Hankel-Based Methods for Latent-State Structured Prediction

Ariadna Quattoni
Xerox Research Centre Europe
Collaborators: R. Bailly, B. Balle, X.Carreras, A. Globerson, F. Luque, A. Mayo

LxMLS 2014

Structured Prediction

- We want to learn functions over structured objects:
  - from sequences to reals (e.g. sequence classification)
  - from pairs of sequences to reals (e.g. sequence tagging)
  - from pairs of sequences and trees to reals (e.g. NLP parsing)
- Stochastic Functions: $0 \leqslant f(x) \leqslant 1$ and $\sum_{x \in X} f(x) = 1$
- Use these models to make predictions: $\text{argmax}_x f(x)$

Latent State Models

- Consider a sequence $abaccb$

$$\begin{aligned} f(abaccb) &= \alpha_f(ab) \cdot \beta_f(accb) \\ &= \alpha_f(ab) \cdot A_a \cdot \beta_f(ccb) \\ &= \alpha_f(aba) \cdot A_c \cdot \beta_f(cb) \end{aligned}$$

  where:

  - n is the dimension of the model
  - α_f maps prefixes to $\mathbb{R}^n$
  - β_f maps suffixes to $\mathbb{R}^n$
  - A_a is a bilinear operator in $\mathbb{R}^{n \times n}$

- F : functions computed by WFAs (HMMs is a subclass)

Latent State Models

- Consider a sequence $abaccb$

$$\begin{aligned} f(abaccb) &= \alpha_f(ab) \cdot \beta_f(accb) \\ &= \alpha_f(ab) \cdot A_a \cdot \beta_f(ccb) \\ &= \alpha_f(aba) \cdot A_c \cdot \beta_f(cb) \end{aligned}$$

  where:
  - n is the dimension of the model
  - α_f maps prefixes to $\mathbb{R}^n$
  - β_f maps suffixes to $\mathbb{R}^n$
  - A_a is a bilinear operator in $\mathbb{R}^{n \times n}$
- F : functions computed by WFAs (HMMs is a subclass)

- Finding $f \in F$: Estimate α_f, β_f and A_σ from S
  - EM
- Many ways of expressing the same $f \in F$
  - Algebraic Methods

Outline

- Spectral Learning of WFA
- Hankel-Based Optimization for Structured Prediction

Outline

- Spectral Learning of WFA 
- Hankel-Based Optimization for Structured Prediction

Outline

- Spectral Learning of WFA → 'Classic spectral learning algorithms'
- Hankel-Based Optimization for Structured Prediction

Outline

- Spectral Learning of WFA → 'Classic spectral learning algorithms'
- Hankel-Based Optimization for Structured Prediction →

Outline

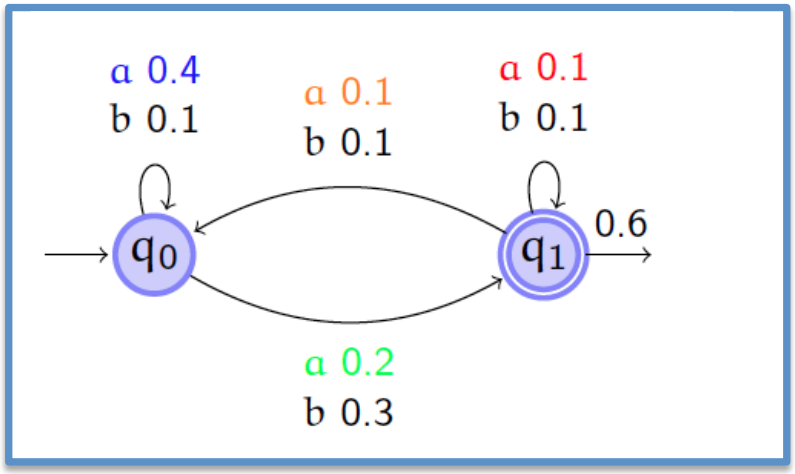
- Spectral Learning of WFA → 'Classic spectral learning algorithms'
- Hankel-Based Optimization for Structured Prediction → 'Different interpretation of spectral trick'

Outline

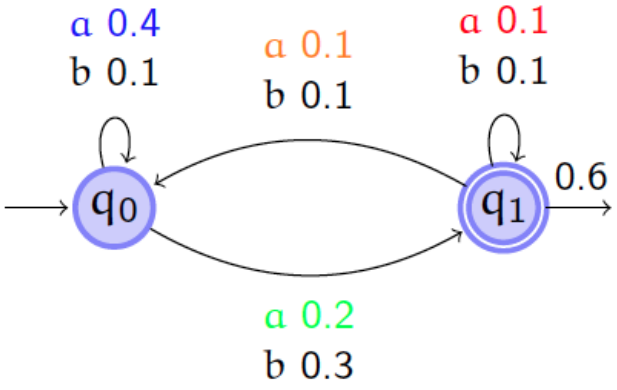
- **Spectral Learning of WFA**
- Hankel-Based Optimization for Structured Prediction

Weighted Automata (WFA)

Weighted Automata (WFA)



