Spectral Methods for Learning Finite State Machines

Borja Balle

(based on joint work with X. Carreras, M. Mohri, and A. Quattoni)

LARCA. Laboratory for Relational Algorithmics, Complexity and Learning

UNIVERSITAT POLITÈCNICA DE CATALUNYA

McGill University

Montreal, September 2012

This work is partially supported by the PASCAL2 Network and a Google Research Award

Outline

Outline

Weighted Automata and Functions over Strings

A Spectral Method for Learning Weighted Automata

Survey of Recent Applications to Learning Problems

Conclusion

Notation

- Finite alphabet $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_r\}$
- Free monoid $\Sigma^\star = \{\epsilon, a, b, aa, ab, ba, bb, aaa, \ldots\}$
- Functions over strings $f : \Sigma^\star \to \mathbb{R}$
- Examples:

$$f(x) = \mathbb{P}[x] \qquad \text{(probability of a string)}$$

$$f(x) = \mathbb{P}[x\Sigma^\star] \qquad \text{(probability of a prefix)}$$

$$f(x) = \mathbb{I}[x \in L] \qquad \text{(characteristic function of language } L\text{)}$$

$$f(x) = |x|_a \qquad \text{(number of } a\text{'s in } x\text{)}$$

$$f(x) = \mathbb{E}[|w|_x] \qquad \text{(expected number of substrings equal to } x\text{)}$$

Weighted Automata

- Class of WA parametrized by alphabet Σ and number of states n

$$\mathbf{A} = \langle \alpha_1, \alpha_\infty, \{A_\sigma\}_{\sigma \in \Sigma} \rangle$$

$$\alpha_1 \in \mathbb{R}^n \qquad \text{(initial weights)}$$

$$\alpha_\infty \in \mathbb{R}^n \qquad \text{(terminal weights)}$$

$$A_\sigma \in \mathbb{R}^{n \times n} \qquad \text{(transition weights)}$$

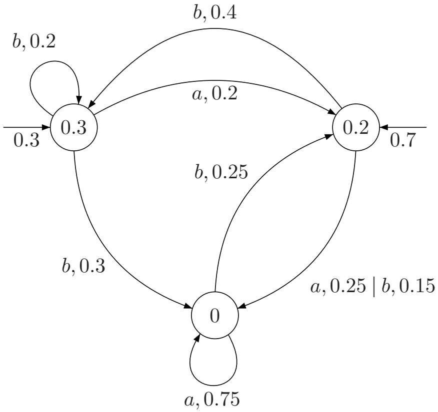
- Computes a function $f_{\mathbf{A}} : \Sigma^\star \to \mathbb{R}$

$$f_{\mathbf{A}}(x) = f_{\mathbf{A}}(x_1 \cdots x_t) = \alpha_1^\top A_{x_1} \cdots A_{x_t} \alpha_\infty = \alpha_1^\top A_x \alpha_\infty$$

Examples – Probabilistic Finite Automata

- Compute / generate distributions over strings $\mathbb{P}[x]$

$$\alpha_1^\top = [0.3 \;\; 0 \;\; 0.7]$$
$$\alpha_\infty^\top = [0.2 \;\; 0 \;\; 0.2]$$
$$A_a = \begin{bmatrix} 0 & 0 & 0.2 \\ 0 & 0.75 & 0 \\ 0 & 0.25 & 0 \end{bmatrix}$$



Examples – Hidden Markov Models

- Generates infinite strings, computes probabilities of prefixes $\mathbb{P}[x\Sigma^\star]$
- Emission and transition are conditionally independent given state

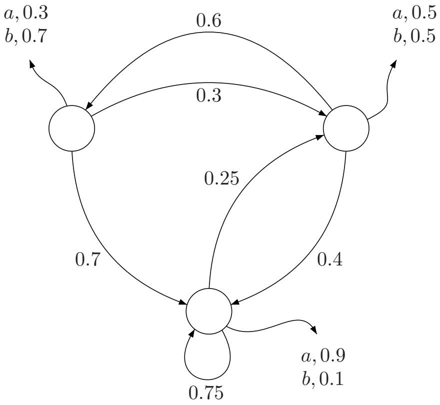
$$\alpha_1^\top = [0.3 \;\; 0.3 \;\; 0.4]$$

$$\alpha_\infty^\top = [1 \;\; 1 \;\; 1]$$

$$A_a = O_a \cdot T$$

$$T = \begin{bmatrix} 0 & 0.7 & 0.3 \\ 0 & 0.75 & 0.25 \\ 0 & 0.4 & 0.6 \end{bmatrix}$$

