Planar Languages and Learnability

Alexander Clark[1], Christophe Costa Florêncio[1],
Chris Watkins[1], and Mariette Serayet[2]

[1] Department of Computer Science, Royal Holloway,
University of London, Egham TW20 0EX, UK
alexc@cs.rhul.ac.uk, chris@cs.rhul.ac.uk, chrisw@cs.rhul.ac.uk
[2] Faculté des Sciences et Techniques, Département Informatique, 23,
Rue du Docteur Paul Michelon, 42023 Saint-Etienne Cedex 2, France
mariette.serayet@bvra.univ-st-etienne.fr

Abstract. Strings can be mapped into Hilbert spaces using feature maps such as the Parikh map. Languages can then be defined as the pre-image of hyperplanes in the feature space, rather than using grammars or automata. These are the *planar* languages. In this paper we show that using techniques from kernel-based learning, we can represent and efficiently learn, from positive data alone, various linguistically interesting context-sensitive languages. In particular we show that the cross-serial dependencies in Swiss German, that established the non-context-freeness of natural language, are learnable using a standard kernel. We demonstrate the polynomial-time identifiability in the limit of these classes, and discuss some language theoretic properties of these classes, and their relationship to the choice of kernel/feature map.

1 Introduction

Formal languages, whether used in linguistics or in computer science, have traditionally been represented either by grammars or by various simple machine formalisms. In linguistics context-free grammars have been widely used as a representative tool. With the discovery of demonstrably non-context-free phenomena in syntax, most famously in Swiss German ([Huy84, Shi85]), attention switched away from grammatical formalisms based on CFG, such as GPSG, to more powerful formalisms that were capable of modelling these phenomena: these are generally restricted to the mildly context-sensitive languages (see [JS96] for an overview).

From a linguistic point of view these formalisms have a number of desirable properties, but from a learnability point of view they are far too unrestricted. Even very simple classes of these formalisms, such as acyclic deterministic finite state automata, are already unlearnable when using modern characterisations of learnability, because intractable cryptographic problems can be embedded in the learning of these tasks.

Chomsky correctly identified accounting for the learnability of language as one of the principal challenges for linguistics. The Principles and Parameters approach is one solution to this problem: by specifying a class of languages

Y. Sakakibara et al. (Eds.): ICGI 2006, LNAI 4201, pp. 148–160, 2006.

parametrised by a small set of parameters the learnability problem can be simplified. Unfortunately problems of lexical acquisition, cross-language ambiguity and the intertwining of parameter settings meant that no satisfactory theory of parameter-based learning has ever been presented. In this paper, rather than taking an overexpressive class of representations and trying to make it learnable by imposing additional restrictions on it, we take an alternative route. We restrict ourselves to a class of representations that are inherently learnable from positive data alone. At this point the class has a limited range, yet it is capable of representing some classic examples of mildly context-sensitive languages.

We define the class of *planar* languages. These are languages that correspond to planes in a feature space. Using standard linear algebra techniques we are then able to learn the minimal plane that contains the data points from positive data alone. This does not restrict us to (semi)linear languages: the function that maps datapoints to feature vectors can, and generally will, do so in a non-linear fashion. This geometrical representation of languages allows us to apply the mature theory of linear algebra to the problems of grammatical inference.

We give a simple example before giving formal definitions. Consider the language L defined as the set of all strings containing equal numbers of as and bs. If we define a feature map from this set of strings to a plane, where the first coordinate is the number of as in the string and the second coordinate the number of bs, so that aab is mapped to the point $(2, 1)$ and so on, we see that all of the strings in the language will be mapped to points that lie on the line $x = y$, such as $(1, 1)$, $(2, 2)$ and so on. Additionally we can see that every string whose image lies on this line is also in the language. Thus L can be defined using a one-dimensional hyperplane in this feature space, and is thus a planar language.

Obviously the feature mapping limits the class of languages that can be defined, and with this trivial mapping, the Parikh map, only very few languages can be characterised. A major reason for this is that the feature mapping is not *injective*, i.e. there are pairs of distinct strings, such as ab and ba, that are mapped to the same point in the feature space, $(1, 1)$. In this paper we will use the implicit feature maps defined by string kernels ([Wat99]), in particular the gap-weighted kernel, that have the property of being injective for suitable choices of the kernel hyperparameters.

