$ the columns of the solution to the following linear problem:

$$C = \Psi(y)(K + \gamma m I_m)^{-1} \tag{5}$$

being K the input kernel matrix and $\Psi(y)$ the $n_o \times m$ matrix with columns $\psi(y)$. Efficient alternatives to the above approach consist of solving each of the n_o regression problems by support vector regression [11] or using maximum margin regression [8, 7].

Pre-image calculation problem. Once the feature space representation of the target has been estimated as $g(x)$, we are left with the problem of inverting the output feature mapping $\psi(y)$ in order to obtain a predicted $f(x) \in \mathcal{Y}$. The approach suggested in [13, 1] consists of searching the space $\mathcal{Y}$ for a structure whose image in $\mathcal{F}_{\mathcal{Y}}$ is close to $g(x)$:

$$f(x) = \arg\min_{y \in \mathcal{Y}} \|g(x) - \psi(y)\|^2 \tag{6}$$

When using kernel ridge regression, it can be shown that

$$\|g(x) - \psi(y)\|^2 = \lambda(y, y) - 2 \sum_{i,j=1}^{m} h_{ij} \lambda(y_i, y) \kappa(x_j, x) \tag{7}$$

being $H = \{h_{ij}\} = (K + \mu I)^{-1}$. In [1] the search problem of Eq.(6) is solved by a graph theoretical algorithm in the case of output strings and k-gram output kernels.

3 Using stochastic grammars

We propose here an alternative KDE formulation where the pre-image problem is solved via probabilistic inference in stochastic grammars. We focus on supervised problems where input instances are arbitrary objects (e.g. strings) and the associated targets y are the result of a generative mechanism described by a stochastic grammar which depends on the input instance. One example is language learning where x is a string in a finite alphabet T and y a parse tree of x (which is not necessarily unique if the grammar is ambiguous). Another example is string transduction where x and y are strings in different alphabets and y is generated by selecting suitable production rules on the basis of x. Let $\mathcal{G}(x) = \{N, T, \mathcal{S}, \Pi(x)\}$ denote the stochastic grammar associated with x, where

N is the set of nonterminal symbols, T the set of terminal symbols, $\mathcal{S}$ the set of production rules and $\Pi(x)$ the corresponding probabilities. In our notation we have made explicit the assumption that the structure $\mathcal{S}$ of the grammar is fixed and given as background knowledge to the learner, while the parameters $\Pi(x)$ depend on x and are unknown. Let $\mathcal{L}(x)$ be the language generated by $\mathcal{G}(x)$ and for $y \in \mathcal{L}(x)$ let $\Pr(y|x)$ denote the probability that $\mathcal{G}(x)$ has generated y: in order to solve the statistical learning problem outlined in the introduction, we need that $\Pr(y|x)$ be a good model for $p(y|x)$.

We link this generative process to KDE by choosing as features $\psi(y)$ a real vector from which $\Pi(x)$ can be conveniently obtained.

While in this phase there is in principle no constraint on the kind of stochastic grammar to employ (context free, context sensitive, or even more expressive!), when we finally get down to the computation of the pre-image of $\psi(y)$ we have to opt for computationally tractable methods. To this end we choose a stochastic context free grammar to model the structured output information so to exploit efficient maximum-propagation algorithms (such as Viterbi [3]) which can, in practice, run with complexity lower than $O(|x|^3)$. In this case $\psi(y)$ is a vector of frequency counts for the rules used in the parse tree y.