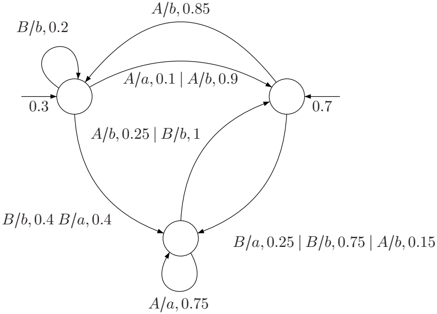
$$O_a = \begin{bmatrix} 0.3 & 0 & 0 \\ 0 & 0.9 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$$



Examples – Probabilistic Finite State Transducers

- Compute conditional probabilities $\mathbb{P}[y|x] = \alpha_1^\top A_x^y \alpha_\infty$ for pairs $(x, y) \in (\Sigma \times \Delta)^\star$, must have $|x| = |y|$
- Can also assume models factorized like in HMM

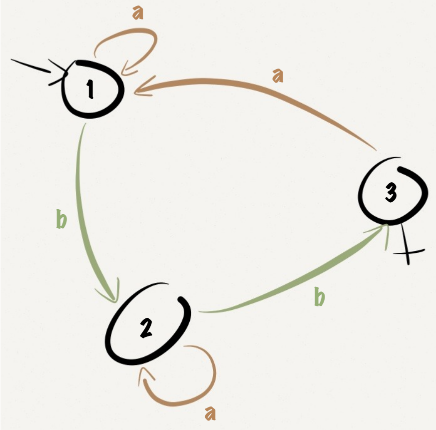
$$\alpha_1^\top = [0.3 \;\; 0 \;\; 0.7]$$
$$\alpha_\infty^\top = [1 \;\; 1 \;\; 1]$$
$$A_B^b = \begin{bmatrix} 0.2 & 0.4 & 0 \\ 0 & 0 & 1 \\ 0 & 0.75 & 0 \end{bmatrix}$$



Examples – Deterministic Finite Automata

- Compute membership in a regular language

$$\alpha_1^\top = [1 \;\; 0 \;\; 0]$$
$$\alpha_\infty^\top = [0 \;\; 0 \;\; 1]$$
$$A_a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$



Facts About Weighted Automata (I)

Invariance Under Change of Basis

- Let $Q \in \mathbb{R}^{n \times n}$ be *invertible*
- Let $Q\mathbf{A}Q^{-1} = \langle Q^{-\top}\alpha_1, Q\alpha_\infty, \{QA_\sigma Q^{-1}\}\rangle$
- Then $f_{\mathbf{A}} = f_{Q\mathbf{A}Q^{-1}}$ since

$$(\alpha_1^\top Q^{-1})(QA_{x_1}Q^{-1})\cdots(QA_{x_t}Q^{-1})(Q\alpha_\infty) = \alpha_1^\top A_{x_1}\cdots A_{x_t}\alpha_\infty$$

Example

$$A_a = \begin{bmatrix} 0.5 & 0.1 \\ 0.2 & 0.3 \end{bmatrix} \qquad Q = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad QA_aQ^{-1} = \begin{bmatrix} 0.3 & -0.2 \\ -0.1 & 0.5 \end{bmatrix}$$

Consequences

- For *learning* WA it is not necessary to recover original parametrization
- PFA is only one way to parametrize *probability distributions*
- Unfortunately, given $\mathbf{A}$ it is *undecidable* whether $\forall x\ f_{\mathbf{A}}(x) \geqslant 0$

Facts About Weighted Automata (II)

Forward–Backward Factorization

- $\mathbf{A}$ defines *forward* and *backward* maps $f_{\mathbf{A}}^{F}, f_{\mathbf{A}}^{B} : \Sigma^{\star} \to \mathbb{R}^{n}$
- Such that for any splitting $x = y \cdot z$ one has $f_{\mathbf{A}}(x) = f_{\mathbf{A}}^{F}(y) \cdot f_{\mathbf{A}}^{B}(z)$

$$f_{\mathbf{A}}^{F}(y) = \alpha_1^{\top} A_y \qquad \text{and} \qquad f_{\mathbf{A}}^{B}(z) = A_z \alpha_{\infty}$$