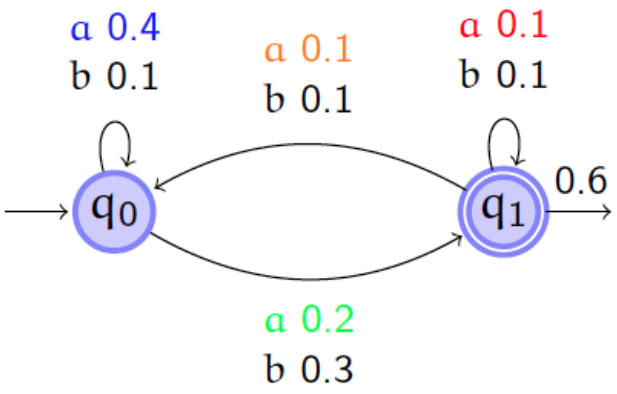
Weighted Automata (WFA)



$$\alpha_1 = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} \qquad A_a = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.1 \end{bmatrix}$$

$$\alpha_\infty = \begin{bmatrix} 0.0 \\ 0.6 \end{bmatrix} \qquad A_b = \begin{bmatrix} 0.1 & 0.3 \\ 0.1 & 0.1 \end{bmatrix}$$

Weighted Automata (WFA)

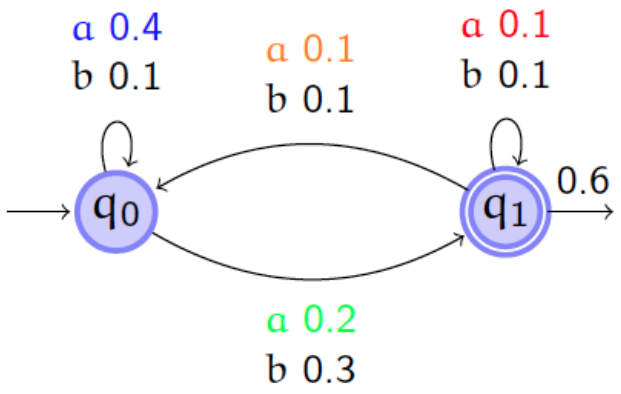


$$\alpha_1 = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} \quad A_a = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.1 \end{bmatrix}$$

$$\alpha_\infty = \begin{bmatrix} 0.0 \\ 0.6 \end{bmatrix} \quad A_b = \begin{bmatrix} 0.1 & 0.3 \\ 0.1 & 0.1 \end{bmatrix}$$

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

Weighted Automata (WFA)



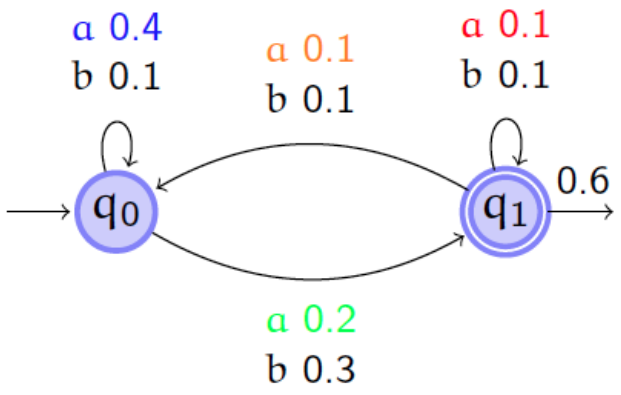
$$\alpha_1 = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} \quad A_a = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.1 \end{bmatrix}$$

$$\alpha_\infty = \begin{bmatrix} 0.0 \\ 0.6 \end{bmatrix} \quad A_b = \begin{bmatrix} 0.1 & 0.3 \\ 0.1 & 0.1 \end{bmatrix}$$

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

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

Weighted Automata (WFA)



$$\alpha_1 = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} \quad A_a = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.1 \end{bmatrix}$$

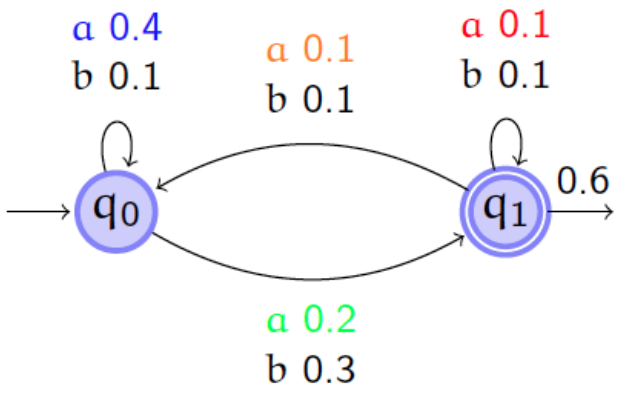
$$\alpha_\infty = \begin{bmatrix} 0.0 \\ 0.6 \end{bmatrix} \quad A_b = \begin{bmatrix} 0.1 & 0.3 \\ 0.1 & 0.1 \end{bmatrix}$$

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

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

$\alpha_1 \in \mathbb{R}^n$

Weighted Automata (WFA)



$$\alpha_1 = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} \quad A_a = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.1 \end{bmatrix}$$

