# Languages as Hyperplanes

## Grammatical Inference with String Kernels

Alex Clark Christophe Costa Florencio Chris Watkins

Department of Computer Science
Royal Holloway, University of London

European Conference on Machine Learning, 2006

# Acknowledgement of Support

We would like to acknowledge support from the Pascal Network of Excellence,

in the form of a ‘pump-priming’ grant 2005-2006 for

*Grammatical Inference with String Kernels*

# Acknowledgement of Support

We would like to acknowledge support from the Pascal Network of Excellence,
in the form of a ‘pump-priming’ grant 2005-2006 for
*Grammatical Inference with String Kernels*

# Original Motivation

## Language Learning from Positive Examples

Learning a language from positive examples only

- Children learn complex grammars
  - without being given negative examples
  - without being corrected
- Classic problem in computational linguistics
- Useful formulation in practice because informative negative examples difficult to obtain/generate

Learnability criterion:

- Learner is presented with a sequence of sentences from a language
- After some (preferably small) number of sentences, learner acquires exact description of language

# Formalising Language Learning

- Finite alphabet $\Sigma$
- Set of finite sequences $\Sigma^*$
- A *language* $L$ is a set of finite sequences, $L \subseteq \Sigma^*$
- Problem: learn to recognise sequences in $L$ from presentation of positive examples only.

After finite number of examples, learner should acquire exact description of language.

PAC-learning type criteria also possible...

# Formalising Language Learning

- Finite alphabet $\Sigma$
- Set of finite sequences $\Sigma^*$
- A *language* $L$ is a set of finite sequences, $L \subseteq \Sigma^*$
- Problem: learn to recognise sequences in $L$ from presentation of positive examples only.

After finite number of examples, learner should acquire exact description of language.

PAC-learning type criteria also possible...

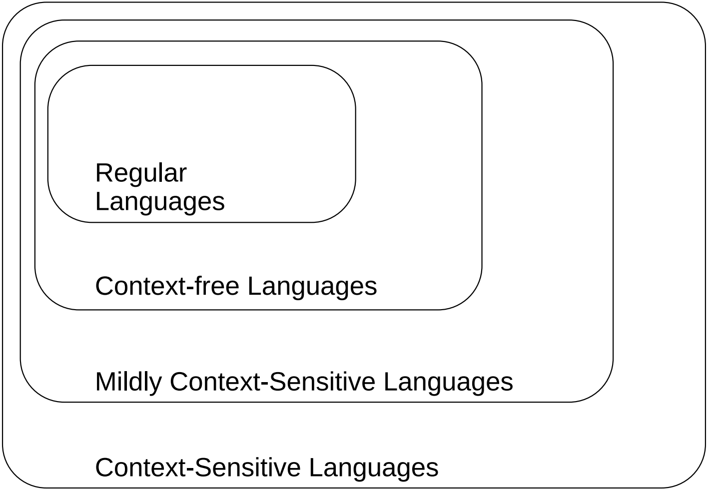
# Formalising Language Learning

- Finite alphabet $\Sigma$
- Set of finite sequences $\Sigma^*$
- A *language* $L$ is a set of finite sequences, $L \subseteq \Sigma^*$
- Problem: learn to recognise sequences in $L$ from presentation of positive examples only.

After finite number of examples, learner should acquire exact description of language.

PAC-learning type criteria also possible...