Formally, suppose $\mathcal{G}(x)$ is a stochastic context free grammar. Production rules $r_{k\ell}$ have the form $A_k \mapsto \alpha_\ell$ with $A_k \in N$ and $\alpha_\ell \in (N \cup T \cup \{\epsilon\})^*$. Each production rule $r_{k\ell}$ has an attached probability $\pi_{k,\ell}$ with constraints $\sum_\ell \pi_{k,\ell} = 1$ for each $k = 1 \ldots, |N|$. These probabilities are linked to the feature vector $\psi(y)$ by the softmax function:

$$\pi_{k,\ell} = \frac{e^{\psi_{k,\ell}}}{\sum_{j=1} e^{\psi_{k,j}}}. \tag{8}$$

In this way, the feature estimation step of KDE consists of solving the regression problem for a multinomial logit model, i.e. a generalized linear model [5]. Note how the expressive power of the grammar describing the output structure is greater than simple SCFG as the probabilities associated to each rule depend on the inputs x. Informally, first we learn $\psi(y)$ i.e. the frequency counts of each rule in $\mathcal{G}$ for the parse tree y then we give these estimates to a SCFG parser that computes the pre-image of $\psi(y)$ i.e. builds the actual parse tree. We call the overall procedure KDE-SCFG.

For another approach to the task of learning the structured output of a SCFG parse tree in terms of its production rules see [9] where they learn a kernel machine that discriminates among the entire space of parse trees factorized in an extended bottom-up tabular representation.

4 Experiments

4.1 The artificial task

We test the proposed method on an artificial dataset. We are interested in problems where instances' output structure cannot be well explained resorting only

to a SCFG (in this case we could use the standard parsing techniques). We simulate a problem of interest in the NLP domain known as the PP-attachment ambiguity resolution problem which is known to be context-sensitive. The task consists in deciding which of two possible structural parse tree that involve a preposition is the correct one. To clarify the issue consider the following two sentences "eat a salad with a fork" and "eat a salad with tomatoes": in the first one the propositional phrase (PP) "with a fork" specifies a characteristic of the action of eating, while in the second case it specifies a property of the salad; in the first case we have a parse tree where the PP is attached to the verb 'eat' as in: (VP (V eat) (NP a salad) (PP with a fork)) while in the second case the PP is attached to the noun 'salad' as in (VP (V eat) (NP a salad (PP with tomatoes))). A CFG that has access to the part of speech (POS) of the sentence words only has no way to discriminate between the two alternatives which are both syntactically correct: the disambiguation can happen only lexicalizing the grammar i.e. making the rules dependent on the actual words, which is a way to introduce a form of context-sensitiveness.

We simulate the PP-attachment problem using the following simple stochastic context free grammar $\mathcal{G}$:

$$\begin{array}{ll} S \rightarrow ScS|NV & w \rightarrow 5 \\ V \rightarrow wNP|vN_P & v \rightarrow 4 \\ N \rightarrow n|ncV & n \rightarrow 2|3 \\ N_P \rightarrow nP|ncVP & p \rightarrow 1 \\ P \rightarrow pn & c \rightarrow 0 \end{array}$$

where $N = \{S, V, N, N_P, P, c, n, p, v, w\}$ and $T = \{0, 1, 2, 3, 4, 5\}$ and $\{c, n, p, v, w\}$ are the POS tags (pre-terminal). The probabilities are all uniform except for the S derivation for which we set a .2 probability of deriving ScS against a .8 probability for NV.

To introduce the context-sensitiveness we collapse the POS tags 'v' and 'w' in a single tag 'x': now, given the structure of $\mathcal{G}$ and a dataset wide global estimate of Π, the SCFG parser has no deterministic information as to which expansion rule to use for the verbal phrase: the resulting grammar is ambiguous. We want to show that exploiting similarities between the inputs in sequential form we can learn the probabilities associated to the different rules used to resolve the ambiguity and inform the SCFG parser on a per example base on which rule to prefer hence obtaining a better parse tree.

4.2 Data preparation

We used Douglas Rohde's Simple Language Generator (SLG) program[1] to randomly produce sets of sentences according to $\mathcal{G}$. We then post-processed the sets

[1] http://tedlab.mit.edu/ dr/SLG/

with two strategies: in the first one (Natural) we filtered out sentences with the same sequence structure (and therefore with the same parse trees), while in the second set (Unique) we filtered out sentences with identical representation in the output feature space $\mathcal{F}_\mathcal{Y}$.