In the rest of this paper we discuss the properties of planar languages. After defining notation, we present simple examples of planar languages, as well as examples of languages that are not planar languages. We then discuss the closure properties of the class of planar languages, their relation to other well-studied classes, learnability properties, and conclude with some directions for future research. Our focus in this paper is on the learnability and language-theoretic issues, so we won't discuss the algorithmic/computational issues involved. We have implemented and tested thoroughly all of the results presented here, see [CCFWS06] for discussion.

2 Planar Languages

We will use the following definitions and notation. We have a finite non-empty set Σ, which we call the vocabulary. We consider the free monoid Σ^* with identity/empty string ε. A language L is a subset of Σ^*. We will write $a, b \ldots$ for elements of Σ^*. We write $|u|$ for the length of a string u, and we will write $|u|_v$ for the number of occurrences of v as a substring of u. We will also write $u[i]$ for the ith character of u, where $1 \leq i \leq |u|$. Sequences of indices are written $\mathbf{i}$, and $s(\mathbf{i})$ will denote the non-contiguous substring of s composed of the characters at the positions in s specified by $\mathbf{i}$.

We will consider feature maps ϕ from Σ^* into a possibly infinite dimensional Hilbert space H. For the purposes of this paper we can consider these to be real vector spaces, with the standard inner product which we will write as $\langle u, v \rangle$. When the dimension of the Hilbert space is very high it can be prohibitively expensive or impossible to calculate $\phi(x)$ directly. Accordingly for every ϕ we can define a kernel function from $\Sigma^* \times \Sigma^* \to R$, which is defined as $\kappa(u, v) = \langle \phi(u), \phi(v) \rangle$. Thus such feature spaces can still be used with many machine learning algorithms, since these are often based on the distance between points in a feature space. This approach is sometimes called the *kernel trick* and was first used in [ABR64], in the context of linear separators.

For the algorithms we discuss here it will be possible to compute $\kappa(u, v)$ in time polynomial in $|u|$ and $|v|$. Details of the dynamic programming techniques that make this possible can be found in [STC04]. It is possible to perform all of the calculations described here using only the kernel computations rather than working directly with the images of the strings in the Hilbert space.

2.1 Planarity

Given a particular kernel κ and associated feature map $\phi : \Sigma^* \to H$ we can give the following definition.

Definition 1. *A language $L \subseteq \Sigma^*$ is κ-planar if there is a finite set of strings $\{u_1, \ldots, u_n\}$ such that $L = \{w \in \Sigma^* \mid \exists \alpha_1 \ldots \alpha_i \in \mathbb{R} : \sum_i^n \alpha_i = 1 \wedge \sum_i^n \alpha_i \phi(u_i) = \phi(w)\}$.*

Given a κ-planar language L we can define the rank of L to be the cardinality of the smallest subset that defines the language. Note that this definition restricts planar languages to those of finite rank, and secondly that we restrict ourselves to affine combinations of strings u_i. This has the effect of slightly increasing the expressive power of the formalism, so that for example the languages do not necessarily contain the preimage of the origin, where it exists.

Definition 2. *For a language $L \subseteq \Sigma^*$, we define*

$$\mathrm{H}(L) = \{h \in H \mid \exists n > 0, w_1 \ldots w_n \in L, \alpha_1, \ldots \alpha_n \text{ such that } \sum_i \alpha_i = 1, \sum_i \alpha_i \phi(w_i) = h\}$$

where H is the feature space. Thus $\mathrm{H}(L)$ is the smallest hyperplane that contains the image of L.

For any finite set of strings $U = \{u_1, \ldots, u_n\}$ and a test string v, and any polynomially evaluable kernel κ with associated feature map ϕ, there is an algorithm for deciding whether $\phi(w) \in \mathrm{H}(u_1, \ldots, u_n)$, which runs in time polynomial in $n, |v|$ and $\sum_i |u_i|$. This algorithm, based on standard techniques, described in [STC04], involves computations only using the matrix of kernel values, and proceeds by normalising this matrix, which has the effect of translating the data in feature space so that the mean of the data lies in the origin, and performing an eigendecomposition of this matrix; it is then easy to project the test point v onto the perpendicular vectors to compute the distance in feature space of the point from the plane formed by U. Neglecting the issue of the numerical stability of these algorithms, which is beyond the scope of this paper, this distance will be zero if and only if $\phi(v)$ lies in the plane $H(L)$.

2.2 Kernels

We will now define the various kernels used in this paper. We will use the convention [STC04] that $\mathbf{i} = (i_1, \ldots, i_{|u|})$ ranges over the set of all strictly ordered tuples of indices. The following kernels are standard kernels in the literature.

