Spectral Learning Techniques for Weighted Automata, Transducers, and Grammars

Borja Balle[♢] Ariadna Quattoni[♡] Xavier Carreras[♡]

McGill

(♢) McGill University

xerox

(♡) Xerox Research Centre Europe

TUTORIAL @ EMNLP 2014

Latest Version Available from:

http://www.lsi.upc.edu/~bballe/slides/tutorial-emnlp14.pdf

Outline

Compositional Functions and Bilinear Operators

- Compositional functions defined in terms of recurrence relations
- Consider a sequence $\mathtt{abaccb}$

$$
\begin{aligned}
f(abaccb) &= \boldsymbol{\alpha}_f(ab) \cdot \boldsymbol{\beta}_f(accb) \\
&= \boldsymbol{\alpha}_f(ab) \cdot \mathbf{A}_a \cdot \boldsymbol{\beta}_f(ccb) \\
&= \boldsymbol{\alpha}_f(aba) \cdot \mathbf{A}_c \cdot \boldsymbol{\beta}_f(cb)
\end{aligned}
$$

where

- n is the dimension of the model
- $\boldsymbol{\alpha}_f$ maps prefixes to $\mathbb{R}^n$
- $\boldsymbol{\beta}_f$ maps suffixes to $\mathbb{R}^n$
- $\mathbf{A}_a$ is a bilinear operator in $\mathbb{R}^{n \times n}$

Problem

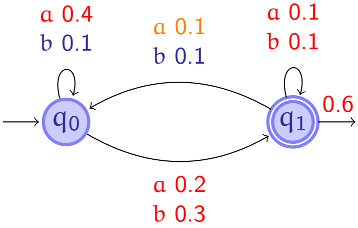
How to estimate $\boldsymbol{\alpha}_f$, $\boldsymbol{\beta}_f$ and $\mathbf{A}_a, \mathbf{A}_b, \ldots$ from "samples" of f?

Weighted Finite Automata (WFA)

An algebraic model
for compositional functions on strings

Weighted Finite Automata (WFA)

Example with 2 states and alphabet $\Sigma = \{a, b\}$



Operator Representation

$$\boldsymbol{\alpha}_0 = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix}$$

$$\boldsymbol{\alpha}_\infty = \begin{bmatrix} 0.0 \\ 0.6 \end{bmatrix}$$

$$\mathbf{A}_a = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.1 \end{bmatrix}$$

$$\mathbf{A}_b = \begin{bmatrix} 0.1 & 0.3 \\ 0.1 & 0.1 \end{bmatrix}$$

$$f(ab) = \boldsymbol{\alpha}_0^\top \mathbf{A}_a \mathbf{A}_b \boldsymbol{\alpha}_\infty$$

Weighted Finite Automata (WFA)

Notation:

- Σ: alphabet – finite set
- n: number of states – positive integer
- $\boldsymbol{\alpha}_0$: initial weights – vector in $\mathbb{R}^n$ (features of empty prefix)
- $\boldsymbol{\alpha}_\infty$: final weights – vector in $\mathbb{R}^n$ (features of empty suffix)
- $\mathbf{A}_\sigma$: transition weights – matrix in $\mathbb{R}^{n \times n}$ $(\forall \sigma \in \Sigma)$

Definition: WFA with n states over Σ

$$A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\} \rangle$$

Compositional Function: Every WFA A defines a function $f_A : \Sigma^\star \to \mathbb{R}$

$$f_A(x) = f_A(x_1 \ldots x_T) = \boldsymbol{\alpha}_0^\top \mathbf{A}_{x_1} \cdots \mathbf{A}_{x_T} \boldsymbol{\alpha}_\infty = \boldsymbol{\alpha}_0^\top \mathbf{A}_x \boldsymbol{\alpha}_\infty$$

Example – Hidden Markov Model

- Assigns probabilities to strings $f(x) = \mathbb{P}[x]$
- Emission and transition are conditionally independent given state

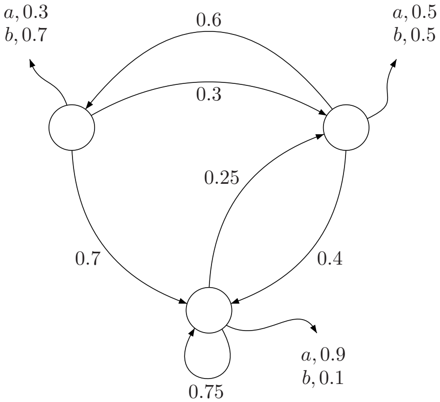
$$\boldsymbol{\alpha}_0^\top = [0.3 \;\; 0.3 \;\; 0.4]$$

$$\boldsymbol{\alpha}_\infty^\top = [1 \;\; 1 \;\; 1]$$

$$\mathbf{A}_a = \mathbf{O}_a \cdot \mathbf{T}$$

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

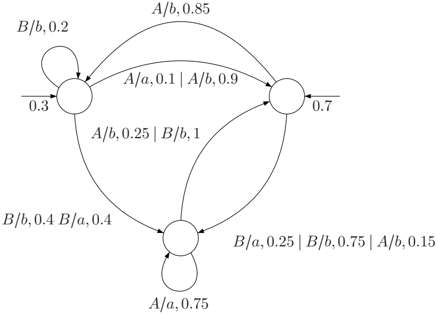
$$\mathbf{O}_a = \begin{bmatrix} 0.3 & 0 & 0 \\ 0 & 0.9 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$$



Example – Probabilistic Transducer

- $\Sigma = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ input alphabet and $\mathcal{Y}$ output alphabet
- Assigns conditional probabilities $f(x, y) = \mathbb{P}[y|x]$ to pairs $(x, y) \in \Sigma^\star$

$$\mathcal{X} = \{A, B\}$$
$$\mathcal{Y} = \{a, b\}$$
$$\boldsymbol{\alpha}_0^\top = [0.3 \;\; 0 \;\; 0.7]$$
$$\boldsymbol{\alpha}_\infty^\top = [1 \;\; 1 \;\; 1]$$
$$\mathbf{A}_B^b = \begin{bmatrix} 0.2 & 0.4 & 0 \\ 0 & 0 & 1 \\ 0 & 0.75 & 0 \end{bmatrix}$$



Other Examples of WFA

Automata-theoretic:

- Probabilistic Finite Automata (PFA)
- Deterministic Finite Automata (DFA)

Dynamical Systems:

- Observable Operator Models (OOM)
- Predictive State Representations (PSR)

Disclaimer: All weights in $\mathbb{R}$ with usual addition and multiplication (*no semi-rings!*)

Applications of WFA

WFA Can Model:

- Probability distributions $f_A(x) = \mathbb{P}[x]$
- Binary classifiers $g(x) = \mathrm{sign}(f_A(x) + \theta)$
- Real predictors $f_A(x)$
- Sequence predictors $g(x) = \mathrm{argmax}_y f_A(x, y)$ (with $\Sigma = \mathcal{X} \times \mathcal{Y}$)

Used In Several Applications:

- Speech recognition [Mohri, Pereira, and Riley '08]
- Image processing [Albert and Kari '09]
- OCR systems [Knight and May '09]
- System testing [Baier, Grösser, and Ciesinski '09]
- *etc.*

Useful Intuitions About f_A

$$f_A(x) = f_A(x_1 \dots x_T) = \boldsymbol{\alpha}_0^\top \mathbf{A}_{x_1} \cdots \mathbf{A}_{x_T} \boldsymbol{\alpha}_\infty = \boldsymbol{\alpha}_0^\top \mathbf{A}_x \boldsymbol{\alpha}_\infty$$

- Sum-Product: $f_A(x)$ is a sum–product computation

$$\sum_{i_0, i_1, \dots, i_T \in [n]} \boldsymbol{\alpha}_0(i_0) \left(\prod_{t=1}^{T} \mathbf{A}_{x_t}(i_{t-1}, i_t) \right) \boldsymbol{\alpha}_\infty(i_T)$$

- Forward-Backward: $f_A(x)$ is dot product between forward and backward vectors

$$f_A(ps) = \left(\boldsymbol{\alpha}_0^\top \mathbf{A}_p\right) \cdot (\mathbf{A}_s \boldsymbol{\alpha}_\infty) = \boldsymbol{\alpha}_p \cdot \boldsymbol{\beta}_s$$

- Compositional Features: $f_A(x)$ is a linear model

$$f_A(x) = \left(\boldsymbol{\alpha}_0^\top \mathbf{A}_x\right) \cdot \boldsymbol{\alpha}_\infty = \boldsymbol{\phi}(x) \cdot \boldsymbol{\alpha}_\infty$$

where $\boldsymbol{\phi} : \Sigma^\star \to \mathbb{R}^n$ *compositional features* (i.e. $\boldsymbol{\phi}(x\sigma) = \boldsymbol{\phi}(x)\mathbf{A}_\sigma$)

Forward–Backward Equations for $\mathbf{A}_\sigma$

Any WFA A defines *forward* and *backward* maps

$$\boldsymbol{\alpha}_A, \boldsymbol{\beta}_A : \Sigma^\star \to \mathbb{R}^n$$

such that for any splitting $x = p \cdot s$ one has

$$\boldsymbol{\alpha}_A(p) = \boldsymbol{\alpha}_0^\top \mathbf{A}_{p_1} \cdots \mathbf{A}_{p_T}$$
$$\boldsymbol{\beta}_A(s) = \mathbf{A}_{s_1} \cdots \mathbf{A}_{s_{T'}} \boldsymbol{\alpha}_\infty$$
$$f_A(x) = \boldsymbol{\alpha}_A(p) \cdot \boldsymbol{\beta}_A(s)$$

Example

- In HMM and PFA one has for every $i \in [n]$

$$[\boldsymbol{\alpha}_A(p)]_i = \mathbb{P}[p\,,\, h_{+1} = i]$$
$$[\boldsymbol{\beta}_A(s)]_i = \mathbb{P}[s \mid h = i]$$

Forward–Backward Equations for $\mathbf{A}_\sigma$

Any WFA A defines *forward* and *backward* maps

$$\boldsymbol{\alpha}_A, \boldsymbol{\beta}_A : \Sigma^\star \to \mathbb{R}^n$$

such that for any splitting $x = p \cdot s$ one has

$$\boldsymbol{\alpha}_A(p) = \boldsymbol{\alpha}_0^\top \mathbf{A}_{p_1} \cdots \mathbf{A}_{p_T}$$
$$\boldsymbol{\beta}_A(s) = \mathbf{A}_{s_1} \cdots \mathbf{A}_{s_{T'}} \boldsymbol{\alpha}_\infty$$
$$f_A(x) = \boldsymbol{\alpha}_A(p) \cdot \boldsymbol{\beta}_A(s)$$

Key Observation

If $f_A(p\sigma s)$, $\boldsymbol{\alpha}_A(p)$, and $\boldsymbol{\beta}_A(s)$ were known for many p, s, then $\mathbf{A}_\sigma$ could be recovered from equations of the form

$$f_A(p\sigma s) = \boldsymbol{\alpha}_A(p) \cdot \mathbf{A}_\sigma \cdot \boldsymbol{\beta}_A(s)$$

Hankel matrices help organize these equations!