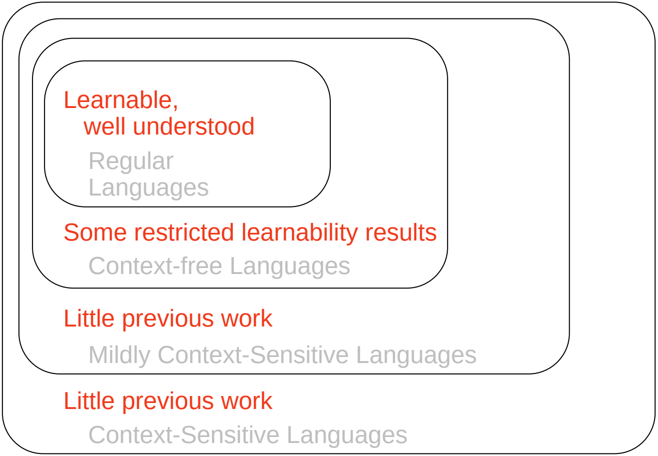
# Representing Languages: the Chomsky Hierarchy




Grammars of increasing intractability

Natural language believed to be mostly context-free, but some mildly context-sensitive phenomena.

# Grammar Induction in Computational Linguistics




But Chomsky hierarchy does not match intuitive notions of complexity...
Some intuitively simple languages are MCS or CS: how to learn them?

# Planar Languages: a New Approach

Map strings to points in a Euclidean feature space $\phi : \Sigma^* \mapsto H$

To define a language, specify a region $U$ in feature space

Language is all strings $s$ such that $\phi(s) \in U$

Language is *pre-image* of $U$ in $\Sigma^*$

So, two questions:
What feature spaces?

What type of region?

# Planar Languages: a New Approach

Map strings to points in a Euclidean feature space $\phi : \Sigma^* \mapsto H$

To define a language, specify a region $U$ in feature space

Language is all strings $s$ such that $\phi(s) \in U$

Language is *pre-image* of $U$ in $\Sigma^*$

So, two questions:
What feature spaces?

What type of region?

# Planar Languages: a New Approach

Map strings to points in a Euclidean feature space $\phi : \Sigma^* \mapsto H$

To define a language, specify a region $U$ in feature space

Language is all strings $s$ such that $\phi(s) \in U$

Language is *pre-image* of $U$ in $\Sigma^*$

So, two questions:
What feature spaces?

What type of region?

# What Type of Region?

Hmm....could use half-spaces? Conventional... Spheres? Manifolds? Tricky... All of these are possible...But

Hyperplanes seem natural
Finite-dimensional hyperplanes are particularly nice...let's see why

# What Type of Region?

Hmm....could use half-spaces? Conventional... Spheres? Manifolds? Tricky... All of these are possible...But

Hyperplanes seem natural
Finite-dimensional hyperplanes are particularly nice...let's see why

# What Type of Region?

Hmm....could use half-spaces? Conventional... Spheres?
Manifolds? Tricky... All of these are possible...But

Hyperplanes seem natural
Finite-dimensional hyperplanes are particularly nice...let's see why

# What Type of Region?

Hmm....could use half-spaces? Conventional... Spheres? Manifolds? Tricky... All of these are possible...But

Hyperplanes seem natural
Finite-dimensional hyperplanes are particularly nice...let's see why

# What Type of Region?

Hmm....could use half-spaces? Conventional... Spheres? Manifolds? Tricky... All of these are possible...But

Hyperplanes seem natural
Finite-dimensional hyperplanes are particularly nice...let's see why

# What Type of Region?

Hmm....could use half-spaces? Conventional... Spheres? Manifolds? Tricky... All of these are possible...But

Hyperplanes seem natural
Finite-dimensional hyperplanes are particularly nice...let's see why

# What Type of Region?

Hmm....could use half-spaces? Conventional... Spheres? Manifolds? Tricky... All of these are possible...But

Hyperplanes seem natural
Finite-dimensional hyperplanes are particularly nice...let's see why

# What Type of Region?

Hmm....could use half-spaces? Conventional... Spheres? Manifolds? Tricky... All of these are possible...But

Hyperplanes seem natural

Finite-dimensional hyperplanes are particularly nice...let's see why

# What Type of Region?

Hmm....could use half-spaces? Conventional... Spheres? Manifolds? Tricky... All of these are possible...But

Hyperplanes seem natural
Finite-dimensional hyperplanes are particularly nice...let's see why