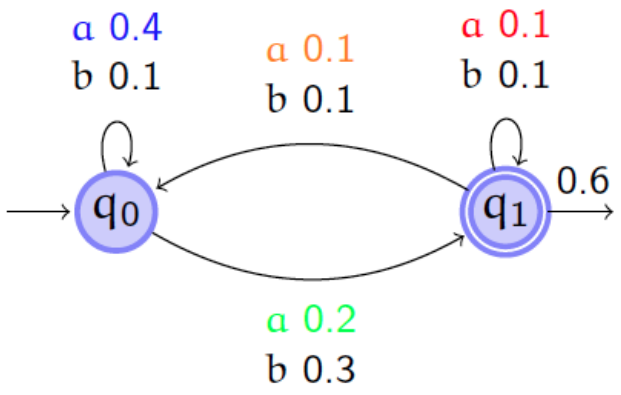
$$\alpha_\infty = \begin{bmatrix} 0.0 \\ 0.6 \end{bmatrix} \quad A_b = \begin{bmatrix} 0.1 & 0.3 \\ 0.1 & 0.1 \end{bmatrix}$$

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

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

$\alpha_1 \in \mathbb{R}^n$ ➡

Weighted Automata (WFA)



$$\alpha_1 = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} \quad A_a = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.1 \end{bmatrix}$$

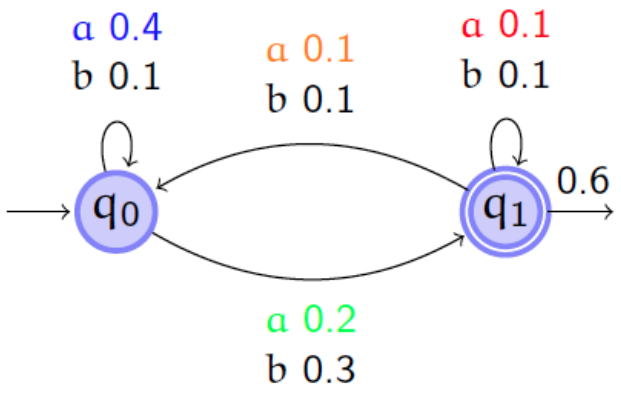
$$\alpha_\infty = \begin{bmatrix} 0.0 \\ 0.6 \end{bmatrix} \quad A_b = \begin{bmatrix} 0.1 & 0.3 \\ 0.1 & 0.1 \end{bmatrix}$$

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

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

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

Weighted Automata (WFA)



$$\alpha_1 = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} \quad A_a = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.1 \end{bmatrix}$$

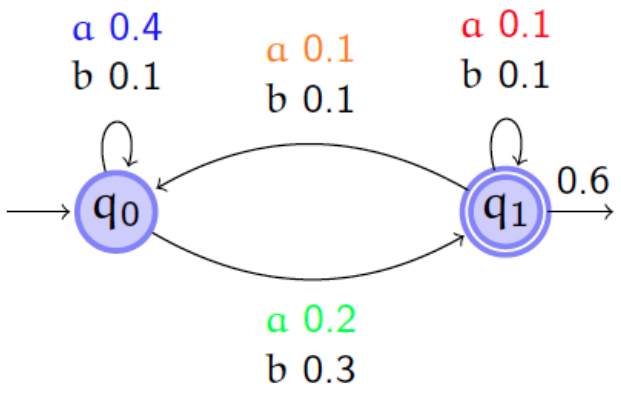
$$\alpha_\infty = \begin{bmatrix} 0.0 \\ 0.6 \end{bmatrix} \quad A_b = \begin{bmatrix} 0.1 & 0.3 \\ 0.1 & 0.1 \end{bmatrix}$$

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

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

$\alpha_1 \in \mathbb{R}^n$ ⇒ (initial weights) $\alpha_\infty \in \mathbb{R}^n$

Weighted Automata (WFA)



$$\alpha_1 = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} \quad A_a = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.1 \end{bmatrix}$$

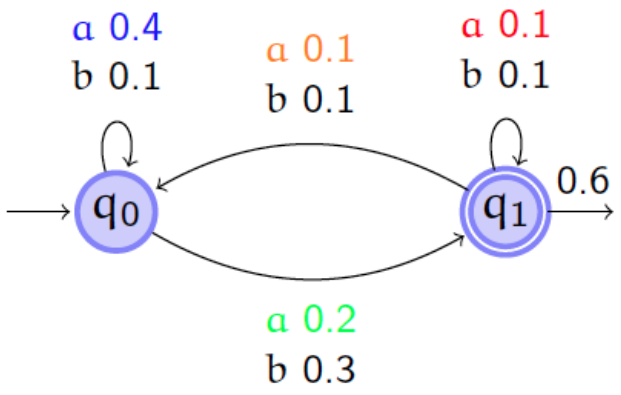
$$\alpha_\infty = \begin{bmatrix} 0.0 \\ 0.6 \end{bmatrix} \quad A_b = \begin{bmatrix} 0.1 & 0.3 \\ 0.1 & 0.1 \end{bmatrix}$$

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

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

$\alpha_1 \in \mathbb{R}^n$ ➡ (initial weights) $\alpha_\infty \in \mathbb{R}^n$ ➡

Weighted Automata (WFA)



$$\alpha_1 = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} \quad A_a = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.1 \end{bmatrix}$$