Example

- For a PFA $\mathbf{A}$ and $i \in [n]$, one has
- $[f_{\mathbf{A}}^{F}(y)]_i = [\alpha_1^{\top} A_y]_i = \mathbb{P}[y\,,\, h_{|y|+1} = i]$
- $[f_{\mathbf{A}}^{B}(z)]_i = [A_z \alpha_{\infty}]_i = \mathbb{P}[z \,|\, h = i]$

Consequences

- String structure has direct relation to computation structure
- In particular, strings sharing prefixes or suffixes share computations
- Information on A_a can be recovered from $f_{\mathbf{A}}(yaz)$, $f_{\mathbf{A}}^{F}(y)$, and $f_{\mathbf{A}}^{B}(z)$:

$$f_{\mathbf{A}}(yaz) = f_{\mathbf{A}}^{F}(y) A_a f_{\mathbf{A}}^{B}(z)$$

Outline

Learning Weighted Automata

Goal

- Given *some kind* of (partial) information about $f : \Sigma^\star \to \mathbb{R}$, find a weighted automata $\mathbf{A}$ such that $f \approx f_{\mathbf{A}}$

Types of Target

- Realizable case, $f = f_{\mathbf{B}}$ – exact learning, PAC learning
- Angostic setting, arbitrary f – agnostic learning, generalization bounds

Information on the Target

- Total knowledge (e.g. via queries) – algorithmic/compression problem
- Approximate global knowledge – noise filtering problem
- Exact local knowledge (e.g. random sampling) – interpolation problem

Learning Weighted Automata

Goal

- Given *some kind* of (partial) information about $f : \Sigma^\star \to \mathbb{R}$, find a weighted automata $\mathbf{A}$ such that $f \approx f_\mathbf{A}$

Types of Target

- Realizable case, $f = f_\mathbf{B}$ – exact learning, PAC learning
- Angostic setting, arbitrary f – agnostic learning, generalization bounds

Information on the Target

- Total knowledge (e.g. via queries) – algorithmic/compression problem
- Approximate global knowledge – noise filtering problem
- Exact local knowledge (e.g. random sampling) – interpolation problem

Precedents and Alternative Approaches

Related Work

- Subspace methods for identification of linear dynamical systems [Overschee–Moor '94]
- Results on identifiability and learning of HMM and phylogenetic trees [Chang '96, Mossel–Roch '06]
- Query learning algorithms for DFA and Multiplicity Automata [Angluin '87, Bergadano–Varrichio '94]

Other Spectral Methods

This presentation does not cover recent spectral learning methods for:

- Mixture models [Anandkumar et al. '12]
- Latent tree graphical models [Parikh et al. '11, Anandkumar et al. '11]
- Tree automata [Bailly et al. '10]
- Probabilistic context-free grammars [Cohen et al. '12]
- Models with continuous observables or feature maps [Song et al. '10]

The Hankel Matrix

- The *Hankel matrix* of $f : \Sigma^\star \to \mathbb{R}$ is $H_f \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$
- For $y, z \in \Sigma^\star$, entries are defined by $H_f(y, z) = f(y \cdot z)$
- Given $\mathcal{P}, \mathcal{S} \subseteq \Sigma^\star$ will consider sub-blocks $H_f(\mathcal{P}, \mathcal{S}) \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$
- Very *redundant* representation for f – $f(x)$ appears $|x| + 1$ times

$$
\begin{array}{c} \\ \epsilon \\ a \\ b \\ aa \\ ab \\ \vdots \end{array}
\begin{array}{c}
\begin{array}{cccccc} \epsilon & a & b & aa & ab & \cdots \end{array} \\
\left[\begin{array}{cccccc}
 & & \vdots & & \vdots & \\
\cdots & \cdots & \vdots & \cdots & f(aab) & \\
 & & \vdots & & & \\
\cdots & \cdots & f(aab) & & & \\
 & & & & & \\
 & & & & &
\end{array} \right]
\end{array}
$$

Schützenberger's Theorem

Theorem: $\mathrm{rank}(H_f) \leqslant n$ if and only if $f = f_{\mathbf{A}}$ with $|\mathbf{A}| = n$

In particular, $\mathrm{rank}(H_f)$ is size of smallest WA for f

Proof ($\Leftarrow$)