# Planar Languages

Definition: A language $L$ is *phi*-planar if

$$L = \{w : \phi(w) \in U \subseteq H\}$$

where $U$ is a $r$-dimensional hyperplane (not necessarily containing the origin)

Useful definition because:

- hyperplane defined by linear equality constraints in feature space, often easy to interpret
- hyperplane learned exactly when $r + 1$ linearly independent strings observed
- very simple learning algorithm

# Planar Languages

Definition: A language $L$ is *phi*-planar if

$$L = \{w : \phi(w) \in U \subseteq H\}$$

where $U$ is a $r$-dimensional hyperplane (not necessarily containing the origin)

Useful definition because:

- hyperplane defined by linear equality constraints in feature space, often easy to interpret
- hyperplane learned exactly when $r + 1$ linearly independent strings observed
- very simple learning algorithm

# Planar Languages

Definition: A language $L$ is *phi*-planar if

$$L = \{w : \phi(w) \in U \subseteq H\}$$

where $U$ is a $r$-dimensional hyperplane (not necessarily containing the origin)

Useful definition because:

- hyperplane defined by linear equality constraints in feature space, often easy to interpret
- hyperplane learned exactly when $r + 1$ linearly independent strings observed
- very simple learning algorithm

# Planar Languages

Definition: A language $L$ is *phi*-planar if

$$L = \{w : \phi(w) \in U \subseteq H\}$$

where $U$ is a $r$-dimensional hyperplane (not necessarily containing the origin)

Useful definition because:

- hyperplane defined by linear equality constraints in feature space, often easy to interpret
- hyperplane learned exactly when $r + 1$ linearly independent strings observed
- very simple learning algorithm

# Example: Strings with equal numbers of a's and b's

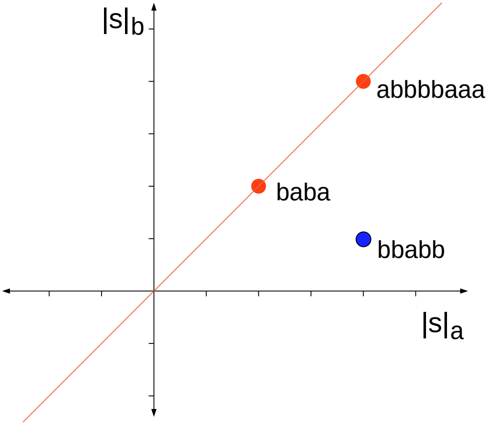
Let $\Sigma = \{a, b\}$

Consider $L = \{s \in \Sigma^* : |s|_a = |s|_b\}$
where $|s|_a$ is the number of $a$'s in $s$

i.e. $L$ consists of strings with equal numbers of $a$ and $b$

Consider feature space $H$ with two dimensions: $|s|_a$ and $|s|_b$

# Example: Strings with equal numbers of a's and b's




Hyperplane is a line

Language acquired once 2 different points on line found

# Learning Hyperplanes: KPCA of centred Gram Matrix

Algorithm (for kernel-defined feature space mapping):

- Form Gram matrix for positive data
- Centre Gram matrix (record the displacement necessary)
- Compute principal components of Gram matrix. Keep significant components.
- Hyperplane now defined!

# Convergence

Planar Target Language:

- Convergence is rapid and exact
- Key factor is rank $r$ of hyperplane
- At least $r$ independent examples required to learn language
- Incomplete learning produces *false negatives* in test set

Non-Planar Target Language:

- Convergence to lowest-dimensional hyperplane containing language which may be all of $\Sigma^*$
- Learning non-planar language produces *false positives* in test set

# Convergence

Planar Target Language:

- Convergence is rapid and exact
- Key factor is rank $r$ of hyperplane
- At least $r$ independent examples required to learn language
- Incomplete learning produces *false negatives* in test set

Non-Planar Target Language:

- Convergence to lowest-dimensional hyperplane containing language which may be all of $\Sigma^*$
- Learning non-planar language produces *false positives* in test set

# Convergence

Planar Target Language:

- Convergence is rapid and exact
- Key factor is rank $r$ of hyperplane
- At least $r$ independent examples required to learn language
- Incomplete learning produces *false negatives* in test set

Non-Planar Target Language:

- Convergence to lowest-dimensional hyperplane containing language which may be all of $\Sigma^*$
- Learning non-planar language produces *false positives* in test set

# Convergence

Planar Target Language:

- Convergence is rapid and exact
- Key factor is rank $r$ of hyperplane
- At least $r$ independent examples required to learn language
- Incomplete learning produces *false negatives* in test set

Non-Planar Target Language:

- Convergence to lowest-dimensional hyperplane containing language which may be all of $\Sigma^*$
- Learning non-planar language produces *false positives* in test set

# Convergence

Planar Target Language:

- Convergence is rapid and exact
- Key factor is rank $r$ of hyperplane
- At least $r$ independent examples required to learn language
- Incomplete learning produces *false negatives* in test set

Non-Planar Target Language:

- Convergence to lowest-dimensional hyperplane containing language which may be all of $\Sigma^*$
- Learning non-planar language produces *false positives* in test set

# Convergence

Planar Target Language:

- Convergence is rapid and exact
- Key factor is rank $r$ of hyperplane
- At least $r$ independent examples required to learn language
- Incomplete learning produces *false negatives* in test set

Non-Planar Target Language:

- Convergence to lowest-dimensional hyperplane containing language which may be all of $\Sigma^*$
- Learning non-planar language produces *false positives* in test set

# Convergence

Planar Target Language:

- Convergence is rapid and exact
- Key factor is rank $r$ of hyperplane
- At least $r$ independent examples required to learn language
- Incomplete learning produces *false negatives* in test set

Non-Planar Target Language:

- Convergence to lowest-dimensional hyperplane containing language which may be all of $\Sigma^*$
- Learning non-planar language produces *false positives* in test set

# Feature Space: Subsequence Counts

We consider two feature spaces (can be induced by string-kernels):

- Sub-sequence counts
- Gap-weighted sub-sequence counts

Example:
Let $s = ababab$
Count of $ab$ in $s$ is $|s|_{ab} = 6$
Gap-weighted count of $ab$ in $s$ with weighting factor $\frac{1}{2}$ is $3 + 2 \times \frac{1}{4} + 1 \times \frac{1}{64}$

# Feature Space: Subsequence Counts

We consider two feature spaces (can be induced by string-kernels):

- Sub-sequence counts
- Gap-weighted sub-sequence counts

Example:
Let $s = ababab$
Count of $ab$ in $s$ is $|s|_{ab} = 6$
Gap-weighted count of $ab$ in $s$ with weighting factor $\frac{1}{2}$ is
$3 + 2 \times \frac{1}{4} + 1 \times \frac{1}{64}$

# Feature Space: Subsequence Counts

We consider two feature spaces (can be induced by string-kernels):

- Sub-sequence counts
- Gap-weighted sub-sequence counts

Example:
Let $s = ababab$
Count of *ab* in *s* is $|s|_{ab} = 6$
Gap-weighted count of *ab* in *s* with weighting factor $\frac{1}{2}$ is $3 + 2 \times \frac{1}{4} + 1 \times \frac{1}{64}$

# Feature Space: Subsequence Counts

We consider two feature spaces (can be induced by string-kernels):

- Sub-sequence counts
- Gap-weighted sub-sequence counts

Example:
Let $s = ababab$
Count of $ab$ in $s$ is $|s|_{ab} = 6$
Gap-weighted count of $ab$ in $s$ with weighting factor $\frac{1}{2}$ is
$3 + 2 \times \frac{1}{4} + 1 \times \frac{1}{64}$