The Hankel Matrix

Two Equivalent Representations

- Functional: $f : \Sigma^\star \to \mathbb{R}$
- Matricial: $\mathbf{H}_f \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$, the *Hankel matrix* of f

Definition: p prefix, s suffix $\Rightarrow$ $\mathbf{H}_f(p, s) = f(p \cdot s)$

Properties

- $|x| + 1$ entries for $f(x)$
- Depends on ordering of $\Sigma^\star$
- Captures structure

$$\mathbf{H}_f = \begin{array}{c} \\ \epsilon \\ a \\ b \\ aa \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccc} \epsilon & a & b & aa & \cdots \end{array} \\ \left[\begin{array}{ccccc} 0 & 1 & 0 & 2 & \cdots \\ 1 & 2 & 1 & 3 & \\ 0 & 1 & 0 & 2 & \\ 2 & 3 & 2 & 4 & \\ \vdots & & & & \ddots \end{array}\right] \end{array}$$

$$\mathbf{H}_f(\epsilon, aa) = \mathbf{H}_f(a, a) = \mathbf{H}_f(aa, \epsilon) = 2$$

A Fundamental Theorem about WFA

Relates the rank of $\mathbf{H}_f$

and the number of states of WFA computing f

Theorem [Carlyle and Paz '71, Fliess '74]

Let $f : \Sigma^\star \to \mathbb{R}$ be any function

1. If $f = f_A$ for some WFA A with n states $\Rightarrow \mathrm{rank}(\mathbf{H}_f) \leqslant n$
2. If $\mathrm{rank}(\mathbf{H}_f) = n \Rightarrow$ exists WFA A with n states s.t. $f = f_A$

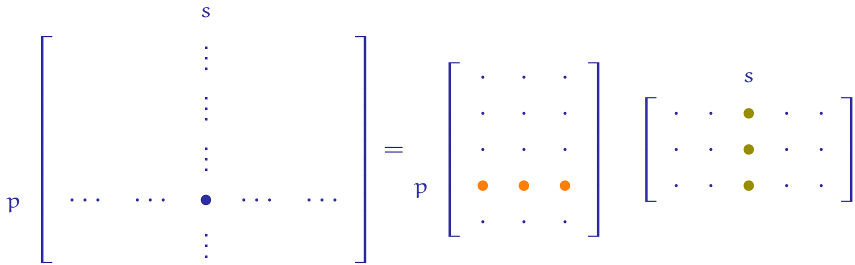
Why Fundamental?

Because proof of (2) gives an algorithm for "recovering" A from the Hankel matrix of f_A

Example: Can "recover" an HMM from the probabilities it assigns to sequences of observations

Structure of Low-rank Hankel Matrices

$\mathbf{H}_f \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$ $\quad$ $\mathbf{P} \in \mathbb{R}^{\Sigma^\star \times n}$ $\quad$ $\mathbf{S} \in \mathbb{R}^{n \times \Sigma^\star}$

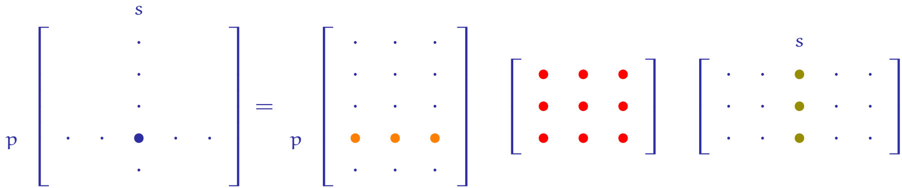


$$f(p_1 \cdots p_T \cdot s_1 \cdots s_{T'}) = \underbrace{\boldsymbol{\alpha}_0^\top \mathbf{A}_{p_1} \cdots \mathbf{A}_{p_T}}_{\boldsymbol{\alpha}_A(p)} \ \underbrace{\mathbf{A}_{s_1} \cdots \mathbf{A}_{s_{T'}} \boldsymbol{\alpha}_\infty}_{\boldsymbol{\beta}_A(s)}$$

$$\boldsymbol{\alpha}_A(p) = \mathbf{P}(p, \cdot) \qquad \boldsymbol{\beta}_A(s) = \mathbf{S}(\cdot, s)$$

Hankel Factorizations and Operators

$\mathbf{H}_\sigma \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$ $\mathbf{P} \in \mathbb{R}^{\Sigma^\star \times n}$ $\mathbf{A}_\sigma \in \mathbb{R}^{n \times n}$ $\mathbf{S} \in \mathbb{R}^{n \times \Sigma^\star}$



$$f(p_1 \cdots p_T \cdot \sigma \cdot s_1 \cdots s_{T'}) = \underbrace{\boldsymbol{\alpha}_0^\top \mathbf{A}_{p_1} \cdots \mathbf{A}_{p_T}}_{\boldsymbol{\alpha}_A(p)} \cdot \mathbf{A}_\sigma \cdot \underbrace{\mathbf{A}_{s_1} \cdots \mathbf{A}_{s_{T'}} \boldsymbol{\alpha}_\infty}_{\boldsymbol{\beta}_A(s)}$$

$$\mathbf{H}_\sigma = \mathbf{P}\ \mathbf{A}_\sigma\ \mathbf{S} \implies \mathbf{A}_\sigma = \mathbf{P}^+\ \mathbf{H}_\sigma\ \mathbf{S}^+$$

Note: Works with finite sub-blocks as well (assuming $\mathrm{rank}(\mathbf{P}) = \mathrm{rank}(\mathbf{S}) = n$)

General Learning Algorithm for WFA

Key Idea: The Hankel Trick

1. Learn a low-rank Hankel matrix that *implicitly* induces "latent" states
2. Recover the states from a decomposition of the Hankel matrix

Limitations of WFA

Invariance Under Change of Basis

For any invertible matrix $\mathbf{Q}$ the following WFA are equivalent:

- $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\} \rangle$
- $B = \langle \mathbf{Q}^\top \boldsymbol{\alpha}_0, \mathbf{Q}^{-1} \boldsymbol{\alpha}_\infty, \{\mathbf{Q}^{-1} \mathbf{A}_\sigma \mathbf{Q}\} \rangle$

$$
\begin{aligned}
f_A(x) &= \boldsymbol{\alpha}_0^\top \mathbf{A}_{x_1} \cdots \mathbf{A}_{x_T} \boldsymbol{\alpha}_\infty \\
&= (\boldsymbol{\alpha}_0^\top \mathbf{Q})(\mathbf{Q}^{-1} \mathbf{A}_{x_1} \mathbf{Q}) \cdots (\mathbf{Q}^{-1} \mathbf{A}_{x_T} \mathbf{Q})(\mathbf{Q}^{-1} \boldsymbol{\alpha}_\infty) = f_B(x)
\end{aligned}
$$

Example

$$
\mathbf{A}_a = \begin{bmatrix} 0.5 & 0.1 \\ 0.2 & 0.3 \end{bmatrix} \qquad \mathbf{Q} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad \mathbf{Q}^{-1} \mathbf{A}_a \mathbf{Q} = \begin{bmatrix} 0.3 & -0.2 \\ -0.1 & 0.5 \end{bmatrix}
$$

Consequences

- There is no *unique* parametrization for WFA
- Given A it is *undecidable* whether $\forall x \; f_A(x) \geqslant 0$
- Cannot expect to recover a *probabilistic* parametrization

Outline

Spectral Learning of Probabilistic Automata

Basic Setup:

- Data are strings sampled from probability distribution on $\Sigma^\star$
- Hankel matrix is estimated by empiricial probabilities
- Factorization and low-rank approximation is computed using SVD

The Empirical Hankel Matrix

Suppose $S = (x^1, \dots, x^N)$ is a sample of N i.i.d. strings

Empirical distribution:

$$\hat{f}_S(x) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}[x^i = x]$$

Empirical Hankel matrix:

$$\hat{\mathbf{H}}_S(p, s) = \hat{f}_S(ps)$$

Example:

$$S = \left\{ \begin{array}{c} aa,\ b,\ bab,\ a, \\ b,\ a,\ ab,\ aa, \\ ba,\ b,\ aa,\ a, \\ aa,\ bab,\ b,\ aa \end{array} \right\} \longrightarrow \hat{\mathbf{H}} = \begin{array}{c} \\ \epsilon \\ a \\ b \\ ba \end{array} \begin{array}{c} \begin{array}{cc} a & b \end{array} \\ \left[\begin{array}{cc} .19 & .25 \\ .31 & .06 \\ .06 & .00 \\ .00 & .13 \end{array} \right] \end{array}$$

(Hankel with rows $\mathcal{P} = \{\epsilon, a, b, ba\}$ and columns $\mathcal{S} = \{a, b\}$)

Finite Sub-blocks of Hankel Matrices

Parameters:

- Set of rows (prefixes) $\mathcal{P} \subset \Sigma^\star$
- Set of columns (suffixes) $\mathcal{S} \subset \Sigma^\star$

$\Sigma^\star$	λ	a	b	aa	ab	...
λ	1	0.3	0.7	0.05	0.25	...
a	0.3	0.05	0.25	0.02	0.03	...
b	0.7	0.6	0.1	0.03	0.2	...
aa	0.05	0.02	0.03	0.017	0.003	...
ab	0.25	0.23	0.02	0.11	0.12	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

$\mathbf{H}$

$\mathbf{H}_a$

- $\mathbf{H}$ for finding $\mathbf{P}$ and $\mathbf{S}$
- $\mathbf{H}_\sigma$ for finding $\mathbf{A}_\sigma$
- $\mathbf{h}_{\lambda,\mathcal{S}}$ for finding $\boldsymbol{\alpha}_0$
- $\mathbf{h}_{\mathcal{P},\lambda}$ for finding $\boldsymbol{\alpha}_\infty$

Low-rank Approximation and Factorization

Parameters:

- Desired number of states n
- Block $\mathbf{H} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$ of the empirical Hankel matrix

Low-rank Approximation: compute truncated SVD of rank n

$$\underbrace{\mathbf{H}}_{\mathcal{P} \times \mathcal{S}} \approx \underbrace{\mathbf{U}_n}_{\mathcal{P} \times n} \underbrace{\mathbf{\Lambda}_n}_{n \times n} \underbrace{\mathbf{V}_n^\top}_{n \times \mathcal{S}}$$

Factorization: $\mathbf{H} \approx \mathbf{P}\mathbf{S}$ already given by SVD

$$\mathbf{P} = \mathbf{U}_n \mathbf{\Lambda}_n \quad \Rightarrow \quad \mathbf{P}^+ = \mathbf{\Lambda}_n^{-1} \mathbf{U}_n^\top \left(= (\mathbf{H}\mathbf{V}_n)^+\right)$$

$$\mathbf{S} = \mathbf{V}_n^\top \quad \Rightarrow \quad \mathbf{S}^+ = \mathbf{V}_n$$

Computing the WFA

Parameters:

- Factorization $\mathbf{H} \approx (\mathbf{U}\boldsymbol{\Lambda})\mathbf{V}^\top$
- Hankel blocks $\mathbf{H}_\sigma$, $\mathbf{h}_{\lambda,\mathcal{S}}$, $\mathbf{h}_{\mathcal{P},\lambda}$

$$\mathbf{A}_\sigma = \boldsymbol{\Lambda}^{-1}\mathbf{U}^\top \mathbf{H}_\sigma \mathbf{V} \left(= (\mathbf{H}\mathbf{V})^+ \mathbf{H}_\sigma \mathbf{V}\right)$$
$$\boldsymbol{\alpha}_0 = \mathbf{V}^\top \mathbf{h}_{\lambda,\mathcal{S}}$$
$$\boldsymbol{\alpha}_\infty = \boldsymbol{\Lambda}^{-1}\mathbf{U}^\top \mathbf{h}_{\mathcal{P},\lambda} \left(= (\mathbf{H}\mathbf{V})^+ \mathbf{h}_{\mathcal{P},\lambda}\right)$$

Computational and Statistical Complexity

Running Time:

- Empirical Hankel matrix: $O(|\mathcal{PS}| \cdot N)$
- SVD and linear algebra: $O(|\mathcal{P}| \cdot |\mathcal{S}| \cdot n)$

Statistical Consistency:

- By law of large numbers, $\hat{\mathbf{H}}_{\mathcal{S}} \to \mathbb{E}[\mathbf{H}]$ when $N \to \infty$
- If $\mathbb{E}[\mathbf{H}]$ is Hankel of some WFA A, then $\hat{A} \to A$
- Works for data coming from PFA and HMM

PAC Analysis: (assuming data from A with n states)

- With high probability, $\|\hat{\mathbf{H}}_{\mathcal{S}} - \mathbf{H}\| \leqslant O(N^{-1/2})$
- When $N \geqslant O(n|\Sigma|^2 T^4 / \varepsilon^2 \mathfrak{s}_n(\mathbf{H})^4)$, then

$$\sum_{|x| \leqslant T} |f_A(x) - f_{\hat{A}}(x)| \leqslant \varepsilon$$

Proofs can be found in [Hsu, Kakade, and Zhang '09, Bailly '11, Balle '13]

Practical Considerations

Basic Setup:

- Data are strings sampled from probability distribution on $\Sigma^\star$
- Hankel matrix is estimated by empiricial probabilities
- Factorization and low-rank approximation is computed using SVD

Advanced Implementations:

- Choice of parameters $\mathcal{P}$ and $\mathcal{S}$
- Scalable estimation and factorization of Hankel matrices
- Smoothing and variance normalization
- Use of prefix and substring statistics

Choosing the Basis

Definition: The pair $(\mathcal{P}, \mathcal{S})$ defining the sub-block is called a *basis*

Intuitions:

- Basis should be choosen such that $\mathbb{E}[\mathbf{H}]$ has full rank
- $\mathcal{P}$ must contain strings reaching each possible state of the WFA
- $\mathcal{S}$ must contain string producing different outcomes for each pair of states in the WFA

Popular Approaches:

- Set $\mathcal{P} = \mathcal{S} = \Sigma^{\leqslant k}$ for some $k \geqslant 1$ [Hsu, Kakade, and Zhang '09]
- Choose $\mathcal{P}$ and $\mathcal{S}$ to contain the K most frequent prefixes and suffixes in the sample [Balle, Quattoni, and Carreras '12]
- Take all prefixes and suffixes appearing in the sample [Bailly, Denis, and Ralaivola '09]

Scalable Implementations

Problem: When $|\Sigma|$ is large, even the simplest basis become huge

Hankel Matrix Representation:

- Use hash functions to map $\mathcal{P}$ ($\mathcal{S}$) to row (column) indices
- Use sparse matrix data structures because statistics are usually sparse
- Never store the full Hankel matrix in memory

Efficient SVD Computation:

- SVD for sparse matrices [Berry '92]
- Approximate randomized SVD [Halko, Matrinsson, and Tropp '11]
- On-line SVD with rank 1 updates [Brand '06]

Refining the Statistics in the Hankel Matrix

Smoothing the Estimates

- Empirical probabilities $\hat{f}_S(x)$ tend to be sparse
- Like in n-gram models, smoothing can help when Σ is large
- Should take into account that strings in $\mathcal{PS}$ have different lengths
- Open Problem: How to smooth empirical Hankels properly

Row and Column Weighting

- More frequent prefixes (suffixes) have better estimated rows (columns)
- Can scale rows and columns to reflect that
- Will lead to more reliable SVD decompositions
- See [Cohen, Stratos, Collins, Foster, and Ungar '13] for details

Substring Statistics

Problem: If the sample contains strings with wide range of lengths, small basis will ignore most of the examples

String Statistics (occurence probability):

$$S = \left\{ \begin{array}{c} aa,\ b,\ bab,\ a, \\ bbab,\ abb,\ babba,\ abbb, \\ ab,\ a,\ aabba,\ baa, \\ abbab,\ baba,\ bb,\ a \end{array} \right\} \longrightarrow \hat{\mathbf{H}} = \begin{array}{c} \\ \epsilon \\ a \\ b \\ ba \end{array} \begin{array}{cc} a & b \\ \hline .19 & .06 \\ .06 & .06 \\ .00 & .06 \\ .06 & .06 \end{array}$$

Substring Statistics (expected number of occurences as substring):

$$\text{Empirical expectation} = \frac{1}{N}\sum_{i=1}^{N}[\text{number of occurences of } x \text{ in } x^i]$$

$$S = \left\{ \begin{array}{c} aa,\ b,\ bab,\ a, \\ bbab,\ abb,\ babba,\ abbb, \\ ab,\ a,\ aabba,\ baa, \\ abbab,\ baba,\ bb,\ a \end{array} \right\} \longrightarrow \hat{\mathbf{H}} = \begin{array}{c} \\ \epsilon \\ a \\ b \\ ba \end{array} \begin{array}{cc} a & b \\ \hline 1.31 & 1.56 \\ .19 & .62 \\ .56 & .50 \\ .06 & .31 \end{array}$$

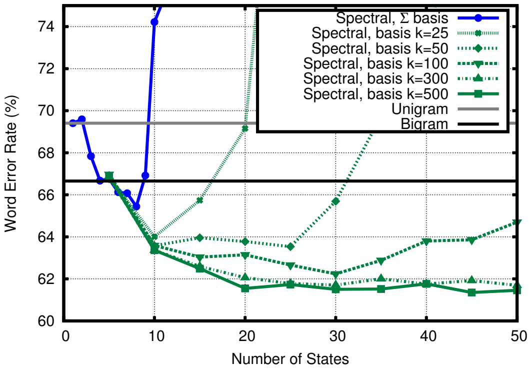
Substring Statistics

Theorem [Balle, Carreras, Luque, and Quattoni '14]
If a probability distribution f is computed by a WFA with n states, then the corresponding substring statistics are also computed by a WFA with n states

Learning from Substring Statistics

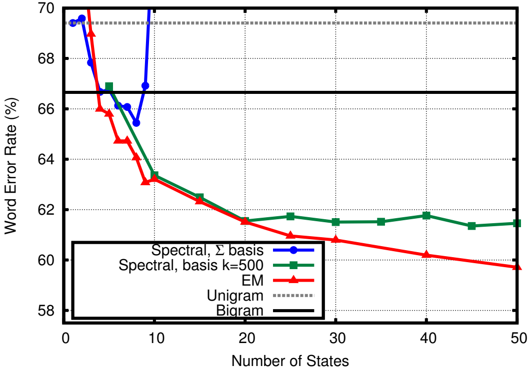
- Can work with smaller Hankel matrices
- But estimating the matrix takes longer

Experiment: PoS-tag Sequence Models



- PTB sequences of simplified PoS tags [Petrov, Das, and McDonald 2012]
- Configuration: expectations on frequent substrings
- Metric: error rate on predicting next symbol in test sequences

Experiment: PoS-tag Sequence Models



- Comparison with a bigram baseline and EM
- Metric: error rate on predicting next symbol in test sequences
- At training, the Spectral Method is > 100 faster than EM

Outline

Sequence Tagging and Transduction

- Many applications involve pairs of input-output sequences:
  - Sequence tagging (one output tag per input token)

    e.g.: part of speech tagging

output:	NNP	NNP	VBZ	NNP	.
input:	Ms.	Haag	plays	Elianti	.