4.3 Results

We compared KDE-SCFG with a standard SCFG parser that makes use of probabilities globally estimated over the entire dataset. We employed a spectrum kernel [4] with k-mers of size 2 to 5 to compute κ. We use Collin's `evalb` program[2] to compute the bracketing F-measure and exact parse matching scores. We randomly split the dataset in two equally sized sets of 1,000 instances each, which were employed for model selection and final evaluation respectively, both performed by a 5-fold cross validation procedure. Table 1 reports micro-averaged results of the 5-fold cross validation evaluation procedure, both for the entire evaluation set and focused on short sequences (< 35 terminals) only. The results indicate that the KDE-SCFG approach outperforms the SCFG parser significantly ($p < .05$ in all pairwise comparisons).

Table 1. Comparison between standard SCFG and KDE-SCFG

Filtering	Natural		Unique	
Measure	F-score	Exact	F-score	Exact
$\text{SCFG}_{<35}$	86.4	10.3	84.7	3.1
SCFG	85.8	8.1	84.4	0.5
$\text{KDE-SCFG}_{<35}$	93.3	33.2	94.3	28.6
KDE-SCFG	91.5	26.1	89.6	4.8

5 Conclusions

We introduced a novel solution to the pre-image problem for kernel dependency estimation using an output feature space associated to the frequency of context free grammar production rules. We showed that learning their frequency on a per instance base is an effective way to approximate a specific context sensitive grammar on a simplified NLP problem significantly outperforming a standard stochastic context free parser. The key ideas introduced in this (preliminary) work can in principle be extended to more complex settings: as an example consider associating the feature space with clauses in a probabilistic inductive logic programming setting and running logical inference procedures to find an explanation (proof) for the target concept.

[2] http://nlp.cs.nyu.edu/evalb/

6 Acknowledgement

This work was in part supported by the APrIL Project under contract no. FP6-508861. The authors would like to thank Manfred Jaeger from Aalborg University for useful comments and discussions.

References

1. C. Cortes, M. Mohri, and J. Weston. A General Regression Technique for Learning Transductions. *Proceedings of the 22nd international conference on Machine learning*, pages 153–160, 2005.
2. F. Costa, V. Lombardo, P. Frasconi, and G. Soda. Wide coverage incremental parsing by learning attachment preferences. In F. Esposito, editor, *AI*IA 2001: Advances in Artificial Intelligence, 7th Congress of the Italian Association for Artificial Intelligence*, volume 2175 of *Lecture Notes in Computer Science*, pages 297–307. Springer, 2001.
3. Karim Lari and Steve J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.
4. C. Leslie, E. Eskin, and W.S. Noble. The spectrum kernel: a string kernel for svm protein classification. In *Proc. of the Pacific Symposium on Biocomputing*, pages 564–575, 2002.
5. P. McCullagh and J.A. Nelder. *Generalized Linear Models*. Chapman & Hall, 1983.
6. P. Sturt, F. Costa, V. Lombardo, and P. Frasconi. Learning first-pass attachment preferences with dynamic grammars and recursive neural networks. *Cognition*, 88(2):133–169, 2003.
7. S. Szedmak and J. Shawe-Taylor. Multiclass and Multiview Learning at One-class Complexity. Technical report, ISIS Group Electronics and Computer Science, 2005.
8. S. Szedmak, J. Shawe-Taylor, and E. Parado-Hernandez. Learning via Linear Operators: Maximum Margin Regression. Technical report, Pascal Research Reports, 2005.
9. Ben Taskar, Dan Klein, Mike Collins, Daphne Koller, and Christopher Manning. Max-margin parsing. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 1–8, Barcelona, Spain, July 2004. Association for Computational Linguistics.
10. I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large Margin Methods for Structured and Interdependent Output Variables. *The Journal of Machine Learning Research*, 6:1453–1484, 2005.
11. V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.
12. A. Vullo and P. Frasconi. A bi-recursive neural network architecture for the prediction of protein coarse contact maps. In *1st IEEE Computer Society Bioinformatics Conference (CSB'02)*, pages 187–196, Stanford, CA, 2002.
13. J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik. Kernel dependency estimation. In *NIPS*, pages 873–880, 2002.