- Write $F = f^F_{\mathbf{A}}(\Sigma^\star) \in \mathbb{R}^{\Sigma^\star \times n}$ and $B = f^B_{\mathbf{A}}(\Sigma^\star) \in \mathbb{R}^{n \times \Sigma^\star}$
- Note $H_f = F \cdot B$
- Then, $\mathrm{rank}(H_f) \leqslant n$

Proof ($\Rightarrow$)

- Assume $\mathrm{rank}(H_f) = n$
- Take *rank factorization* $H_f = F \cdot B$ with $F \in \mathbb{R}^{\Sigma^\star \times n}$ and $B \in \mathbb{R}^{n \times \Sigma^\star}$
- Let $\alpha_1^\top = F(\epsilon, [n])$ and $\alpha_\infty = B([n], \epsilon)$ (note $\alpha_1^\top \alpha_\infty = f(\epsilon)$)
- Let $A_\sigma = B([n], \sigma \cdot \Sigma^\star) \cdot B^+ \in \mathbb{R}^{n \times n}$ (note $A_\sigma \cdot B([n], x) = B([n], \sigma \cdot x)$)
- By induction on $|x|$ we get $\alpha_1^\top A_x \alpha_\infty = f(x)$

Schützenberger's Theorem

Theorem: $\mathrm{rank}(H_f) \leqslant n$ if and only if $f = f_{\mathbf{A}}$ with $|\mathbf{A}| = n$
In particular, $\mathrm{rank}(H_f)$ is size of smallest WA for f

Proof ($\Leftarrow$)

- Write $F = f^F_{\mathbf{A}}(\Sigma^\star) \in \mathbb{R}^{\Sigma^\star \times n}$ and $B = f^B_{\mathbf{A}}(\Sigma^\star) \in \mathbb{R}^{n \times \Sigma^\star}$
- Note $H_f = F \cdot B$
- Then, $\mathrm{rank}(H_f) \leqslant n$

Proof ($\Rightarrow$)

- Assume $\mathrm{rank}(H_f) = n$
- Take *rank factorization* $H_f = F \cdot B$ with $F \in \mathbb{R}^{\Sigma^\star \times n}$ and $B \in \mathbb{R}^{n \times \Sigma^\star}$
- Let $\alpha_1^\top = F(\epsilon, [n])$ and $\alpha_\infty = B([n], \epsilon)$ (note $\alpha_1^\top \alpha_\infty = f(\epsilon)$)
- Let $A_\sigma = B([n], \sigma \cdot \Sigma^\star) \cdot B^+ \in \mathbb{R}^{n \times n}$ (note $A_\sigma \cdot B([n], x) = B([n], \sigma \cdot x)$)
- By induction on $|x|$ we get $\alpha_1^\top A_x \alpha_\infty = f(x)$

Schützenberger's Theorem

Theorem: $\text{rank}(H_f) \leqslant n$ if and only if $f = f_{\mathbf{A}}$ with $|\mathbf{A}| = n$

In particular, $\text{rank}(H_f)$ is size of smallest WA for f

Proof ($\Leftarrow$)

- Write $F = f_{\mathbf{A}}^{F}(\Sigma^\star) \in \mathbb{R}^{\Sigma^\star \times n}$ and $B = f_{\mathbf{A}}^{B}(\Sigma^\star) \in \mathbb{R}^{n \times \Sigma^\star}$
- Note $H_f = F \cdot B$
- Then, $\text{rank}(H_f) \leqslant n$

Proof ($\Rightarrow$)

- Assume $\text{rank}(H_f) = n$
- Take *rank factorization* $H_f = F \cdot B$ with $F \in \mathbb{R}^{\Sigma^\star \times n}$ and $B \in \mathbb{R}^{n \times \Sigma^\star}$
- Let $\alpha_1^\top = F(\epsilon, [n])$ and $\alpha_\infty = B([n], \epsilon)$ (note $\alpha_1^\top \alpha_\infty = f(\epsilon)$)
- Let $A_\sigma = B([n], \sigma \cdot \Sigma^\star) \cdot B^+ \in \mathbb{R}^{n \times n}$ (note $A_\sigma \cdot B([n], x) = B([n], \sigma \cdot x)$)
- By induction on $|x|$ we get $\alpha_1^\top A_x \alpha_\infty = f(x)$