  - Transductions (sequence lenghts might differ)

    e.g.: spelling correction

    output: a p p l e

    input: a p l e

- Finite-state automata are classic methods to model these relations. Spectral methods apply naturally to this setting.

Sequence Tagging

- Notation:
  - Input alphabet $\mathcal{X}$
  - Output alphabet $\mathcal{Y}$
  - Joint alphabet $\Sigma = \mathcal{X} \times \mathcal{Y}$
- Goal: map input sequences to output sequences of the same length
- Approach: learn a function

$$f : (\mathcal{X} \times \mathcal{Y})^\star \to \mathbb{R}$$

Then, given an input $x \in \mathcal{X}^T$ return

$$\operatorname*{argmax}_{y \in \mathcal{Y}^T} f(x, y)$$

(note: this maximization is not tractable in general)

Weighted Finite Tagger

- Notation:
  - $\mathcal{X} \times \mathcal{Y}$: joint alphabet – finite set
  - n: number of states – positive integer
  - $\boldsymbol{\alpha}_0$: initial weights – vector in $\mathbb{R}^n$ (features of empty prefix)
  - $\boldsymbol{\alpha}_\infty$: final weights – vector in $\mathbb{R}^n$ (features of empty suffix)
  - $\mathbf{A}_a^b$: transition weights – matrix in $\mathbb{R}^{n \times n}$ ($\forall a \in \mathcal{X}, b \in \mathcal{Y}$)
- Definition: WFTagger with n states over $\mathcal{X} \times \mathcal{Y}$

$$A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_a^b\} \rangle$$

- Compositional Function: Every WFTagger defines a function $f_A : (\mathcal{X} \times \mathcal{Y})^\star \to \mathbb{R}$

$$f_A(x_1 \dots x_T, y_1 \dots y_T) = \boldsymbol{\alpha}_0^\top \mathbf{A}_{x_1}^{y_1} \cdots \mathbf{A}_{x_T}^{y_T} \boldsymbol{\alpha}_\infty = \boldsymbol{\alpha}_0^\top \mathbf{A}_x^y \boldsymbol{\alpha}_\infty$$

The Spectral Method for WFTaggers

- Assume $f(x, y) = \mathbb{P}(x, y)$
  - Same mechanics as for WFA, with $\Sigma = \mathcal{X} \times \mathcal{Y}$
  - In a nutshell:
    1. Choose set of prefixes and suffixes to define Hankel
       $\rightarrow$ in this case they are bistrings
    2. Estimate Hankel with prefix-suffix training statistics
    3. Factorize Hankel using SVD
    4. Compute α and β projections,
       and compute operators $\langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\} \rangle$
- Other cases:
  - $f_A(x, y) = \mathbb{P}(y \mid x)$ — see [Balle et al., 2011]
  - $f_A(x, y)$ non-probabilistic — see [Quattoni et al., 2014]

Prediction with WFTaggers

- Assume $f_A(x, y) = \mathbb{P}(x, y)$
- Given $x_{1:T}$, compute most likely output tag at position t:

$$\operatorname*{argmax}_{a \in \mathcal{Y}} \mu(t, a)$$

where

$$\begin{aligned}
\mu(t, a) \triangleq \mathbb{P}(y_t = a \mid x) &= \sum_{y = y_1 \ldots a \ldots y_T} \mathbb{P}(x, y) \\
&= \sum_{y = y_1 \ldots a \ldots y_T} \boldsymbol{\alpha}_0^\top \mathbf{A}_x^y \boldsymbol{\alpha}_\infty \\
&= \underbrace{\boldsymbol{\alpha}_0^\top \left(\sum_{y_1 \ldots y_{t-1}} \mathbf{A}_{x_{1:t-1}}^{y_{1:t-1}} \right)}_{\alpha_A^\star(x_{1:t-1})} A_{x_t}^a \underbrace{\left(\sum_{y_{t+1} \ldots y_T} \mathbf{A}_{x_{t+1:T}}^{y_{i+1:T}} \right) \boldsymbol{\alpha}_\infty}_{\beta_A^\star(x_{t+1:T})}
\end{aligned}$$

$$\alpha_A^\star(x_{1:t}) = \alpha_A^\star(x_{1:t-1}) \left(\sum_{b \in \mathcal{Y}} \mathbf{A}_{x_t}^b \right) \qquad \beta_A^\star(x_{t:T}) = \left(\sum_{b \in \mathcal{Y}} \mathbf{A}_{x_t}^b \right) \beta_A^\star(x_{t+1:T})$$

Prediction with WFTaggers (II)

- Assume $f_A(x, y) = \mathbb{P}(x, y)$
- Given $x_{1:T}$, compute most likely output bigram ab at position t:

$$\operatorname*{argmax}_{a,b \in \mathcal{Y}} \mu(t, a, b)$$

where

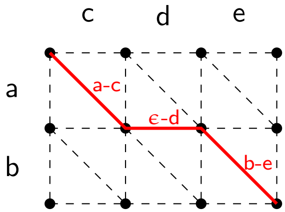
$$\begin{aligned} \mu(t, a, b) &= \mathbb{P}(y_t = a, y_{t+1} = b \mid x) \\ &= \alpha^\star_A(x_{1:t-1}) \mathbf{A}^a_{x_t} \mathbf{A}^b_{x_{t+1}} \beta^\star_A(x_{t+2:T}) \end{aligned}$$

- Compute most likely full sequence y – intractable
  In practice, use Minimum Bayes-Risk decoding:

$$\operatorname*{argmax}_{y \in \mathcal{Y}^T} \sum_t \mu(t, y_t, y_{t+1})$$

Finite State Transducers

(ab,cde)



- A WFTransducer evaluates aligned strings, using the empty symbol ϵ to produce one-to-one alignments:

$$f\binom{c\ d\ e}{a\ \epsilon\ b} = \boldsymbol{\alpha}_0^\top \mathbf{A}_a^c \mathbf{A}_\epsilon^d \mathbf{A}_b^e$$

- Then, a function can be defined on unaligned strings by aggregating alignments

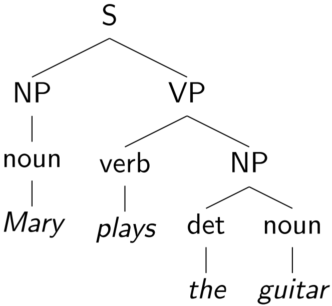
$$g(ab, cde) = \sum_{\pi \in \Pi(ab, cde)} f(\pi)$$

Finite State Transducers: Main Problems

- Inference: given an FST A, how to ...
  - Compute $g(x, y)$ for unaligned strings?
    $\rightarrow$ using edit-distance recursions
  - Compute marginal quantities $\mu(edge) = \mathbb{P}(edge \mid x)$?
    $\rightarrow$ also using edit-distance recursions
  - Compute most-likely y for given x?
    $\rightarrow$ use MBR-decoding with marginal scores

- Unsupervised Learning: learn an FST from pairs of unaligned strings
  - Unlike with EM, the spectral method can not recover latent structure such as alignments
    (recall: alignments are needed to estimate Hankel entries)
  - See [Bailly et al., 2013b] for a solution based on Hankel matrix completion

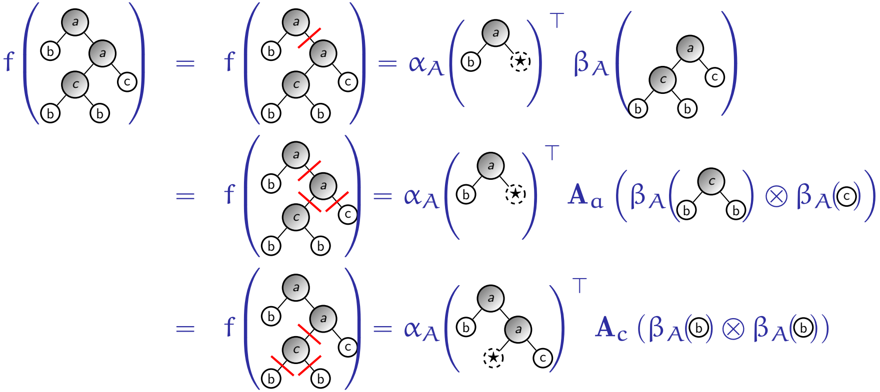
Spectral Learning of Tree Automata and Grammars



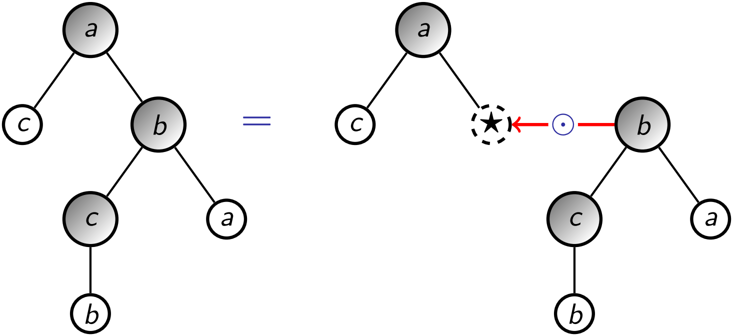
Some References:

- Tree Series: [Bailly et al., 2010, Bailly et al., 2010]
- Latent-annotated PCFG: [Cohen et al., 2012, Cohen et al., 2013b]
- Dependency parsing: [Luque et al., 2012, Dhillon et al., 2012]
- Unsupervised learning of WCFG: [Bailly et al., 2013a, Parikh et al., 2014]
- Synchronous grammars: [Saluja et al., 2014]