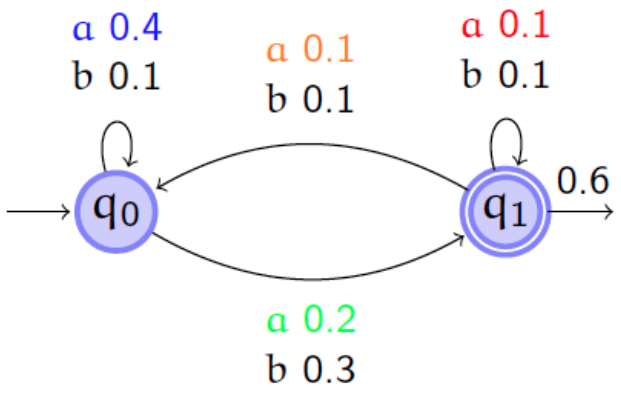
$$\alpha_\infty = \begin{bmatrix} 0.0 \\ 0.6 \end{bmatrix} \quad A_b = \begin{bmatrix} 0.1 & 0.3 \\ 0.1 & 0.1 \end{bmatrix}$$

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

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

$\alpha_1 \in \mathbb{R}^n$ ➡ (initial weights) $\alpha_\infty \in \mathbb{R}^n$ ➡ (terminal weights)

Weighted Automata (WFA)



$$\alpha_1 = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} \quad A_a = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.1 \end{bmatrix}$$

$$\alpha_\infty = \begin{bmatrix} 0.0 \\ 0.6 \end{bmatrix} \quad A_b = \begin{bmatrix} 0.1 & 0.3 \\ 0.1 & 0.1 \end{bmatrix}$$

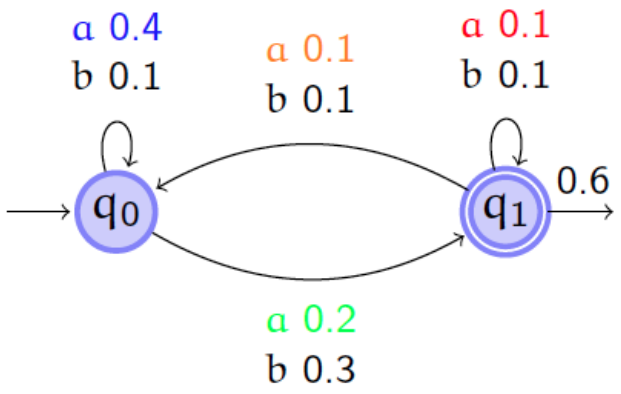
- Class of WA parametrized by alphabet Σ and number of states n

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

$\alpha_1 \in \mathbb{R}^n$ ➡ (initial weights) $\alpha_\infty \in \mathbb{R}^n$ ➡ (terminal weights)

$$A_\sigma \in \mathbb{R}^{n \times n}$$

Weighted Automata (WFA)



$$\alpha_1 = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} \quad A_a = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.1 \end{bmatrix}$$

$$\alpha_\infty = \begin{bmatrix} 0.0 \\ 0.6 \end{bmatrix} \quad A_b = \begin{bmatrix} 0.1 & 0.3 \\ 0.1 & 0.1 \end{bmatrix}$$

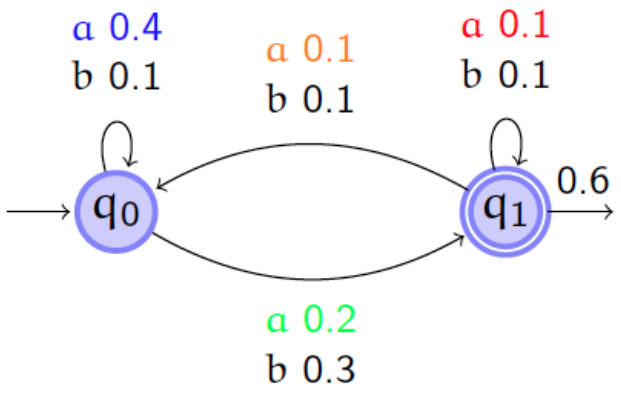
- Class of WA parametrized by alphabet Σ and number of states n

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

$\alpha_1 \in \mathbb{R}^n$ ➡ (initial weights) $\alpha_\infty \in \mathbb{R}^n$ ➡ (terminal weights)

$A_\sigma \in \mathbb{R}^{n \times n}$ ➡

Weighted Automata (WFA)



$$\alpha_1 = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} \quad A_a = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.1 \end{bmatrix}$$

$$\alpha_\infty = \begin{bmatrix} 0.0 \\ 0.6 \end{bmatrix} \quad A_b = \begin{bmatrix} 0.1 & 0.3 \\ 0.1 & 0.1 \end{bmatrix}$$