Towards the Spectral Method

Remarks about the $(\Rightarrow)$ proof

- A *finite* sub-block $\mathrm{H} = \mathrm{H}_f(\mathcal{P}, \mathcal{S})$ such that $\mathrm{rank}(\mathrm{H}) = \mathrm{rank}(\mathrm{H}_f)$ is *sufficient* – $(\mathcal{P}, \mathcal{S})$ is called a *basis* when also $\epsilon \in \mathcal{P} \cap \mathcal{S}$
- A *compatible* factorization of H and $\mathrm{H}_\sigma = \mathrm{H}_f(\mathcal{P}, \sigma \cdot \mathcal{S})$ is needed – $A_\sigma = \mathrm{B}([n], \sigma \cdot \mathcal{S}) \cdot \mathrm{B}([n], \mathcal{S})^+$ (in fact, for all σ)

Another expression for A_σ

- Instead of factorizing H and H_σ, do ...
- Factorize only $\mathrm{H} = \mathrm{B} \cdot \mathrm{F}$ and note $\mathrm{H}_\sigma = \mathrm{B} \cdot A_\sigma \cdot \mathrm{F}$
- Solving yields $A_\sigma = \mathrm{B}^+ \cdot \mathrm{H}_\sigma \cdot \mathrm{F}^+$
- Also, $\alpha_1^\top = \mathrm{H}(\epsilon, \mathcal{S}) \cdot \mathrm{F}^+$ and $\alpha_\infty = \mathrm{B}^+ \cdot \mathrm{H}(\mathcal{P}, \epsilon)$

The Spectral Method

Idea: Use SVD decomposition to obtain a factorization of H

- Given H and H_σ over basis $(\mathcal{P}, \mathcal{S})$
- Compute *compact* SVD as $H = USV^\top$ with

$$U \in \mathbb{R}^{\mathcal{P} \times n} \qquad S \in \mathbb{R}^{n \times n} \qquad V \in \mathbb{R}^{\mathcal{S} \times n}$$

- Let $A_\sigma = (HV)^+(H_\sigma V)$ – corresponds to rank factorization $H = (HV)V^\top$

Properties

- Easy to implement: just linear algebra
- Fast to compute: $O(\max\{|\mathcal{P}|, |\mathcal{S}|\}^3)$
- Noise tolerant: $\hat{H} \approx H$ and $\hat{H}_\sigma \approx H_\sigma$ implies $\hat{A}_\sigma \approx A_\sigma$
  $\Rightarrow$ learning!

Outline

Overview (of a biased selection)

Direct Applications

- Learning *stochastic rational languages* – any probability distribution computed by WA
- Learning *probabilistic finite state transducers* – learn $\mathbb{P}[y|x]$ from examples pairs (x, y)

Composition with Other Methods

- Combination with *matrix completion* for learning *non-stochastic* functions – when $f : \Sigma^\star \to \mathbb{R}$ is not related to a probability distribution

Algorithmic and Miscellaneous Problems

- Interpretation as an *optimization problem* – from linear algebra to convex optimization
- Finding a *basis* via random sampling – knowing $(\mathcal{P}, \mathcal{S})$ is a prerequisite for learning

Learning Stochastic Rational Languages [HKZ'09, BDR'09, etc.]

Idea: Given sample from probability distribution $f_{\mathbf{A}}$ over $\Sigma^\star$ find a WA $\hat{\mathbf{A}}$

Algorithms

- Given a basis, use the sample to compute $\hat{H}$ and $\hat{H}_\sigma$
- Apply the spectral method to obtain $\hat{A}_\sigma$

Properties

- Can PAC learn any distribution computed by a WA (w.r.t. L_1 distance)
- May not output a probability distribution
- Sample bound $\mathrm{poly}(1/\varepsilon, \log(1/\delta), n, |\Sigma|, |\mathcal{P}|, |\mathcal{S}|, 1/s_n(H), 1/s_n(B))$

Open problems / Future Work

- Learn models *guaranteed to be probability distributions* [Bailly '11]
- Study *inference* problems in such models
- Provide *smoothing* procedures, PAC learn w.r.t. KL
- How do *"infrequent"* states in the target affect learning?