Compositional Functions over Trees



Inside-Outside Composition of Trees



$$t = t_o \odot t_i$$

note: i-o composition generalizes the notion of concatenation in strings,
i.e., outside trees are prefixes, inside trees are suffixes

Weighted Finite Tree Automata (WFTA)

An algebraic model for compositional functions on trees

WFTA Notation (I)

Labeled Trees

- $\{\Sigma^k\} = \{\Sigma^0, \Sigma^1, \ldots, \Sigma^r\}$ – ranked alphabet
- $\mathcal{T}$ – space of labeled trees over some ranked alphabet

Tree:

- $t \in \mathcal{T} = \langle V, E, l(v) \rangle$: a labeled tree
- $V = \{1, \ldots, m\}$: the set of vertices
- $E = \{\langle i, j \rangle\}$: the set of edges forming a tree
- $l(v) \rightarrow \{\Sigma^k\}$: returns the label of v – (i.e. a symbol in $\{\Sigma^k\}$)

WFTA Notation (II)

Labeled Trees

- $\{\Sigma^k\} = \{\Sigma^0, \Sigma^1, \ldots, \Sigma^r\}$ – ranked alphabet
- $\mathcal{T}$ – space of labeled trees over some ranked alphabet

Leaf Trees and Inside Compositions:

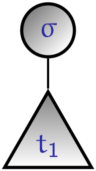
leaf tree

$\sigma \in \Sigma^0$

$t = \sigma$

unary composition

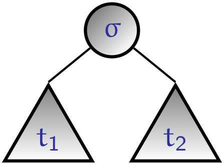
$\sigma \in \Sigma^1,\ t_1 \in \mathcal{T}$



$t = \sigma[t_1]$

binary composition

$\sigma \in \Sigma^2,\ t_1, t_2 \in \mathcal{T}$



$t = \sigma[t_1, t_2]$

WFTA Notation (III)

Labeled Trees

- $\{\Sigma^k\} = \{\Sigma^0, \Sigma^1, \ldots, \Sigma^r\}$ – ranked alphabet
- $\mathcal{T}$ – space of labeled trees over some ranked alphabet

Useful functions (to access the nodes of a tree t):

- $r(t)$: returns the root node of t
- $p(t, v)$: returns the parent of v
- $a(t, v)$: returns the arity of v (number of children of v)
- $c(t, v)$: returns the children of v
  - if $c(t, v) = [v_1, \ldots v_k]$ we use $c_i(t, v)$ for i-th child
  - children are assumed to be ordered from left to right

Notation for WFTA (IV): Tensors

Kronecker product:

- for $v_1 \in \mathbb{R}^n$ and $v_2 \in \mathbb{R}^n$:
- $v_1 \otimes v_2 \in \mathbb{R}^{n^2}$ contains all products between elements of v_1 and v_2
- Example:
  - $v_1 = [a, b]$
  - $v_2 = [c, d]$
  - $v_1 \otimes v_2 = [ac, ad, bc, bd]$

Simplifying assumption:

- We consider trees with $a(t, v) \leqslant 2$
  $\rightarrow$ i.e. tensors of order 3 (two children per parent)

Weighted Finite Tree Automata (WFTA)

$\Sigma = \{\Sigma^0, \Sigma^1, \Sigma^2\}$: ranked alphabet of order 2 – finite set

Definition: WFTA with n states over Σ

$$A = \langle \boldsymbol{\alpha}_\star, \{\boldsymbol{\beta}_\sigma\}, \{\mathbf{A}_\sigma^1\}, \{\mathbf{A}_\sigma^2\} \rangle$$

- n: number of states – positive integer
- $\boldsymbol{\alpha}_\star \in \mathbb{R}^n$: root weights
- $\boldsymbol{\beta}_\sigma \in \mathbb{R}^n$: leaf weights – $(\forall \sigma \in \Sigma^0)$
- $\mathbf{A}_\sigma^1 \in \mathbb{R}^{n \times n}$: transition weights – $(\forall \sigma \in \Sigma^1)$
- $\mathbf{A}_\sigma^2 \in R^{n \times n^2}$: transition weights – $(\forall \sigma \in \Sigma^2)$
- Note: $\mathbf{A}_\sigma^2$ is a tensor in $\mathbb{R}^{n \times n \times n}$ packed as a matrix

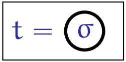
WFTA: Inside Function

Definition: Any WFTA $\mathcal{A}$ defines an inside function:

$$\beta_{\mathcal{A}} : \mathcal{T} \rightarrow \mathbb{R}^n$$ – maps a tree to a vector in $\mathbb{R}^n$

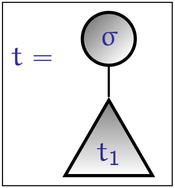
- if t is a leaf:

  $\beta_{\mathcal{A}}(t = \sigma) = \boldsymbol{\beta}_{\sigma}$



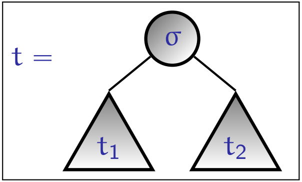
- if t results from a unary composition:

  $\beta_{\mathcal{A}}(t = \sigma[t_1]) = \mathbf{A}_{\sigma}^{1} \beta_{\mathcal{A}}(t_1)$



- if t results from a binary composition:

  $\beta_{\mathcal{A}}(t = \sigma[t_1, t_2]) = \mathbf{A}_{\sigma}^{2} \left(\beta_{\mathcal{A}}(t_1) \otimes \beta_{\mathcal{A}}(t_2)\right)$



WFTA Function:

Every WFTA A defines a function

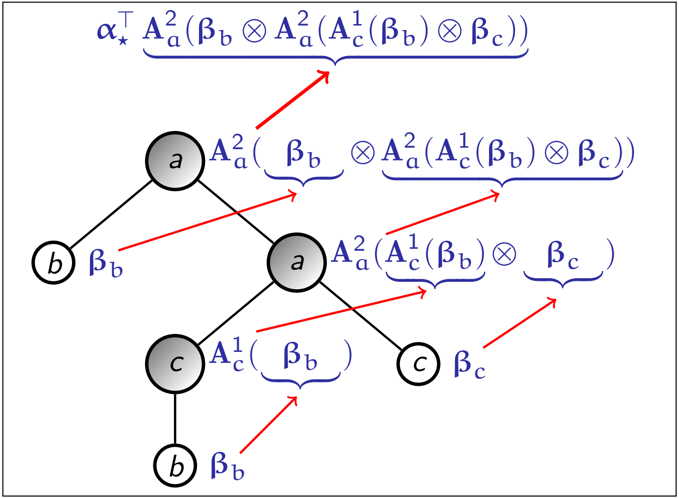
$$f_A : \mathcal{T} \rightarrow \mathbb{R}$$

computed as:

$$f_A(t) = \boldsymbol{\alpha}_\star^\top \beta_A(t)$$

Weighted Finite Tree Automaton (WFTA)

Example of inside computation:



Useful Intuition: Latent-variable Models as WFTA

$$f_A(t) = \boldsymbol{\alpha}_\star^t \beta_A(t)$$

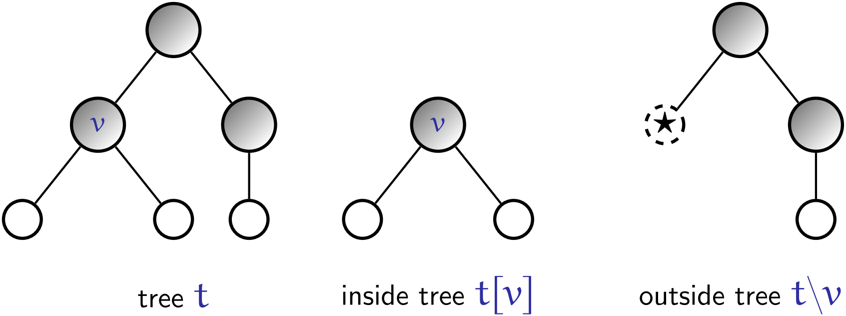
- Each labeled node v is *decorated* with a latent variable $h_v \in [n]$
- $f_A(t)$ is a sum–product computation:

$$\sum_{h_0, h_1, \ldots, h_{|V|} \in [n]} \Bigg(\boldsymbol{\alpha}_\star(h_0) \prod_{v \in V : a(v)=0} \boldsymbol{\beta}_{l(v)}[h_v] \times \prod_{v \in V : a(v)=1} \mathbf{A}^1_{l(v)}[h_v, h_{c(t,v)}] \times \prod_{v \in V : a(v)=2} \mathbf{A}^2_{l(v)}[h_v, h_{c_1(t,v)}, h_{c_2(t,v)}] \Bigg)$$

- $f_A(t)$ is a linear model in the latent space defined by $\beta_A : \mathcal{T} \to \mathbb{R}^n$

$$f_A(t) = \sum_{i=1}^{n} \boldsymbol{\alpha}_\star[i] \; \beta_A(t)[i]$$

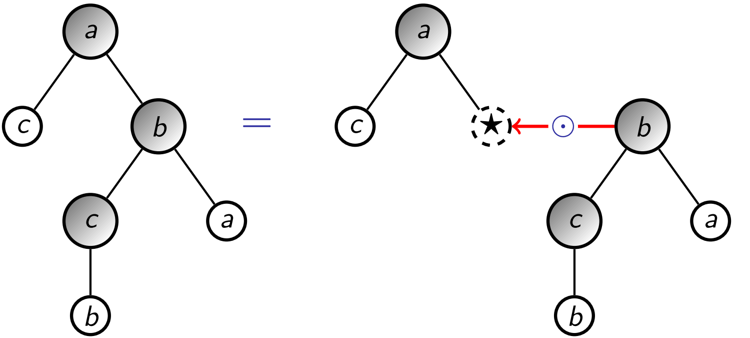
Inside/Outside Decomposition



Consider a tree t and one node v:

- Inside tree $t[v]$: the subtree of t rooted at v
  - $t[v] \in \mathcal{T}$
- Outside tree $t \backslash v$: the rest of t when removing $t[v]$
  - $\mathcal{T}_\star$: the space of outside trees, i.e. $t \backslash v \in \mathcal{T}_\star$
  - Foot node $\star$: a tree insertion point (a special symbol $\star \notin \{\Sigma^k\}$)
  - An outside tree has exactly one foot node in the leaves

Inside/Outside Composition

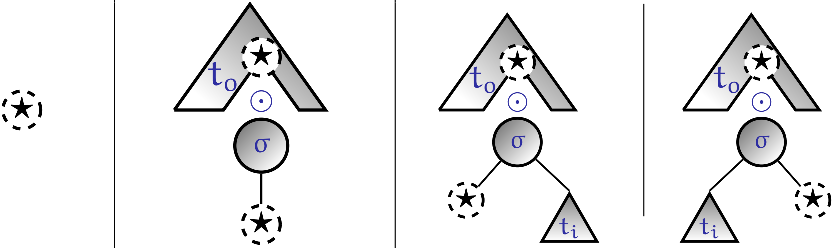


- A tree is formed by composing an outside tree with an inside tree
  $\longrightarrow$ generalizes prefix/suffix concatenation in strings
- Multiple ways to decompose a full tree into inside/outside trees
  $\longrightarrow$ as many as nodes in a tree

Outside Trees

- Outside trees $t_\star \in \mathcal{T}_\star$ are defined recursively using compositions:

foot node	unary composition	binary composition	
	$t_o \in \mathcal{T}_\star,\ \sigma \in \Sigma^1$	$t_o \in \mathcal{T}_\star,\ \sigma \in \Sigma^2,\ t_i \in \mathcal{T}$	



$t_\star = \star$	$t_\star = t_o \odot \sigma[\star]$	$t_\star = t_o \odot \sigma[\star, t_i]$	$t_\star = t_o \odot \sigma[t_i, \star]$

WFTA: Outside Function

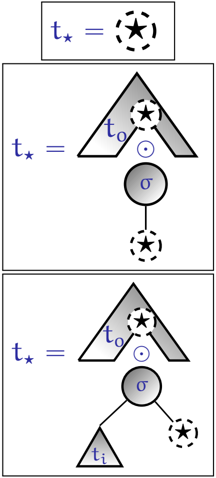
Definition: Any WFTA A defines an outside function:

$$\alpha_A : \mathcal{T}_\star \to \mathbb{R}^n$$ – maps an outside tree to a vector in $\mathbb{R}^n$

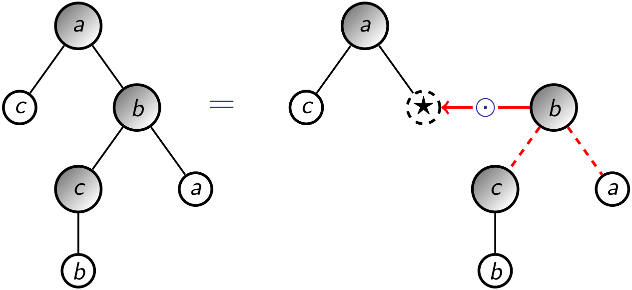
- if $t_\star$ is a foot node:
  $\alpha_A(t_\star = \star) = \boldsymbol{\alpha}_0$
- if $t_\star$ results from a unary composition:
  $\alpha_A(t_\star = t_o \odot \sigma[\star]) = \alpha_A(t_o)^\top \mathbf{A}_\sigma^1$
- if $t_\star$ results from a binary composition:

$$\alpha_A(t_\star = t_o \odot \sigma[t_i, \star]) = \alpha_A(t_o)^\top \mathbf{A}_\sigma^2 (\beta_A(t_i) \otimes \mathbf{1}^n)$$

(note: similar expression for $t_\star = t_o \odot \sigma[\star, t_i]$)



WFTA are fully compositional



For any inside-outside decomposition of a tree:

$$f_A(t) = \alpha_A(t_o)^\top \beta_A(t_i) \qquad \text{(let } t = t_o \odot t_i\text{)}$$

$$= \alpha_A(t_o)^\top \mathbf{A}_\sigma^2(\beta_A(t_1) \otimes \beta_A(t_2)) \qquad \text{(let } t_i = \sigma[t_1, t_2]\text{)}$$

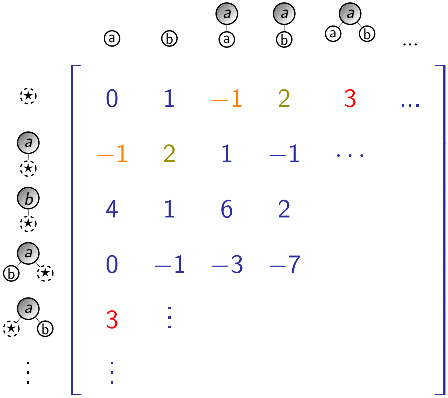
Consequences:

- We can isolate the α_A and β_A vector spaces
- Given α_A and β_A, we can isolate the operators $\mathbf{A}_\sigma^k$

Hankel Matrices of functions over Labeled Trees

Two Equivalent Representations

- Functional: $f_A : \mathcal{T} \to \mathbb{R}$
- Matricial: $H_{f_A} \in \mathbb{R}^{|\mathcal{T}_\star| \times |\mathcal{T}|}$ *(the Hankel matrix of* f_A*)*



- Definition:
  $H(t_o, t_i) = f(t_o \odot t_i)$
- Sublock for σ:
  $H_\sigma(t_o, \sigma[t_1, t_2]) = f(t_o \odot \sigma[t_1, t_2])$
- Properties:
  - $|V| + 1$ entries for $f(t)$
  - Depends on ordering of $\mathcal{T}_\star$ and $\mathcal{T}$
  - Captures structure

A Fundamental Theorem about WFTA

Relates the rank of $\mathbf{H}_f$

and the number of states of WFTA computing f

Let $f : \mathcal{T} \to \mathbb{R}$ be any function over labeled trees.

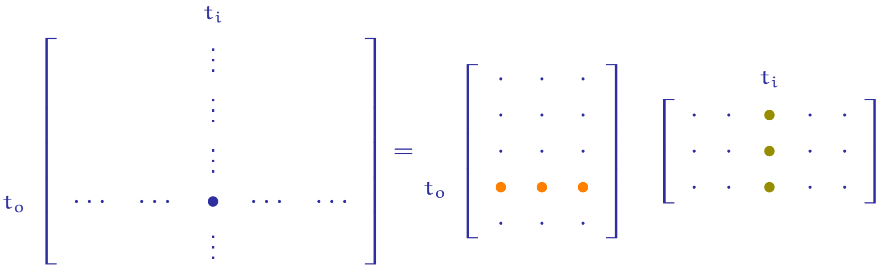
1. If $f = f_A$ for some WFTA A with n states $\Rightarrow \text{rank}(\mathbf{H}_f) \leqslant n$
2. If $\text{rank}(\mathbf{H}_f) = n \Rightarrow$ exists WFTA A with n states s.t. $f = f_A$

Why Fundamental?

Proof of (2) gives an algorithm for "recovering" A from the Hankel matrix of f_A

Structure of Low-rank Hankel Matrices

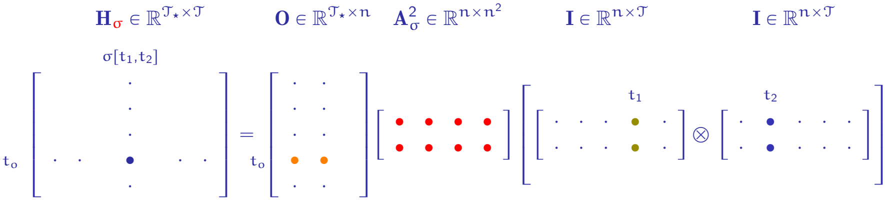
$\mathbf{H}_f \in \mathbb{R}^{\mathcal{T}_\star \times \mathcal{T}}$ $\mathbf{O} \in \mathbb{R}^{\mathcal{T}_\star \times n}$ $\mathbf{I} \in \mathbb{R}^{n \times \mathcal{T}}$



$$f(t_o \odot t_i) = \alpha_A(t_o)^\top \beta_A(t_i)$$

$$\alpha_A(t_o) = \mathbf{O}(t_o, \cdot) \qquad \beta_A(t_i) = \mathbf{I}(\cdot, t_i)$$

Hankel Factorizations and Operators



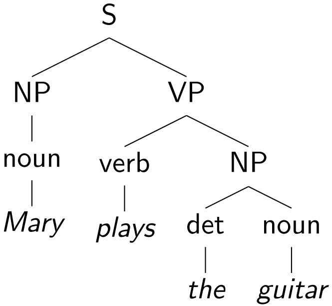
$$f(t_o \odot \underbrace{\sigma[t_1, t_2]}_{t_i}) = \alpha_A(t_o)^\top \underbrace{\mathbf{A}_\sigma^2(\beta_A(t_1) \otimes \beta_A(t_2))}_{\beta_A(t_i)}$$

Hankel Factorizations and Operators

$$\mathbf{H}_\sigma = \mathbf{O}\ \mathbf{A}_\sigma^2\ [\mathbf{I} \otimes \mathbf{I}] \quad \Longrightarrow \quad \mathbf{A}_\sigma^2 = \mathbf{O}^+\ \mathbf{H}_\sigma\ [\mathbf{I} \otimes \mathbf{I}]^+$$

Note: Works with finite sub-blocks as well
(assuming $\text{rank}(\mathbf{O}) = \text{rank}(\mathbf{I}) = n$)

WFTA: Application to Parsing



Some intuitions:

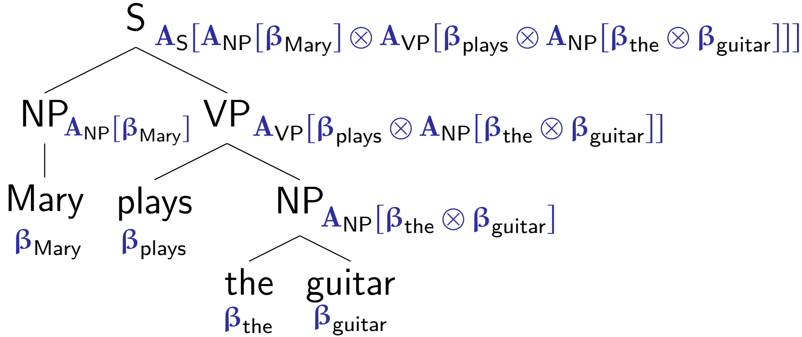
- Derivation = Labeled Tree
- Learning compositional functions over derivations
  $\Longrightarrow$ learning functions over trees
- We are interested in functions computed by WFTA

WFTA for Parsing: Key Questions

- What is the latent state representing?
  - For example: latent real valued embeddings of words and phrases
- What form of supervision do we get?
  - Full Derivations (labeled trees)
    i.e., supervised learning of latent-variable grammars
  - Derivation skeletons (unlabeled trees)
    e.g. [Pereira and Schabes, 1992]
  - Yields from the grammar (only the leaves)
    i.e., grammar induction

Parsing and Tree Automaton

$$\boldsymbol{\alpha}_0^\top \mathbf{A}_{\mathsf{S}}[\mathbf{A}_{\mathsf{NP}}[\boldsymbol{\beta}_{\mathsf{Mary}}] \otimes \mathbf{A}_{\mathsf{VP}}[\boldsymbol{\beta}_{\mathsf{plays}} \otimes \mathbf{A}_{\mathsf{NP}}[\boldsymbol{\beta}_{\mathsf{the}} \otimes \boldsymbol{\beta}_{\mathsf{guitar}}]]]$$



Phrase Embeddings using WFTA

Assume a WCFG in Chomsky Normal Form

- n – number of states; i.e. dimensionality of the embedding.
- Ranked alphabet:
  - $\Sigma^0 = \{$*the*, *Mary*, *plays*, $\ldots\}$ – terminal words
  - $\Sigma^1 = \{$noun, verb, det, NP, VP, $\ldots\}$ – unary non-terminals
  - $\Sigma^2 = \{$S, NP, VP, $\ldots\}$ – binary non-terminals
- $\boldsymbol{\alpha}_\star$ – final weights
- $\{\boldsymbol{\beta}_w\}$ for all $w \in \Sigma^0$ – word embeddings
- $\{\mathbf{A}^1_{N_1}\}$ for all $N_1 \in \Sigma^1$ – computes phrase embedding
- $\{\mathbf{A}^2_{N_2}\}$ for all $N_2 \in \Sigma^2$ – computes phrase embedding

Phrase Embeddings using WFTA

- $A = \langle \boldsymbol{\alpha}_\star, \{\boldsymbol{\beta}_w\}, \{\mathbf{A}^1_{N_1}\}, \{\mathbf{A}^2_{N_2}\} \rangle$ – WFTA
- $f_A(t) = \boldsymbol{\alpha}_\star^t \beta_A(t, S)$ – scores a derivation
- $\beta_A(t, S)$ – is the n-dimensional embedding of derivation t

Spectral Learning Algorithm for WFTA

Assume A is stochastic – i.e. it computes a distribution over derivations

- General Algorithm:
  - Chose a basis – i.e. a set of inside and outside trees
  - Estimate their empirical probabilities from a sample of derivations
  - Compute H and $\{\mathrm{H}_{\mathrm{N}}\}$
  - $\{\mathrm{H}_{\mathrm{N}}\}$ – one for each non-terminal
  - Perform SVD on H
  - Recover parameters of A using the WFTA Theorem
  - Note: We can also use sub-tree statistics

Spectral Learning of Tree Automata

- WFTA are a general algebraic framework for compositional functions
- WFTA can exploit real-valued embeddings
- There are simple algorithms for learning WFTAs from samples

Outline

1. Weighted Automata and Hankel Matrices
2. Spectral Learning of Probabilistic Automata
3. Spectral Methods for Transducers and Grammars
   - Sequence Tagging
   - Finite-State Transductions
   - Tree Automata
4. Hankel Matrices with Missing Entries
5. Conclusion
6. References

Learning WFA in More General Settings

Question: How do we use these approach to learn $f : \Sigma^\star \to \mathbb{R}$ where $f(x)$ does not have a probabilistic interpretation?

Examples:

- Classification $f : \Sigma^\star \to \{+1, -1\}$
- Unconstrained real-valued predictions $f : \Sigma^\star \to \mathbb{R}$
- General scoring functions for tagging: $f : (\Sigma \times \Delta)^\star \to \mathbb{R}$

Example: Hankel Matrices with Missing Entries

When learning probabilistic functions. . .

entries in $\mathbf{H}_f$ are estimated from empirical counts, e.g. $f(x) = \mathbb{P}[x]$

$$\left\{ \begin{array}{c} \text{aa, b, bab, a,} \\ \text{b, a, ab, aa,} \\ \text{ba, b, aa, a,} \\ \text{aa, bab, b, aa} \end{array} \right\} \longrightarrow \begin{array}{c} \\ \epsilon \\ a \\ b \\ ba \end{array} \begin{array}{c} \begin{array}{cc} a & b \end{array} \\ \left[\begin{array}{cc} .19 & .25 \\ .31 & .06 \\ .06 & .00 \\ .00 & .13 \end{array} \right] \end{array}$$

But in a general regression setting...

entries in $\mathbf{H}_f$ are labels observed in the sample, and many may be missing

$$\left\{ \begin{array}{c} (\text{bab},1) \\ (\text{bbb},0) \\ (\text{aaa},3) \\ (\text{a},1) \\ (\text{ab},1) \\ (\text{aa},2) \\ (\text{aba},2) \\ (\text{bb},0) \end{array} \right\} \longrightarrow \begin{array}{c} \\ a \\ b \\ aa \\ ab \\ ba \\ bb \end{array} \begin{array}{c} \begin{array}{ccc} \epsilon & a & b \end{array} \\ \left[\begin{array}{ccc} 1 & 2 & 1 \\ ? & ? & 0 \\ 2 & 3 & ? \\ 1 & 2 & ? \\ ? & ? & 1 \\ 0 & ? & 0 \end{array} \right] \end{array}$$

Inference of Hankel Matrices

Goal: Learn a Hankel matrix $\mathbf{H} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$ from partial information, *then apply the Hankel trick*

Information Models:

- Subset of entries: $\{\mathbf{H}(p, s) | (p, s) \in I\}$
- Linear measurements: $\{\mathbf{H}\mathbf{v} | \mathbf{v} \in V\}$
- Bilinear measurements: $\{\mathbf{u}^\top \mathbf{H}\mathbf{v} | \mathbf{u} \in U, \mathbf{v} \in V\}$
- Constraints between entries: $\{\mathbf{H}(p, s) \geqslant \mathbf{H}(p', s') | (p, s, p', s') \in I\}$
- *Noisy versions of all the above*

Constraints and Inductive Bias:

- Hankel constraints $\mathbf{H}(p, s) = \mathbf{H}(p', s')$ if $ps = p's'$
- Constraints on entries $|\mathbf{H}(p, s)| \leqslant C$
- Low-rank constraints/regularization $\mathrm{rank}(\mathbf{H})$

Empirical Risk Minimization Approach

Data: $\{(x^i, y^i)\}_{i=1}^N$, $x^i \in \Sigma^\star$, $y^i \in \mathbb{R}$

Parameters:

- Rows and columns $\mathcal{P}, \mathcal{S} \subset \Sigma^\star$
- (Convex) Loss function $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$
- Regularization parameter λ / rank bound R

Optimization (constrained formulation):

$$\operatorname*{argmin}_{\mathbf{H} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}} \frac{1}{N} \sum_{i=1}^{N} \ell(y^i, \mathbf{H}(x^i)) \text{ subject to } \operatorname{rank}(\mathbf{H}) \leqslant R$$

Optimization (regularized formulation):

$$\operatorname*{argmin}_{\mathbf{H} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}} \frac{1}{N} \sum_{i=1}^{N} \ell(y^i, \mathbf{H}(x^i)) + \lambda \operatorname{rank}(\mathbf{H})$$

Note: These optimization problems are *non-convex*!

Nuclear Norm Relaxation

Nuclear Norm: matrix $\mathbf{M}$, $\|\mathbf{M}\|_* = \sum \mathfrak{s}_i(\mathbf{M})$

In machine learning, minimizing the nuclear norm is a commonly used convex surrogate for minimizing the rank

Convex Optimization for Hankel matrix estimation

$$\operatorname*{argmin}_{H \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}} \frac{1}{N} \sum_{i=1}^{N} \ell(y^i, \mathbf{H}(x^i)) + \lambda \|\mathbf{H}\|_*$$

Optimization Algorithms for Hankel Matrix Estimation

Optimizing the Nuclear Norm Surrogate

- Projected/Proximal sub-gradient (e.g. [Duchi and Singer '09])
- Frank–Wolfe [Jaggi and Sulovsk '10]
- Singular value thresholding [Cai, Candès, and Shen '10]

Non-Convex "Heuristics"

- Alternating minimization (e.g. [Jain, Netrapalli, and Sanghavi '13])

Applications of Hankel Matrix Estimation

- Max-margin taggers [Quattoni, Balle, Carreras, and Globerson '14]
- Unsupervised transducers [Bailly, Quattoni, and Carreras '13]
- Unsupervised WCFG [Bailly, Carreras, Luque, Quattoni '13]

Outline

Conclusion

- Spectral methods provide new tools to learn compositional functions by means of algebraic operations

- Key result:
  forward-backward recursions $\Leftrightarrow$ low-rank Hankel matrices

- Applicable to a wide range of compositional formalisms: finite-state automata and transducers, context-free grammars, ...