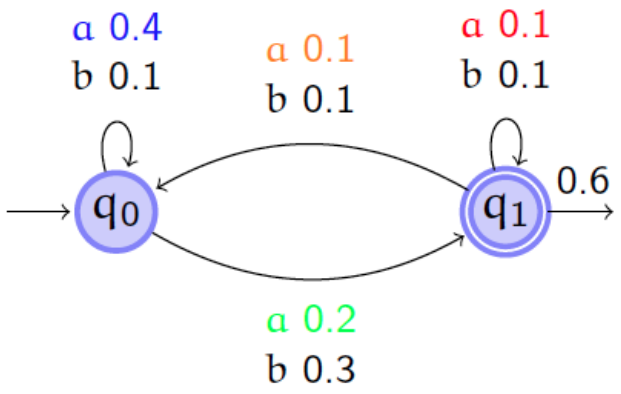
- Class of WA parametrized by alphabet Σ and number of states n

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

$\alpha_1 \in \mathbb{R}^n$ ➡ (initial weights) $\alpha_\infty \in \mathbb{R}^n$ ➡ (terminal weights)

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

Weighted Automata (WFA)



$$\alpha_1 = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} \quad A_a = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.1 \end{bmatrix}$$

$$\alpha_\infty = \begin{bmatrix} 0.0 \\ 0.6 \end{bmatrix} \quad A_b = \begin{bmatrix} 0.1 & 0.3 \\ 0.1 & 0.1 \end{bmatrix}$$

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

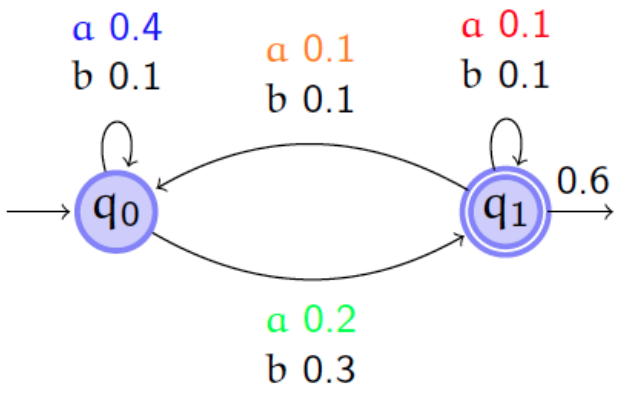
$$\mathbf{A} = \langle \alpha_1, \alpha_\infty, \{A_\sigma\}_{\sigma \in \Sigma} \rangle$$

$\alpha_1 \in \mathbb{R}^n$ ➡ (initial weights) $\alpha_\infty \in \mathbb{R}^n$ ➡ (terminal weights)

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

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

Weighted Automata (WFA)



$$\alpha_1 = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} \quad A_a = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.1 \end{bmatrix}$$

$$\alpha_\infty = \begin{bmatrix} 0.0 \\ 0.6 \end{bmatrix} \quad A_b = \begin{bmatrix} 0.1 & 0.3 \\ 0.1 & 0.1 \end{bmatrix}$$

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

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

$\alpha_1 \in \mathbb{R}^n$ ➡ (initial weights) $\alpha_\infty \in \mathbb{R}^n$ ➡ (terminal weights)

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

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

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

Examples of functions computed by WFAs

Generates infinite strings, computes probabilities of prefixes $\mathbb{P}[x\Sigma^\star]$

Emission and transition are conditionally independent given state

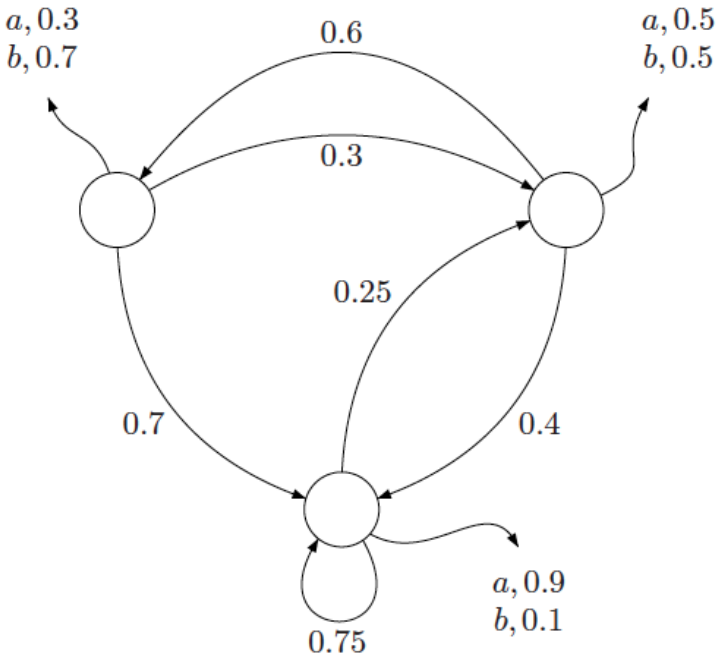
$$\alpha_1^\top = [0.3 \;\; 0.3 \;\; 0.4]$$