# Experiments: Setup

- A range of languages tried, mostly already defined in computational linguistics literature
- 500 positive examples given; tested on 500 positive and negative examples
- specially informative negative test examples generated where necessary

We use two feature spaces:

- Counts of sub-sequences of lengths one and two
- Gap-weighted counts of sub-sequences of lengths one and two

# Experiments: Setup

- A range of languages tried, mostly already defined in computational linguistics literature
- 500 positive examples given; tested on 500 positive and negative examples
- specially informative negative test examples generated where necessary

We use two feature spaces:

- Counts of sub-sequences of lengths one and two
- Gap-weighted counts of sub-sequences of lengths one and two

# Experiments: Even

| | | |
|---|---|---|
| Even (Regular) | Even number of symbols<br>Alphabet $\{a, b, c\}$ | abcb, ba, babacc, aaaa |
| Bracket (CF) | Balanced brackets<br>Alphabet $\{(,)\}$ | (), ()(), (()(())) |

| | PCFG | | HMM | | SUBS | | | GPWT | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | FP | FN | FP | FN | FP | FN | R | FP | FN | R |
| Even | 0 | 0 | 0 | 0 | 100 | 0 | 12 | 100 | 0 | 12 |
| Bracket | 0 | 0 | 3.4 | 1.3 | 10.8 | 0 | 3 | 10.8 | 0 | 5 |

# Experiments: Even

| | | |
|---|---|---|
| Even<br>(Regular) | Even number of symbols<br>Alphabet $\{a, b, c\}$ | abcb, ba,<br>babacc,<br>aaaa |
| Bracket<br>(CF) | Balanced brackets<br>Alphabet $\{(, )\}$ | (), ()(),<br>(()(())) |

| | PCFG | | HMM | | SUBS | | | GPWT | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | FP | FN | FP | FN | FP | FN | R | FP | FN | R |
| Even | 0 | 0 | 0 | 0 | 100 | 0 | 12 | 100 | 0 | 12 |
| Bracket | 0 | 0 | 3.4 | 1.3 | 10.8 | 0 | 3 | 10.8 | 0 | 5 |

# A Regular Language with Long-Range Dependency

| | | |
|---|---|---|
| GWLang (Regular) | Strings of form *uavbw*, with $u, v, w \in \{c, d, e, f, g, h\}^*$, with $\lvert v \rvert = 3$ | acccb, feaghfbec, gahefbdg |

| | PCFG | | HMM | | SUBS | | | GPWT | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | FP | FN | FP | FN | FP | FN | R | FP | FN | R |
| GWLang | - | - | 25.4 | 5 | 0 | 0 | 53 | 0 | 1 | 64 |

# Planar Languages not Learned by HMMs or PCFGs

A German Verb Construction (Mildly Context Sensitive)
Strings in two halves: first half from $\{a, b, c, d\}$, second from $\{e, f, g, h\}$
For each $a$ in first half, there must be a $e$ in the second half, etc, but in any order
Example strings: *abfe*, *abcdeghf*, *aabefe*

# Planar Languages not Learned by HMMs or PCFGs

Strings of the form:
$n$ characters from $\{a, b\}$,
followed by $n$ characters from $\{c, d\}$,
followed by $n$ characters from $\{e, f\}$
etc

*babcccfee*, *adf*, *aaaaccccefef*

Depending on the number of sections these languages are context-free, mildly context-sensitive, or context-sensitive. Easily defined by linear equality constraints.

# Languages with Long-Range Sequence Dependencies

For example:
Palindromes: *abcabbaabbacba*
Single Repeats: *abcaabca*, *abbababbab*
Some of these languages context-free; others mildly context-sensitive
These are approximately planar in the sense that, for a given length of subsequences in the feature space, only sequences up to a certain length are fully specified by the feature vector.
e.g. *abba* and *baab* have the same feature vector for subsequences of length up to 2
For subsequences of length 3, shortest pair of indistinguishable sequences is of length 7.