Definition 3. *The* **All-k-Subsequences** *kernel.*
The feature space associated with the **All-k-Subsequences** *kernel is indexed by* $I = \cup_{i=0}^{k} \Sigma^i$*, with the embedding given by* $\phi_u(s) = |\{\mathbf{i} : u = s(\mathbf{i})\}|, u \in I$.

The Parikh kernel is simply the special case of **All-k-Subsequences** with $k = 1$.

Definition 4. *The* **p-Spectrum** *kernel.*
The feature space associated with the **p-Spectrum** *kernel is indexed by* $I = \Sigma^p$*, with the embedding given by* $\phi_u^p(s) = |\{(v_1, v_2) : s = v_1 u v_2\}|, u \in \Sigma^p$*. The associated kernel is defined as* $\kappa_p(s,t) = \langle \phi^p(s), \phi^p(t) \rangle = \Sigma_{u \in \Sigma^p} \phi_u^p(s) \phi_u^p(t)$.

We now define the schema kernel, which has not been discussed anywhere else before. We do not use it directly but just as a means to prove the injectivity of another kernel.

Definition 5. *Let* $\Sigma' = \Sigma \cup \{?\}$*. A* schema *is any sequence* $\sigma \in \Sigma'^{+}$*. A* gapped *schema is an element of* $\Sigma \times \{?\}^* \times \Sigma$*, or an element of* Σ.

Given a string u and a schema σ the count of the schema in a string is the number of times it matches exactly, where ? matches any symbol. More formally: $|\{i | \forall j = 1, \ldots |\sigma|, \sigma[i] =? \text{ or } \sigma[i] = u[i+j]\}|$.

Definition 6. *The* **gapSchemas** *kernel.*
*The gapped schema feature map (***gapSchemas***) maps strings to vectors, where each feature is indexed by a gapped schema, and the value of the feature is the count of the schema in the string. The gapped schema kernel is the kernel based on the gapped schema feature map.*

Example 7. The feature vector obtained from $\phi(abba)$ assigns 2 to features a and b, 1 to features $ab, ba, bb, a?b, b?a, a??a$, and 0 to all others.

Proposition 8. *The* **gapSchemas** *kernel is injective.*

Proof. By construction of the inverse function. Given an feature representation $h \in H$, and a $w \in \Sigma^*$ such that $\phi(w) = h$. Let l be the length of the longest schema with a non-zero value in h, which will be unique. Clearly $|w| = l$.

1. The first character of the string is the first character of the longest schema.
2. The nth character (for $n > 1$) contributes as first character to one of the schemas of length $l - n + 1$, but not to the schemas of length $l - n + 2$. So, let S_n be the multiset enumerating just all first characters of schemas of $l - n + 1$. Then the nth character is the only element of the set $S_n - S_{n-1}$.

□

Definition 9. *The gap-weighted subsequences feature map (***gapWeighted***): All non-contiguous subsequences of length p where each occurrence is weighted exponentially by the number of gaps. This is adjusted by a hyperparameter λ. $\phi_u(s) = \sum_{\mathbf{i}:u=s(\mathbf{i})} \lambda^{l(\mathbf{i})}$ where $|u| = p$. $l(i)$ is defined as $1 + i_{|i|} - i_1$.*

The gap-weighted subsequences feature kernel is the kernel based on the gap-weighted subsequences feature map.

Definition 10. *The gap-weighted subsequences plus feature map (***gapWeighted+***): Combines* **gapWeighted** *and the Parikh kernel times λ.*[1] *We also define the associated kernel.*

Example 11. For $p = 2$:

$$\phi(bab) = \begin{pmatrix} a: \lambda \\ b: 2\lambda \\ aa: 0 \\ ab: \lambda^2 \\ ba: \lambda^2 \\ bb: \lambda^3 \end{pmatrix}$$

2.3 Injectivity of the gapWeighted+ Kernel

The proof of the following proposition makes use of the existence of transcendental numbers. A transcendental number is any complex number that is not algebraic, that is, not the solution of a non-zero polynomial equation with integer (or, equivalently, rational) coefficients; some standard examples are π and e. The intuition behind the proof is that there is a correspondence between gapped schemas and the complex polynomials obtained from the **gapWeighted+** kernel.

[1] We could use the standard Parikh kernel here, the λ factor is purely for technical reasons.