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

$$A_a = O_a \cdot T$$

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

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



HMMs

Examples of functions computed by WFAs

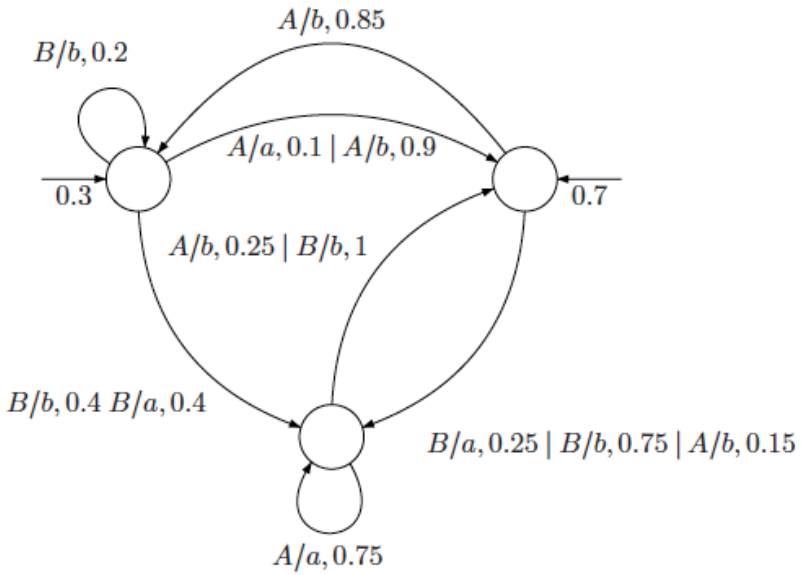
Compute conditional probabilities $\mathbb{P}[y|x] = \alpha_1^\top A_x^y \alpha_\infty$ for pairs $(x, y) \in (\Sigma \times \Delta)^\star$, must have $|x| = |y|$

Can also assume models factorized like in HMM

$$\alpha_1^\top = [0.3 \;\; 0 \;\; 0.7]$$

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

$$A_B^b = \begin{bmatrix} 0.2 & 0.4 & 0 \\ 0 & 0 & 1 \\ 0 & 0.75 & 0 \end{bmatrix}$$



FSTs

Forward-Backward Mappings

Forward-Backward Mappings

Any WA $\mathbf{A}$ defines *forward* and *backward* maps

Forward-Backward Mappings

Any WA $\mathbf{A}$ defines *forward* and *backward* maps 

Forward-Backward Mappings

Any WA $\mathbf{A}$ defines *forward* and *backward* maps $\Rightarrow$ $\alpha_{\mathbf{A}}, \beta_{\mathbf{A}} : \Sigma^{\star} \rightarrow \mathbb{R}^{n}$

Forward-Backward Mappings

Any WA $\mathbf{A}$ defines *forward* and *backward* maps $\Rightarrow$ $\alpha_{\mathbf{A}}, \beta_{\mathbf{A}} : \Sigma^\star \to \mathbb{R}^n$ such that for any splitting $x = p \cdot s$ one has

Forward-Backward Mappings

Any WA $\mathbf{A}$ defines *forward* and *backward* maps $\Rightarrow$ $\alpha_{\mathbf{A}}, \beta_{\mathbf{A}} : \Sigma^\star \to \mathbb{R}^n$

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

Forward-Backward Mappings

Any WA $\mathbf{A}$ defines *forward* and *backward* maps $\Rightarrow$ $\alpha_{\mathbf{A}}, \beta_{\mathbf{A}} : \Sigma^\star \to \mathbb{R}^n$ such that for any splitting $x = p \cdot s$ one has

$$\alpha_{\mathbf{A}}(p)^\top = \alpha_1^\top A_{p_1} \cdots A_{p_t}$$

Forward-Backward Mappings

Any WA $\mathbf{A}$ defines *forward* and *backward* maps $\Rightarrow$ $\alpha_{\mathbf{A}}, \beta_{\mathbf{A}} : \Sigma^\star \to \mathbb{R}^n$ such that for any splitting $x = p \cdot s$ one has

$$\alpha_{\mathbf{A}}(p)^\top = \alpha_1^\top A_{p_1} \cdots A_{p_t}$$

$$\beta_{\mathbf{A}}(s)^\top = A_{s_1} \cdots A_{s_{t'}} \alpha_\infty$$

Forward-Backward Mappings

Any WA $\mathbf{A}$ defines *forward* and *backward* maps $\Rightarrow$ $\alpha_{\mathbf{A}}, \beta_{\mathbf{A}} : \Sigma^\star \to \mathbb{R}^n$ such that for any splitting $x = p \cdot s$ one has

$$\alpha_{\mathbf{A}}(p)^\top = \alpha_1^\top A_{p_1} \cdots A_{p_t}$$
$$\beta_{\mathbf{A}}(s)^\top = A_{s_1} \cdots A_{s_{t'}} \alpha_\infty$$
$$f_{\mathbf{A}}(x) = \alpha_{\mathbf{A}}(p)^\top \beta_{\mathbf{A}}(s)$$

Forward-Backward Mappings

Any WA $\mathbf{A}$ defines *forward* and *backward* maps $\Rightarrow$ $\alpha_{\mathbf{A}}, \beta_{\mathbf{A}} : \Sigma^\star \to \mathbb{R}^n$ such that for any splitting $x = p \cdot s$ one has

$$\alpha_{\mathbf{A}}(p)^\top = \alpha_1^\top A_{p_1} \cdots A_{p_t}$$
$$\beta_{\mathbf{A}}(s)^\top = A_{s_1} \cdots A_{s_{t'}} \alpha_\infty$$
$$f_{\mathbf{A}}(x) = \alpha_{\mathbf{A}}(p)^\top \beta_{\mathbf{A}}(s)$$