# Languages with Long-Range Sequence Dependencies

For example:
Palindromes: *abcabbaabbacba*
Single Repeats: *abcaabca*, *abbababbab*
Some of these languages context-free; others mildly context-sensitive
These are approximately planar in the sense that, for a given length of subsequences in the feature space, only sequences up to a certain length are fully specified by the feature vector.
e.g. *abba* and *baab* have the same feature vector for subsequences of length up to 2
For subsequences of length 3, shortest pair of indistinguishable sequences is of length 7.

# Languages with Long-Range Sequence Dependencies

For example:

Palindromes: *abcabbaabbacba*

Single Repeats: *abcaabca*, *abbababbab*

Some of these languages context-free; others mildly context-sensitive

These are approximately planar in the sense that, for a given length of subsequences in the feature space, only sequences up to a certain length are fully specified by the feature vector.

e.g. *abba* and *baab* have the same feature vector for subsequences of length up to 2

For subsequences of length 3, shortest pair of indistinguishable sequences is of length 7.

# Languages with Long-Range Sequence Dependencies

For example:
Palindromes: *abcabbaabbacba*
Single Repeats: *abcaabca*, *abbababbab*
Some of these languages context-free; others mildly context-sensitive
These are approximately planar in the sense that, for a given length of subsequences in the feature space, only sequences up to a certain length are fully specified by the feature vector.
e.g. *abba* and *baab* have the same feature vector for subsequences of length up to 2
For subsequences of length 3, shortest pair of indistinguishable sequences is of length 7.

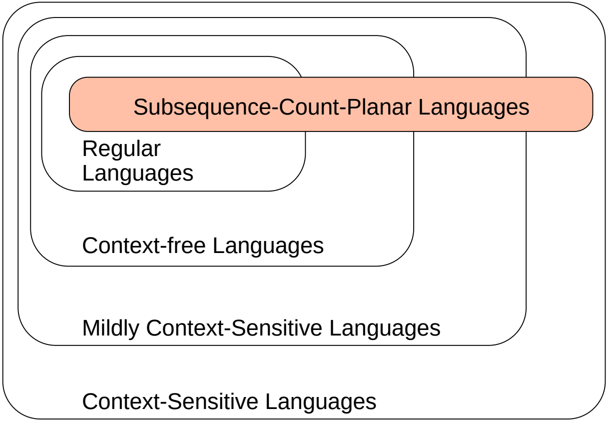
# Languages with Long-Range Sequence Dependencies

For example:
Palindromes: *abcabbaabbacba*
Single Repeats: *abcaabca*, *abbababbab*
Some of these languages context-free; others mildly context-sensitive
These are approximately planar in the sense that, for a given length of subsequences in the feature space, only sequences up to a certain length are fully specified by the feature vector.
e.g. *abba* and *baab* have the same feature vector for subsequences of length up to 2
For subsequences of length 3, shortest pair of indistinguishable sequences is of length 7.

# Languages with Long-Range Sequence Dependencies

For example:
Palindromes: *abcabbaabbacba*
Single Repeats: *abcaabca*, *abbababbab*
Some of these languages context-free; others mildly context-sensitive
These are approximately planar in the sense that, for a given length of subsequences in the feature space, only sequences up to a certain length are fully specified by the feature vector.
e.g. *abba* and *baab* have the same feature vector for subsequences of length up to 2
For subsequences of length 3, shortest pair of indistinguishable sequences is of length 7.

# Subsequence Planar Languages Cross-Cut Chomsky Hierarchy



# Summary

- Planar languages: a new(?) approach to language induction from positive examples
- Natural to represent some languages in terms of linear constraints on sub-sequences
- Certain types of long-range dependency of intervals, counts, and sequences can be
  - neatly represented
  - reliably learned
- For simple subsequence kernels, planar languages cross-cut Chomsky hierarchy
- Our example languages are intuitively simple
  Intuitively, earning them should be simple
  Our approach makes it simple
- Further directions
  - Characterise set of planar languages with respect to various kernels
  - String processing in features space...