Proposition 12. *There exists an injective function f from* **gapSchemas** *feature representations to* **gapWeighted+** *feature representations, for $p = 2$.*

Proof. We write H_1 for the space of **gapSchemas** representations and ϕ_1 for the associated feature map, and H_2, ϕ_2 for the space of **gapWeighted+** feature representations, and its map. We define f as follows: $f_a(h) = \lambda h_a$, $f_{ab}(h) = h_{ab}\lambda^2 + h_{a?b}\lambda^3 + \dots$.

By construction we can see that $\phi_2(w) = f(\phi_1(w))$. Suppose we have two elements h, h' of H_1, that lie in $\phi_1(\Sigma^*)$, such that $f(h) = f(h')$. This means that for every feature ab, we have $f_{ab}(h) = f_{ab}(h')$ and therefore $h_{ab}\lambda^2 + h_{a?b}\lambda^3 + \dots = h'_{ab}\lambda^2 + h'_{a?b}\lambda^3 + \dots$. This means that λ is the root of the polynomial $(h_{ab} - h'_{ab})\lambda^2 + (h_{a?b} - h'_{a?b})\lambda^3 + \dots = 0$. Note that this is a polynomial since we will only have finitely many non-zero feature values for images of strings. Since λ is transcendental this means that all of the coefficients must be zero, i.e. $h_{ab} = h'_{ab}, h_{a?b} = h'_{a?b}, \dots$. Therefore $h = h'$ and the map is injective. □

An algorithm implementing f^{-1} exists: from the **gapWeighted+** feature representation the length of the string, l, and thus of the longest schema, can be calculated. Consider the total sum of all the values from the **gapWeighted+** feature representation. We can simply enumerate all possible values for this sum; λ^2 for $l = 2$, $2\lambda^2+\lambda^3$ for $l = 3$, $3\lambda^2+2\lambda^3+\lambda^4$ for $l = 4$ etc, until we find one equal or very close to the total sum (note this is an increasing sequence, so termination is guaranteed).

Given the value of l and λ, the value v_i of every coordinate i of the **gapWeighted+** feature representation can be matched against an exhaustive list of all polynomials possible for a single coordinate: $c_1\lambda+c_2\lambda^2+c_3\lambda^3+\dots+c_l\lambda^l$, where $0 \leq c_2 \leq \min(\text{floor}(v_i/\lambda), l-1), 0 \leq c_3 \leq \min(\text{floor}(v_i/\lambda), l-2), \dots, 0 \leq c_l \leq \min(\text{floor}(v_i/\lambda), 1)$ (note that the length of this list is bounded by l). Each of these polynomials corresponds to one unique combination of schemas, and from the **gapSchemas** feature representation the string can easily be generated.

Proposition 13. *The* **gapWeighted+** *kernel is injective for any transcendental value for λ and $p = 2$.*

Proof. Since there exists an injective function f from **gapSchemas** feature representations to **gapWeighted+** feature representations, and since the composition of two injective functions is also injective, the **gapWeighted+** kernel is injective. □

Note that the choice of value for λ is crucial, and that when $\lambda \to 0$, for example, the **gapWeighted+** kernel is not injective. In this case the kernel behaves like the ***p*-Spectrum** kernel, which is not injective: for $p = 2$, *aaacaaa* and *aacaaaa* have the same image.

Note that the inclusion of the Parikh kernel in **gapWeighted+** is necessary only for dealing with languages containing strings of length 1. If we restrict the domain to a class of languages whose shortest strings are of length at least $l, l > 1$, then **gapWeighted** (and thus **gapWeighted+**) is injective for this domain for any $p \geq 2$.

3 Generative Capacity

Clearly the generative capacity of a given class of planar languages depends crucially on the kernel used. Consider the kernel $\kappa_L(u, v) = 1$ if $u, v \in L$ and 0 otherwise, for any language L: using this trivial kernel L is κ_L-planar. Similarly, planes in the feature space defined by the discrete kernel $\kappa(u, v) = 1$ if $u = v$ and 0 otherwise, are the images of the finite languages, where the rank is the cardinality of the language.