Learning Probabilistic Finite State Transducers [BQC'11]

Idea: Learn a function $f : (\Sigma \times \Delta)^\star \to \mathbb{R}$ computing $\mathbb{P}[y|x]$

Learning Model

- Input is sample of aligned sequences (x^i, y^i), $|x^i| = |y^i|$
- Drawn i.i.d. from distribution $\mathbb{P}[x, y] = \mathbb{P}[y|x]\, D(x)$
- Want to assume as little as possible on D
- Performance measured against x generated from D

Properties

- Assuming independece $A^\delta_\sigma = O_\delta \cdot T_\sigma$, sample bound scales mildly with input alphabet $|\Sigma|$
- For applications, need to align sequences prior to learning – or use iterative procedures

Open problems / Future Work

- Deal with *alignments* inside the model
- *Smoothing and inference* questions (again!)

Matrix Completion and Spectral Learning [BM'12]

Idea

- In *stochastic* learning tasks (e.g. $\mathbb{P}[x]$, $\mathbb{P}[x\Sigma^\star]$, $\mathbb{E}[|w|_x]$) a sample S yields *global approximate* knowledge $\hat{f}_S$
- *Supervised learning* setup is given pairs $(x, f(x))$, where $x \sim D$
- But spectral method needs *(approximate) information on sub-blocks*
- Matrix completion finds missing entries under contraints (e.g. low rank Hankel matrix), then apply spectral method

$$\{(x^i, f(x^i))\} \longrightarrow \begin{bmatrix} 2 & \star & 0 \\ 1 & \star & \star \\ 0 & 1 & 4 \end{bmatrix} \xrightarrow{\text{matrix completion}} \begin{bmatrix} 2 & 1.1 & 0 \\ 1 & 2.3 & 1.1 \\ 0 & 1 & 4 \end{bmatrix}$$

Result: Generalization bounds for some MC + SM combinations

Open problems / Future Work

- Design *specific convex optimization algorithm* for completion problem
- Analyze combination with *other completion algorithms*

An Optimization Point of View [BQC'12]

Idea: Replace *linear algebra* with *optimization* primitives – make it possible to use the "ML optimization toolkit"

Algorithms

- Spectral optimization: $\min_{\{A_\sigma\}, V_n^\top V_n = I} \sum_{\sigma \in \Sigma} \|H V_n A_\sigma - H_\sigma V_n\|_F^2$
- Convex relaxation: $\min_{A_\Sigma} \|H A_\Sigma - H_\Sigma\|_F^2 + \tau \|A_\Sigma\|_\star$

Properties

- Equivalent in some situations and choice of parameters
- Experiments show convex relaxation can be better in cases known to be difficult for the spectral method

Open problems / Future Work

- Design *problem-specific* optimization algorithms
- Constrain learned models imposing further *regularizations*, e.g. sparsity

Finding a Basis [BQC '12]

Idea: Choose a basis in a data-driven manner – as oposed to using a fixed set of prefixes and suffixes

Algorithm

Input: strings $(x^1, \ldots, x^N)$
Initialize $\mathcal{P} \leftarrow \emptyset$, $\mathcal{S} \leftarrow \emptyset$
for $i = 1$ **to** N **do**
  Choose $0 \leqslant t \leqslant |x^i|$ u.a.r.
  Split $x^i = u^i v^i$ with $|u^i| = t$
  and $|v^i| = |x^i| - t$
  Add u_i to $\mathcal{P}$ and v^i to $\mathcal{S}$
end for

Result

- x^i i.i.d. from distribution D over $\Sigma^\star$ with full support
- $f = f_{\mathbf{A}}$ with $\|A_\sigma\| \leqslant 1$
- If $N \geqslant C\eta(f, D)\log(1/\delta)$ then $(\mathcal{P}, \mathcal{S})$ is basis w.h.p.

Open problems / Future Work

- Do something more *smart* and *practical*
- Find *smaller* basis containing *shorter* strings

Outline

Take-home Message

- *Efficient, easy to implement* learning method
- Alternative to EM not suffering from *local minima*
- Can be extended to *many* probabilistic (and some non-probabilistic) models
- Comes with *theoretical analysis*, quantifies *hardness* of models, provides *intuitions*
- Lots of interesting open problems, theoretical *and* practical

Spectral Methods for Learning Finite State Machines

Borja Balle
(based on joint work with X. Carreras, M. Mohri, and A. Quattoni)

LARCA. Laboratory for Relational Algorithmics, Complexity and Learning
UNIVERSITAT POLITÈCNICA DE CATALUNYA

McGill University
Montreal, September 2012

This work is partially supported by the PASCAL2 Network and a Google Research Award