# Summary

- Planar languages: a new(?) approach to language induction from positive examples
- Natural to represent some languages in terms of linear constraints on sub-sequences
- Certain types of long-range dependency of intervals, counts, and sequences can be
  - neatly represented
  - reliably learned
- For simple subsequence kernels, planar languages cross-cut Chomsky hierarchy
- Our example languages are intuitively simple
  Intuitively, earning them should be simple
  Our approach makes it simple
- Further directions
  - Characterise set of planar languages with respect to various kernels
  - String processing in features space...

# Summary

- Planar languages: a new(?) approach to language induction from positive examples
- Natural to represent some languages in terms of linear constraints on sub-sequences
- Certain types of long-range dependency of intervals, counts, and sequences can be
  - neatly represented
  - reliably learned
- For simple subsequence kernels, planar languages cross-cut Chomsky hierarchy
- Our example languages are intuitively simple
  Intuitively, earning them should be simple
  Our approach makes it simple
- Further directions
  - Characterise set of planar languages with respect to various kernels
  - String processing in features space...

# Summary

- Planar languages: a new(?) approach to language induction from positive examples
- Natural to represent some languages in terms of linear constraints on sub-sequences
- Certain types of long-range dependency of intervals, counts, and sequences can be
  - neatly represented
  - reliably learned
- For simple subsequence kernels, planar languages cross-cut Chomsky hierarchy
- Our example languages are intuitively simple
  Intuitively, earning them should be simple
  Our approach makes it simple
- Further directions
  - Characterise set of planar languages with respect to various kernels
  - String processing in features space...

# Summary

- Planar languages: a new(?) approach to language induction from positive examples
- Natural to represent some languages in terms of linear constraints on sub-sequences
- Certain types of long-range dependency of intervals, counts, and sequences can be
  - neatly represented
  - reliably learned
- For simple subsequence kernels, planar languages cross-cut Chomsky hierarchy
- Our example languages are intuitively simple
  Intuitively, earning them should be simple
  Our approach makes it simple
- Further directions
  - Characterise set of planar languages with respect to various kernels
  - String processing in features space...

# Summary

- Planar languages: a new(?) approach to language induction from positive examples
- Natural to represent some languages in terms of linear constraints on sub-sequences
- Certain types of long-range dependency of intervals, counts, and sequences can be
  - neatly represented
  - reliably learned
- For simple subsequence kernels, planar languages cross-cut Chomsky hierarchy
- Our example languages are intuitively simple
  Intuitively, earning them should be simple
  Our approach makes it simple
- Further directions
  - Characterise set of planar languages with respect to various kernels
  - String processing in features space...

# Summary

- Planar languages: a new(?) approach to language induction from positive examples
- Natural to represent some languages in terms of linear constraints on sub-sequences
- Certain types of long-range dependency of intervals, counts, and sequences can be
  - neatly represented
  - reliably learned
- For simple subsequence kernels, planar languages cross-cut Chomsky hierarchy
- Our example languages are intuitively simple
  Intuitively, earning them should be simple
  Our approach makes it simple
- Further directions
  - Characterise set of planar languages with respect to various kernels
  - String processing in features space...

# Summary

- Planar languages: a new(?) approach to language induction from positive examples
- Natural to represent some languages in terms of linear constraints on sub-sequences
- Certain types of long-range dependency of intervals, counts, and sequences can be
  - neatly represented
  - reliably learned
- For simple subsequence kernels, planar languages cross-cut Chomsky hierarchy
- Our example languages are intuitively simple
  Intuitively, earning them should be simple
  Our approach makes it simple
- Further directions
  - Characterise set of planar languages with respect to various kernels
  - String processing in features space...