The class of **gapWeighted** planar languages contains quite expressive languages. It contains the copy language with disjoint alphabet: given an alphabet Σ split into two disjoint sets Σ_1, Σ_2, with a bijection f between them, it is defined as $\{uv | u \in \Sigma_1^*, v \in \Sigma_2^*, v[i] = f(u[i])\}$. For example, with $\Sigma = \{a, b\}$ and $\Sigma' = \{c, d\}$, $f(a) = c, f(b) = d$, *abacdc* is grammatical. Informally, the plane corresponding to this language (given this kernel) is simply defined by $|s|_{ba} = 0$ (for all $b \in \Sigma 2$, $a \in \Sigma_1$), $|s|_a = |s|_c, |s|_{ab} = |s|_{cd}$ for all $a, b \in \Sigma_1$, $c, d \in \Sigma_2$. This language is a direct model of the Swiss German crossing dependencies construction mentioned in the introduction, the u substring represents a list of noun phrases, the v part a list of verb phrases.

The language $a^n b^m c^n d^m$ is planar for **All-k-Subsequences**, $k = 2$. It can be expressed in linear constraints: $|u|_a = |u|_c$, $|u|_b = |u|_d$, $|u|_{ba} = |u|_{ca} \ldots = 0$. It's easy to see that the MIX language (all permutations of $a^n b^n c^n$ for all n) and dependent branches language ($a^n b^m c^m d^l e^l$, $n = m + l > 0$) can be expressed in similar fashion.

Some languages that can be easily expressed with linear constraints can be expressed in more conventional formalisms only with very large grammars. For example, [Asv06] shows that the size of context-free grammars in Chomksy normal form that generate finite languages containing all permutations of n different symbols grow by a function exponential in n.

3.1 Planar Languages and the Chomsky Hierarchy

Planar languages *cross-cut* the Chomsky Hierarchy. Exactly what subset of each degree of the hierarchy they include depends on the kernel used.

Example 14. The class of languages $\mathcal{L}_{All2}$, all languages planar in the feature space of the
All-k-Subsequences kernel with $k = 2$, contains:

1. Finite languages: both $\{a\}, \{a, ab\}$ are in L_{All2}, but $\{abba\}$ isn't. The string *abba* maps to a point in feature space that *baab* maps to as well, so $\{abba, baab\}$ *is* in $\mathcal{L}_{All2}$.
2. Regular languages: $\{a^*\}$ is in $\mathcal{L}_{All2}$, but not **Even**, the language containing all and only strings of even length.
3. Context-free languages: $a^n b^n$ is in $\mathcal{L}_{All2}$, but not the bracket language.
4. Mildly context-sensitive languages: $a^n b^n c^n$.
5. Non-mildly context-sensitive languages: $a^n b^n c^n d^n e^n$, but not a^{n^2}.

3.2 Relationship with k-testable Languages

The class of k-testable languages has been shown to be learnable, and have been applied in bioinformatics, see [YK98].

If κ_p is the p-**Spectrum** kernel, then the k-testable languages are κ_p-planar, which is obvious from the definitions. However, the converse does not hold, the language with equal numbers of as and bs is κ_1 planar but not locally testable.

4 Closure Properties of Planar Languages

Planar languages do not enjoy most of the well-known closure properties. It is easy to see that for example the class of Parikh-planar languages is not closed under concatenation: both a^* and b^* are Parikh-planar, but $\{a^*b^*\}$ isn't. Similarly they are not closed under union, or homomorphism. Two exceptions are the obvious property of being closed under reversal, and under intersection:

Proposition 15. *The intersection of two κ-planar languages is κ-planar.*

Proof. Suppose that L_1 and L_2 are κ-planar for some kernel κ. Consider $H(L_1 \cap L_2)$, the least plane that contains the image of the intersection of L_1 and L_2 (Recall Definition 2).

Suppose $w \in L_1$ and $w \in L_2$. Clearly $\phi(w) \in H(L_1 \cap L_2)$. Conversely suppose we have a w such that $\phi(w) \in H(L_1 \cap L_2)$. Now $H(L_1 \cap L_2) \subset H(L_1)$ so $\phi(w) \in H(L_1)$. Since L_1 is planar, $w \in L_1$. Similarly $w \in L_2$. So, $\phi(w) \in H(L_1 \cap L_2)$ if and only if $w \in L_1 \cap L_2$. Since L_1 and L_2 are planar, $H(L_1)$ has finite dimension, and thus so does $H(L_1 \cap L_2)$. Therefore $L_1 \cap L_2$ is planar. □

Note also that the resulting hyperplane will be of lower dimensionality than the intersecting planes (and will have the same dimensionality only if these are identical).