- Relation to loss-regularized methods, by means of matrix-completion techniques

Spectral Learning Techniques for Weighted Automata, Transducers, and Grammars

Borja Balle[♢] Ariadna Quattoni[♡] Xavier Carreras[♡]

(♢) McGill University

(♡) Xerox Research Centre Europe

TUTORIAL @ EMNLP 2014

Outline

Anandkumar, A., Chaudhuri, K., Hsu, D., Kakade, S. M., Song, L., and Zhang, T. (2011).
Spectral methods for learning multivariate latent tree structure.
In *NIPS*.

Anandkumar, A., Foster, D. P., Hsu, D., Kakade, S. M., and Liu, Y. (2012a).
A spectral algorithm for latent Dirichlet allocation.
In *NIPS*.

Anandkumar, A., Ge, R., Hsu, D., and Kakade, S. (2013a).
A tensor spectral approach to learning mixed membership community models.
In *COLT*.

Anandkumar, A., Ge, R., Hsu, D., Kakade, S. M., and Telgarsky, M. (2012b).
Tensor decompositions for learning latent variable models.
CoRR, abs/1210.7559.

Anandkumar, A., Hsu, D., Huang, F., and Kakade, S. M. (2012c).
Learning mixtures of tree graphical models.

In *NIPS*.

Anandkumar, A., Hsu, D., and Kakade, S. M. (2012d).
A method of moments for mixture models and hidden markov models.
In *25th Annual Conference on Learning Theory*.

Anandkumar, A., Hsu, D., and Kakade, S. M. (2012e).
A method of moments for mixture models and hidden Markov models.
In *COLT*.

Anandkumar, A., Hsu, D., and Kakade, S. M. (2013b).
Tutorial: Tensor decomposition algorithms for latent variable model estimation.
In *ICML*.

Bailly, R. (2011).
Quadratic weighted automata: Spectral algorithm and likelihood maximization.
In *ACML*.

Bailly, R., Carreras, X., Luque, F., and Quattoni, A. (2013a).
Unsupervised spectral learning of WCFG as low-rank matrix completion.

In *EMNLP*.

Bailly, R., Carreras, X., and Quattoni, A. (2013b).
Unsupervised spectral learning of finite state transducers.
In *NIPS*.

Bailly, R., Denis, F., and Ralaivola, L. (2009).
Grammatical inference as a principal component analysis problem.
In *ICML*.

Bailly, R., Habrard, A., and Denis, F. (2010).
A spectral approach for probabilistic grammatical inference on trees.
In *ALT*.

Balle, B. (2013).
Learning Finite-State Machines: Algorithmic and Statistical Aspects.
PhD thesis, Universitat Politècnica de Catalunya.

Balle, B., Carreras, X., Luque, F., and Quattoni, A. (2013).
Spectral learning of weighted automata: A forward-backward perspective.
Machine Learning.

Balle, B. and Mohri, M. (2012).
Spectral learning of general weighted automata via constrained matrix completion.
In *NIPS*.

Balle, B., Quattoni, A., and Carreras, X. (2011).
A spectral learning algorithm for finite state transducers.
In *ECML-PKDD*.

Balle, B., Quattoni, A., and Carreras, X. (2012).
Local loss optimization in operator models: A new insight into spectral learning.
In *ICML*.

Berry, M. W. (1992).
Large-scale sparse singular value computations.

Boots, B. and Gordon, G. (2011).
An online spectral learning algorithm for partially observable dynamical systems.
In *Association for the Advancement of Artificial Intelligence*.

Boots, B., Gretton, A., and Gordon, G. (2013).
Hilbert space embeddings of predictive state representations.
In *UAI*.

Boots, B., Siddiqi, S., and Gordon, G. (2009).
Closing the learning-planning loop with predictive state representations.
In *Proceedings of Robotics: Science and Systems VI*.

Boots, B., Siddiqi, S., and Gordon, G. (2011).
Closing the learning planning loop with predictive state representations.
International Journal of Robotics Research.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011).
Distributed optimization and statistical learning via the alternating direction method of multipliers.
Foundations and Trends in Machine Learning.

Brand, M. (2006).
Fast low-rank modifications of the thin singular value decomposition.
Linear algebra and its applications, 415(1):20–30.

Carlyle, J. W. and Paz, A. (1971).
Realizations by stochastic finite automata.
Journal of Computer Systems Science.

Chaganty, A. T. and Liang, P. (2013).
Spectral experts for estimating mixtures of linear regressions.
In *ICML*.

Cohen, S. B., Collins, M., Foster, D. P., Stratos, K., and Ungar, L. (2013a).
Tutorial: Spectral learning algorithms for natural language processing.
In *NAACL*.

Cohen, S. B., Stratos, K., Collins, M., Foster, D. P., and Ungar, L. (2012).
Spectral learning of latent-variable PCFGs.
ACL.

Cohen, S. B., Stratos, K., Collins, M., Foster, D. P., and Ungar, L. (2013b).
Experiments with spectral learning of latent-variable PCFGs.
In *NAACL-HLT*.

Cohen, S. B., Stratos, K., Collins, M., Foster, D. P., and Ungar, L. (2014).
Spectral learning of latent-variable PCFGs: Algorithms and sample complexity.
Journal of Machine Learning Research.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977).
Maximum likelihood from incomplete data via the EM algorithm.
Journal of the Royal Statistical Society.

Denis, F. and Esposito, Y. (2008).
On rational stochastic languages.
Fundamenta Informaticae.

Dhillon, P. S., Rodu, J., Collins, M., Foster, D. P., and Ungar, L. H. (2012).
Spectral dependency parsing with latent variables.
In *EMNLP-CoNLL*.

Dupont, P., Denis, F., and Esposito, Y. (2005).
Links between probabilistic automata and hidden Markov models: probability distributions, learning models and induction algorithms.

Pattern Recognition.

Fliess, M. (1974).
Matrices de Hankel.
Journal de Mathématiques Pures et Appliquées.

Foster, D. P., Rodu, J., and Ungar, L. H. (2012).
Spectral dimensionality reduction for HMMs.
CoRR, abs/1203.6130.

Gordon, G. J., Song, L., and Boots, B. (2012).
Tutorial: Spectral approaches to learning latent variable models.
In *ICML*.

Halko, N., Martinsson, P., and Tropp, J. (2011).
Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions.
SIAM Review, 53(2):217–288.

Hamilton, W., Fard, M., and Pineau, J. (2013a).
Modelling sparse dynamical systems with compressed predictive state representations.

In *ICML*.

Hamilton, W. L., Fard, M. M., and Pineau, J. (2013b).
Modelling sparse dynamical systems with compressed predictive state representations.
In *Proceedings of the 30th International Conference on Machine Learning*.

Hsu, D. and Kakade, S. M. (2013).
Learning mixtures of spherical Gaussians: moment methods and spectral decompositions.
In *ITCS*.

Hsu, D., Kakade, S. M., and Zhang, T. (2009).
A spectral algorithm for learning hidden Markov models.
In *COLT*.

Hulden, M. (2012).
Treba: Efficient numerically stable EM for PFA.
In *ICGI*.

Jaeger, H. (2000).
Observable operator models for discrete stochastic time series.

Neural Computation, 12(6):1371–1398.

Littman, M., Sutton, R. S., and Singh, S. (2002).
Predictive representations of state.
In *Advances In Neural Information Processing Systems*.

Luque, F., Quattoni, A., Balle, B., and Carreras, X. (2012).
Spectral learning in non-deterministic dependency parsing.
In *EACL*.

Marcus, M., Marcinkiewicz, M., and Santorini, B. (1993).
Building a large annotated corpus of english: The Penn Treebank.
Computational linguistics, 19(2):313–330.

Parikh, A., Song, L., Ishteva, M., Teodoru, G., and Xing, E. (2012).
A spectral algorithm for latent junction trees.
In *UAI*.

Parikh, A. P., Cohen, S. B., and Xing, E. (2014).
Spectral unsupervised parsing with additive tree metrics.
In *Proceedings of ACL*.

Parikh, A. P., Song, L., and Xing, E. (2011).

A spectral algorithm for latent tree graphical models.
In *ICML*.

Pereira, F. and Schabes, Y. (1992).
Inside-outside reestimation from partially bracketed corpora.
In *Proceedings of the 30th annual meeting on Association for Computational Linguistics*, pages 128–135. Association for Computational Linguistics.

Quattoni, A., Balle, B., Carreras, X., and Globerson, A. (2014).
Spectral regularization for max-margin sequence tagging.
In *ICML*.

Rabiner, L. R. (1990).
A tutorial on hidden markov models and selected applications in speech recognition.
In Waibel, A. and Lee, K., editors, *Readings in speech recognition*, pages 267–296.

Recasens, A. and Quattoni, A. (2013).
Spectral learning of sequence taggers over continuous sequences.
In *ECML-PKDD*.

Rosencrantz, M., Gordon, G., and Thrun, S. (2004).
Learning low dimensional predictive representations.
In *Proceedings of the 21st International Conference on Machine learning*.

Saluja, A., Dyer, C., and Cohen, S. B. (2014).
Latent-variable synchronous CFGs for hierarchical translation.
Proceedings of EMNLP.

Siddiqi, S. M., Boots, B., and Gordon, G. (2010).
Reduced-rank hidden Markov models.
In *AISTATS*.

Singh, S., James, M., and Rudary, M. (2004).
Predictive state representations: a new theory for modeling dynamical systems.
In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*.

Song, L., Boots, B., Siddiqi, S., Gordon, G., and Smola, A. (2010).
Hilbert space embeddings of hidden Markov models.

In *ICML*.

Song, L., Ishteva, M., Parikh, A., Xing, E., and Park, H. (2013).
Hierarchical tensor decomposition of latent tree graphical models.
In *ICML*.

Stratos, K., Rush, A. M., Cohen, S. B., and Collins, M. (2013).
Spectral learning of refinement hmms.
In *CoNLL*.

Verwer, S., Eyraud, R., and Higuera, C. (2012).
Results of the PAutomaC probabilistic automaton learning competition.
In *ICGI*.

Wiewiora, E. (2007).
Modeling probability distributions with predictive state representations.
PhD thesis, University of California at San Diego.