Kernel on Automata

Cousins of String Kernels and Dynamic Systems Kernels?

S.V.N. "Vishy" Vishwanathan
vishy@csa.iisc.ernet.in

Indian Insitutute of Science
Bangalore, India

Joint work with Alex Smola

Overview

- Introduction and Motivation
- Definition of Automata
- Kernels on Automata
- Kernels defined by Automata
- Applications of Automata Kernels

Introduction

- Automata are powerful abstractions
- HMM's, Dynamical systems, graphs etc. can be viewed as special cases
- Sometimes even input data can be modeled as Automata
- Many times we want to define kernels by using Automata
- We may also want to compare two Automata by defining kernels on them
- Our Automata kernels are also related to diffusion kernels on graphs, rational kernels on transducers and kernels on Dynamical systems

Notation

- Characters make up the alphabet set Σ
- Sequence of characters is a string
- A string is accepted by an Automata if there are a sequence of state transitions which lead from the initial state to the final state
- Set of all strings accepted by an Automata define its language (denoted by L)
- The language accepted by various families of Automata are well studied
- Computers can be modeled as Turing machines which are a kind of Automata !

Basic Idea

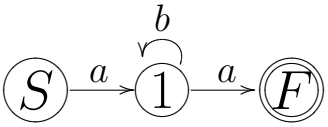
- Given two strings, if the state transitions they induce is *similar* then the two strings are *similar*
- If a set of strings result in *similar* state transitions in two different Automata then the Automata themselves are *similar*
- Using these two ideas we can talk of kernels defined by Automata and kernels on Automata
- This is a very generic framework and does not impose any restrictions on how you define similarity
- This means, for example, that time warped kernels can also be considered for defining similarity

Finite State Automata (FSA)

- Mathematical models to describe regular languages
- FSA is denoted by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$
- Q is the finite set of states
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is a the set of final states
- δ is a transition function mapping $Q \times \Sigma \rightarrow Q$
- In case of Non-deterministic FSA δ is a transition function $Q \times \Sigma \rightarrow 2^Q$
- In case of weighted FSA we also have weights associated with the transitions

Finite State Automata contd . . .

- Any language accepted by a NFA can also be accepted by a FSA
- Addition of ϵ transitions does not add to the expressive power of either FSA or NFA
- Addition of weighted transitions does not add to the expressive power of the NFA



Kernel Definition

- Every $x \in L$ induces a set of state transitions (denoted by $q(x)$) of the form $q_0 Q^k f$
- $s \sqsubseteq q(x)$ denotes that s occurs as a sub-sequence of some element of $q(x)$
- The generic kernel is defined as

$$k(x, x') = \sum_{\mathbf{x} \sqsubseteq q(x), \mathbf{x}' \sqsubseteq q(x')} \kappa(\mathbf{x}, \mathbf{x}')$$

- $\kappa(.,.)$ is a kernel function and depends on the application domain
- Sometimes a normalizing term is also added
- Note the correspondence with R-Convolution kernels of Haussler

Special Cases

Bag of States: Counts common states

$$\kappa(\mathbf{x}, \mathbf{x}') = \begin{cases} w_{\mathbf{x}} \delta_{\mathbf{x},\mathbf{x}'} & \text{if } w_{\mathbf{x}} \in Q \\ 0 & \text{otherwise} \end{cases}$$

Bag of State Sub-Sequences: Includes context

$$\kappa(\mathbf{x}, \mathbf{x}') = w_{\mathbf{x}} \delta_{\mathbf{x},\mathbf{x}'}$$

- Weights can also be assigned based on location of match
- Time warped sequence kernels may also be used but you have to pay the computational cost
- Gap penalities, decay factors and other fancy ideas can also be used

Context Free Grammar

- A Contex Free Grammar is denoted by $G = (V, T, P, S)$
- V is a finite set of variables
- T is a finite set of terminals
- S is a special variable called the start symbol
- P is a finite set of productions of the form $A \rightarrow \alpha$, where A is a variable and α is a string of symbols from $(V \cup T)^*$

Context Free Grammar . . .

- A string x is said to belong to the language if productions in the CFG can derive the string starting from S
- A parse tree of x is the tree representation of the productions that derive x
- In the case of an un-ambigous CFG each string x in the langugage corresponds to an unique parse tree
- A Push-down Automata is an abstraction which can accept an un-ambigous CFG

Kernel using a CFG

- Given two strings x and x' in the language generate their parse trees
- Compute the kernels using the parse trees
- Not as simple minded as it looks
- Structured languages like XML or HTML are parsed by a Push-down Automata to produce a DOM
- Our idea can also be used to compute kernels between say two web pages

Other Cases

- Every programming language is defined by a CFG which is accepted by some Push-down Automata
- This means we can now compute kernels between say two C programs!
- If we ignore the actual names of the variables, code duplication, plagarism etc. can be detected !
- Also has applications in efficient compression of structured text

Kernels on Dynamical Systems

- We consider very simple linear systems described by

$$x_{\mathbf{A}}(t) := \mathbf{A}(t)x \text{ for } \mathbf{A} \in \mathcal{A}$$

- For simplicity, in this talk, we assume that $\mathcal{A}$ consists of only single transformation
- We also assume that noise is absent (for details on how to use noisy models talk to me or Alex!)
- We define the kernel for this simple case as

$$k(x, \tilde{x}) := \mathbf{E}_{\mathbf{A}} \left[k((x, \mathbf{A}), (\tilde{x}, \mathbf{A})) \right].$$

- Cranking a few equations yields a kernel of the form

$$\sum_{t=0}^{\infty} e^{-\lambda t} \langle A^t x_0, \tilde{A}^t \tilde{x}_0 \rangle = \mathrm{tr}(\tilde{x}_0 x_0^\top) M$$

- Here M satisfies $e^{-\lambda} A^\top M \tilde{A} + \mathbf{1} = M$
- Such equations are called Sylvester equations and can be solved in $O(n^3)$ time by using widely available packages
- Challenge lies in finding efficient special cases which can be solved cheaply

Summary

- Automata are important abstractions
- It is important to define similarities using Automata
- They are closely related to Dynamical systems kernels