5 Learnability

As was previously mentioned, the main motivation for studying κ-planar languages is their inherent learnability. There exists a simple and efficient algorithm to learn any of them from positive data: just use as the representation all linear combinations of the sample data, i.e. find the smallest hyperplane that contains all data points. We define a slightly modified algorithm SPAN(σ) that checks for any new data-point whether it is generated by the current hypothesis (this can easily be determined, since distance from the plane can be computed in polynomial time), and only changes its conjecture when necessary. The new conjecture will have the new data-point added to the base description. Algorithm 1 presents this more formally:

At any given point we have a set of strings that form a basis in feature space for the plane. These strings will be linearly independent: $|U|$ will be equal to the rank of $H(U)$. For any language L such that $H(L)$ has finite rank r, we can see

Algorithm 1. SPAN learning algorithm

```
Inputs: kernel κ, training data S = {w_1, ... w_l}
U = {}
for i = 1 to i = l do
  if φ(w_i) ∉ H(U) then
    U = U ∪ {w_i}
  end if
end for
```

that SPAN, will converge to a representation of the preimage of $H(L)$ after at most r mind changes.

The notion of learnability we will apply here is known as identification in the limit ([Gol67]). Within this framework, a class of languages is considered learnable if there exists a (computable) function over sequences of input data that converges on a correct hypothesis after a finite amount of data (assuming all data is presented eventually).

5.1 Resource-Based Constraints on Learners

Identification in the limit of a class guarantees the existence of learning algorithms for that class, but the learning problem is not necessarily tractable. Since we are interested in applications we need to specify further constraints on the learner. Ideally we would use some notion of polynomial identification in the limit. There are several around ([dlH95, Yok91] among others), but they are all of a somewhat ad-hoc nature. The latter is one of the more restrictive ones and thus suits our present purpose best. [Yok91] defines a class as *polynomial-time identifiable in the limit from positive data* if there exists a learning algorithm for that class such that both the number of explicit errors of prediction and the computation time it needs for any sequence of data are bounded by polynomials over the complexity of the representation (rank, in this case).

5.2 Behavioural Constraints on Learners

Identification in the limit provides criteria for the success of a learning process, but grants total freedom to learners prior to convergence. Generally speaking it's desirable to be able to impose additional constraints, to guarantee 'rational' behaviour of the learner. Formally this simply means choosing a subset of possible learners.

Many different constraints have been studied in the literature, we define the ones relevant to this discussion:

Definition 16. ***Consistent learning***
A learning function φ is consistent *on $\mathcal{G}$ if for any $L \in \mathrm{L}(\mathcal{G})$ and for any finite sequence $\langle s_0, \ldots, s_i \rangle$ of elements of L, either $\varphi(\langle s_0, \ldots, s_i \rangle)$ is undefined or $\{s_0, \ldots, s_i\} \subseteq \mathrm{L}(\varphi(\langle s_0, \ldots, s_i \rangle))$.*

Definition 17. ***(Strong) Monotonicity***
The learning function φ is monotone increasing *or* strong monotonic *if for all finite sequences $\langle s_0, \ldots, s_n \rangle$ and $\langle s_0, \ldots, s_{n+m} \rangle$, whenever $\varphi(\langle s_0, \ldots, s_n \rangle)$ and $\varphi(\langle s_0, \ldots, s_{n+m} \rangle)$ are defined,* $\mathrm{L}(\varphi(\langle s_0, \ldots, s_n \rangle)) \subseteq \mathrm{L}(\varphi(\langle s_0, \ldots, s_{n+m} \rangle))$.

Definition 18. ***Incrementality***
The learning function φ is incremental *if there exists a computable function ψ such that*
$\varphi(\langle s_0, \ldots, s_{n+1} \rangle) \simeq \psi(\varphi(\langle s_0, \ldots, s_n \rangle), s_{n+1})$.

5.3 Learnability of Planar Languages

We are now in a position to demonstrate a strong learnability result for planar languages. Even under a combination of various constraints on the use of resources and on behaviour this class is learnable:

Theorem 19. *The class of κ-planar languages is identifiable in the limit, with polynomial size characteristic set, a polynomial number of mind changes, and polynomial computation, by a learner that is simultaneously consistent, monotone increasing and incremental.*

Proof. For any plane of rank r there is a set C of strings such that $|C| = r$ and for any enumeration e of C, $|\mathrm{SPAN}(e)| = r$ (trivial). It follows immediately that for any enumeration e of C, $\mathrm{SPAN}(e)$ defines a plane for L, and thus that C is a characteristic set for L.

Consider $C' = C \cup S$, where S is a finite subset of L. By definition of planar language and SPAN, S is in the language defined by plane $\mathrm{SPAN}(e)$, so for any enumeration e' of C', $\mathrm{SPAN}(e') = \mathrm{SPAN}(e)$, both define a plane for L.

Therefore the learning function based on SPAN will, after encountering all elements from a characteristic set C, hypothesise L and will not diverge from this hypothesis. This proves identifiability in the limit of the class of planar languages.

The characteristic set has a size[2] polynomial (in fact linear) in the rank of the target plane, the learner need only change its mind as many times as the size of the characteristic set, and SPAN can be implemented to run in polynomial time.

A learner for planar languages based on SPAN is consistent and monotone increasing by definition. Incrementality is trivial: the plane is defined in terms of the set of basis strings; this basis and the new data-point are enough to generate a new hypothesis. □

5.4 Finite Elasticity

Learnability, and related properties like existence of a mind change bound, are largely determined by topological properties of the class under consideration.

[2] Here 'size' simply means cardinality of the set. It makes little sense to include the length of the strings in this definition, since the class includes for example the finite language with one element $a^{10^{100}}$.

One such property is the existence of an *infinite ascending chain* of languages. This means that for $L_0, L_1, \ldots, L_n, \ldots$ in that class, $L_0 \subset L_1 \subset \ldots \subset L_n \subset \ldots$. This implies the weaker property known as *infinite elasticity*:

Definition 20. ***(In)finite elasticity****[Wri89, MSW91]*
A class $\mathcal{L}$ of languages is said to have infinite elasticity *if there exists an infinite sequence $\langle s_n \rangle_{n \in \mathbb{N}}$ of sentences and an infinite sequence $\langle L_n \rangle_{n \in \mathbb{N}}$ of languages in $\mathcal{L}$ such that for all $n \in \mathbb{N}$, $s_n \notin L_n$, and $\{s_0, \ldots, s_n\} \subseteq L_{n+1}$.*

A class $\mathcal{L}$ of languages is said to have finite elasticity *if it does not have infinite elasticity.*

Finite elasticity is a sufficient condition for learnability under two conditions, as shown in [Wri89]:

Theorem 21. ***(Wright)*** *Let $\mathcal{G}$ be a class of grammars for a class of recursive languages, where $G \in \mathcal{G}$ is at least semi-decidable. If $\mathrm{L}(\mathcal{G})$ has finite elasticity, then $\mathcal{G}$ is identifiable in the limit.*

Thus one route to proving learnability of a class is demonstrating it has finite elasticity, which has the added benefit of allowing one to easily define a conservative learning algorithm. In this context this is not necessary, however we do get nice closure properties for free.

Proposition 22. *If the feature space defined by κ has finite dimension, then the class of κ-planar languages has finite elasticity.*

Proof. Suppose the class has infinite elasticity, with strings $s_1, \ldots$ and languages $L_1 \ldots$. If we define $S_n = H(\{s_0, \ldots, s_n\})$, then $S_{n-1} \subseteq L_n$, and $S_n \not\subseteq L_n$. Obviously, $S_0 \subset S_1 \ldots$, which constitutes an infinite ascending chain. Since S_n must be of greater rank than S_{n-1}, and every S_n is included in a language in the class, there can't be any bound on the rank of the planes for languages in the class. This is in contradiction with the hypothesis that the feature space has finite dimension. □

This does not hold for all planar languages. Any class of planar languages that contains all finite languages (c.f. the kernel based on all substrings) has an infinite ascending chain. It *does* hold for **gapWeighted-** and **gapWeighted+** planar languages.

It is straightforward to establish the following corollary (see [Wri89]):

Corollary 23. *Any finite union of classes of κ-planar languages where κ is finite dimensional has finite elasticity.*

Thus planar languages can be generalised to larger classes of learnable languages. Unfortunately, the naive learning algorithms for these classes will run in exponential time.

6 Conclusion

There is very little work to which the current approach can be compared. [Sal05] presents some steps towards defining languages using constraints on subsequence counts. [Kon04] discusses an approach to embedding languages in a feature space, but using different techniques, and only modelling the locally testable languages.

Our approach has a number of limitations. First, while we can learn a number of simple languages, it is not clear that this approach will scale to learning much more complex languages. To get learnability for large scale problems, we will need to combine these techniques with other, perhaps more traditional, grammatical inference algorithms. Secondly, the size of the representations can be quite large, particularly if we use kernels that involve longer substrings; in the worst case this size can be $|\Sigma|^p$ where p is the length of the subsequences represented in the kernel. Finally, though the algorithms are polynomial, their naive application has a computational cost that is cubic in the number of examples. Thus it will be difficult to solve problems with more than several thousand examples on current workstations, without careful optimisations.

We have introduced the class of planar languages, an inherently learnable class of languages with high expressive power, defined in term of string kernels. This is the first application of this family of techniques to the problems of grammatical inference. This class contains interesting context-sensitive languages, including some classic examples from computational linguistics. We have carried out extensive experimental verification of the approach described here [CCFWS06] and have confirmed the practical efficiency of these techniques. These classes are defined in terms of planes in a feature space, which can be efficiently learned with standard machine learning techniques from positive data alone. The choice of kernel determines the expressive power and closure properties of the resulting class. Two kernels have been shown to be injective (depending on the choice of hyperparameters), a particular important property in the context of GI. We have also shown that for some kernels, planar languages have the desirable property of finite elasticity. This allows easy extension to richer classes of languages whilst retaining good learnability properties.

Planar languages are a novel approach to GI, and it seems we have only scratched the surface. An interesting topic for future research would be language classes defined using linear inequalities, which would correspond to half-spaces in the feature space. Such classes would allow the expression of natural language phenomena such as Chinese number words. Another direction would be the definition of new string kernels for specific purposes. The kernels we have considered are in most cases standard, well-studied kernels, so it is likely that new ones can be designed that are better suited for grammatical inference.

Acknowledgements

This work has benefitted from the support of the EU funded PASCAL Network of Excellence on Pattern Analysis, Statistical Modelling and Computational Learning.

References

[ABR64] M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer. Theoretical foundations of the potential function method in pattern recognition. *Automation and Remote Control*, 25:821–837, 1964.

[Asv06] Peter R. J. Asveld. Generating all permutations by context-free grammars in Chomsky normal form. *Theoretical Computer Science (TCS)*, 354(1):118–130, 2006.

[CCFWS06] Alexander Clark, Christophe Costa Florêncio and Chris Watkins. Languages as hyperplanes: grammatical inference with string kernels. In *ECML, 17th European Conference on Machine Learning*. Springer-Verlag, 2006.

[dlH95] Colin de la Higuera. Characteristic sets for polynomial grammatical inference. In *Proceedings of the International Colloquium on Grammatical Inference ICGI-96*. Springer-Verlag, 1995.

[Gol67] E. Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[Huy84] Riny Huybregts. The weak inadequacy of context-free phrase structure grammars. In Ger J. de Haan, Mieke Trommelen, and Wim Zonneveld, editors, *Van Periferie naar Kern*. Foris, Dordrecht, 1984.

[JS96] Aravind K. Joshi and Yves Schabes. Tree-adjoining grammars. In Grzegorz Rosenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69–123. Springer-Verlag, New York, 1996.

[Kon04] Leonid Kontorovich. Learning linearly separable languages. Technical Report CMU-CALD-04-105, School of Computer Science, CMU, 2004.

[MSW91] Tatsuya Motoki, Takeshi Shinohara, and Keith Wright. The correct definition of finite elasticity: Corrigendum to identification of unions. In *The Fourth Workshop on Computational Learning Theory*. San Mateo, Calif.: Morgan Kaufmann, 1991.

[Sal05] Arto Salomaa. On languages defined by numerical parameters. Technical Report 663, Turku Centre for Computer Science, 2005.

[Shi85] Stuart M. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343, 1985.

[STC04] John Shawe-Taylor and Nello Christianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[Wat99] Chris Watkins. Dynamic alignment kernels. Technical Report CSD-TR-98-11, Department of Computer Science, Royal Holloway College, University of London, 1999.

[Wri89] Keith Wright. Identification of unions of languages drawn from an identifiable class. In *The 1989 Workshop on Computational Learning Theory*, pages 328–333. San Mateo, Calif.: Morgan Kaufmann, 1989.

[YK98] Takashi Yokomori and Satoshi Kobayashi. Learning local languages and their application to DNA sequence analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(10):1067–1079, 1998.

[Yok91] Takashi Yokomori. Polynomial-time learning of very simple grammars from positive data. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 213–227, University of California, Santa Cruz, 5–7 August 1991. ACM Press.