Example, for HMMs:

Forward-Backward Mappings

Any WA $\mathbf{A}$ defines *forward* and *backward* maps $\Rightarrow$ $\alpha_{\mathbf{A}}, \beta_{\mathbf{A}} : \Sigma^\star \to \mathbb{R}^n$ such that for any splitting $x = p \cdot s$ one has

$$\alpha_{\mathbf{A}}(p)^\top = \alpha_1^\top A_{p_1} \cdots A_{p_t}$$
$$\beta_{\mathbf{A}}(s)^\top = A_{s_1} \cdots A_{s_{t'}} \alpha_\infty$$
$$f_{\mathbf{A}}(x) = \alpha_{\mathbf{A}}(p)^\top \beta_{\mathbf{A}}(s)$$

Example, for HMMs:

Forward-Backward Mappings

Any WA $\mathbf{A}$ defines *forward* and *backward* maps $\Rightarrow$ $\alpha_{\mathbf{A}}, \beta_{\mathbf{A}} : \Sigma^\star \to \mathbb{R}^n$ such that for any splitting $x = p \cdot s$ one has

$$\alpha_{\mathbf{A}}(p)^\top = \alpha_1^\top A_{p_1} \cdots A_{p_t}$$
$$\beta_{\mathbf{A}}(s)^\top = A_{s_1} \cdots A_{s_t} \alpha_\infty$$
$$f_{\mathbf{A}}(x) = \alpha_{\mathbf{A}}(p)^\top \beta_{\mathbf{A}}(s)$$

Example, for HMMs:

$$[\alpha_{\mathbf{A}}(p)]_i = \mathbb{P}[p\,,\, h_{+1} = i]$$

Forward-Backward Mappings

Any WA $\mathbf{A}$ defines *forward* and *backward* maps $\Rightarrow \alpha_{\mathbf{A}}, \beta_{\mathbf{A}} : \Sigma^\star \to \mathbb{R}^n$ such that for any splitting $x = p \cdot s$ one has

$$\alpha_{\mathbf{A}}(p)^\top = \alpha_1^\top A_{p_1} \cdots A_{p_t}$$
$$\beta_{\mathbf{A}}(s)^\top = A_{s_1} \cdots A_{s_t} \alpha_\infty$$
$$f_{\mathbf{A}}(x) = \alpha_{\mathbf{A}}(p)^\top \beta_{\mathbf{A}}(s)$$

Example, for HMMs:

$$[\alpha_{\mathbf{A}}(p)]_i = \mathbb{P}[p\,,\, h_{+1} = i]$$
$$[\beta_{\mathbf{A}}(s)]_i = \mathbb{P}[s \mid h = i]$$

Forward-Backward Mappings

Any WA $\mathbf{A}$ defines *forward* and *backward* maps $\Rightarrow$ $\alpha_{\mathbf{A}}, \beta_{\mathbf{A}} : \Sigma^\star \to \mathbb{R}^n$ such that for any splitting $x = p \cdot s$ one has

$$\alpha_{\mathbf{A}}(p)^\top = \alpha_1^\top A_{p_1} \cdots A_{p_t}$$
$$\beta_{\mathbf{A}}(s)^\top = A_{s_1} \cdots A_{s_t} \alpha_\infty$$
$$f_{\mathbf{A}}(x) = \alpha_{\mathbf{A}}(p)^\top \beta_{\mathbf{A}}(s)$$

Example, for HMMs:

$$[\alpha_{\mathbf{A}}(p)]_i = \mathbb{P}[p\,,\, h_{+1} = i]$$
$$[\beta_{\mathbf{A}}(s)]_i = \mathbb{P}[s \mid h = i]$$

Consequences:

Forward-Backward Mappings

Any WA $\mathbf{A}$ defines *forward* and *backward* maps $\alpha_{\mathbf{A}}, \beta_{\mathbf{A}} : \Sigma^\star \to \mathbb{R}^n$ such that for any splitting $x = p \cdot s$ one has

$$\alpha_{\mathbf{A}}(p)^\top = \alpha_1^\top A_{p_1} \cdots A_{p_t}$$
$$\beta_{\mathbf{A}}(s)^\top = A_{s_1} \cdots A_{s_{t'}} \alpha_\infty$$
$$f_{\mathbf{A}}(x) = \alpha_{\mathbf{A}}(p)^\top \beta_{\mathbf{A}}(s)$$

Example, for HMMs:

$$[\alpha_{\mathbf{A}}(p)]_i = \mathbb{P}[p\,,\, h_{+1} = i]$$
$$[\beta_{\mathbf{A}}(s)]_i = \mathbb{P}[s \mid h = i]$$

Consequences:

Forward-Backward Mappings

Any WA $\mathbf{A}$ defines *forward* and *backward* maps $\Rightarrow$ $\alpha_{\mathbf{A}}, \beta_{\mathbf{A}} : \Sigma^\star \to \mathbb{R}^n$ such that for any splitting $x = p \cdot s$ one has

$$\alpha_{\mathbf{A}}(p)^\top = \alpha_1^\top A_{p_1} \cdots A_{p_t}$$
$$\beta_{\mathbf{A}}(s)^\top = A_{s_1} \cdots A_{s_t} \alpha_\infty$$
$$f_{\mathbf{A}}(x) = \alpha_{\mathbf{A}}(p)^\top \beta_{\mathbf{A}}(s)$$

Example, for HMMs:

$$[\alpha_{\mathbf{A}}(p)]_i = \mathbb{P}[p\,,\, h_{+1} = i]$$
$$[\beta_{\mathbf{A}}(s)]_i = \mathbb{P}[s \mid h = i]$$

Consequences:

A_σ can be recovered from $f_{\mathbf{A}}(p\sigma s)$, $\alpha_{\mathbf{A}}(p)$, and $\beta_{\mathbf{A}}(s)$:

Forward-Backward Mappings

Any WA $\mathbf{A}$ defines *forward* and *backward* maps $\Rightarrow$ $\alpha_{\mathbf{A}}, \beta_{\mathbf{A}} : \Sigma^\star \to \mathbb{R}^n$ such that for any splitting $x = p \cdot s$ one has

$$\alpha_{\mathbf{A}}(p)^\top = \alpha_1^\top A_{p_1} \cdots A_{p_t}$$
$$\beta_{\mathbf{A}}(s)^\top = A_{s_1} \cdots A_{s_t} \alpha_\infty$$
$$f_{\mathbf{A}}(x) = \alpha_{\mathbf{A}}(p)^\top \beta_{\mathbf{A}}(s)$$

Example, for HMMs:

$$[\alpha_{\mathbf{A}}(p)]_i = \mathbb{P}[p\,,\, h_{+1} = i]$$
$$[\beta_{\mathbf{A}}(s)]_i = \mathbb{P}[s \mid h = i]$$

Consequences:

A_σ can be recovered from $f_{\mathbf{A}}(p\sigma s)$, $\alpha_{\mathbf{A}}(p)$, and $\beta_{\mathbf{A}}(s)$:

$$f_{\mathbf{A}}(p\sigma s) = \alpha_{\mathbf{A}}(p)^\top A_\sigma \, \beta_{\mathbf{A}}(s)$$

The Hankel Matrix

The Hankel Matrix

The *Hankel matrix* of $f : \Sigma^\star \to \mathbb{R}$ is

The Hankel Matrix

The *Hankel matrix* of $f : \Sigma^\star \to \mathbb{R}$ is 

The Hankel Matrix

The *Hankel matrix* of $f : \Sigma^\star \to \mathbb{R}$ is  $H_f \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$

The Hankel Matrix

The *Hankel matrix* of $f : \Sigma^\star \to \mathbb{R}$ is $\Rightarrow$ $H_f \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$

For $p, s \in \Sigma^\star$, entries are defined by

The Hankel Matrix

The *Hankel matrix* of $f : \Sigma^\star \to \mathbb{R}$ is $\Rightarrow$ $H_f \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$

For $p, s \in \Sigma^\star$, entries are defined by $\Rightarrow$

The Hankel Matrix

The *Hankel matrix* of $f : \Sigma^\star \to \mathbb{R}$ is $\Rightarrow$ $H_f \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$

For $p, s \in \Sigma^\star$, entries are defined by $\Rightarrow$ $H_f(p, s) = f(p \cdot s)$

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 $p, s \in \Sigma^\star$, entries are defined by $H_f(p, s) = f(p \cdot s)$

The Hankel Matrix

The *Hankel matrix* of $f : \Sigma^\star \to \mathbb{R}$ is $\mathbf{H}_f \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$

For $p, s \in \Sigma^\star$, entries are defined by $\mathbf{H}_f(p, s) = f(p \cdot s)$

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

The Hankel Matrix

The *Hankel matrix* of $f : \Sigma^\star \to \mathbb{R}$ is $\mathbf{H}_f \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$

For $p, s \in \Sigma^\star$, entries are defined by $\mathbf{H}_f(p, s) = f(p \cdot s)$

Very *redundant* representation

	ϵ	a	b	aa	ab	$\cdots$
ϵ			$\vdots$		$\vdots$	
a	$\cdots$	$\cdots$	$\vdots$	$\cdots$	$f(aab)$	
b			$\vdots$			
aa	$\cdots$	$\cdots$	$f(aab)$			
ab						
$\vdots$						

The Hankel Matrix

The *Hankel matrix* of $f : \Sigma^\star \to \mathbb{R}$ is $\Rightarrow$ $H_f \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$

For $p, s \in \Sigma^\star$, entries are defined by $\Rightarrow$ $H_f(p, s) = f(p \cdot s)$

Very *redundant* representation

we will work with finite sub-blocks of H

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

Hankel Matrix of a WFA

Hankel Matrix of a WFA

Schützenberger's Theorem

Hankel Matrix of a WFA

Schützenberger's Theorem 

Hankel Matrix of a WFA

Schützenberger's Theorem

Theorem: $\text{rank}(H_f) = n \iff f = f_{\mathbf{A}}$ with minimal $\mathbf{A}$ of size n

Hankel Matrix of a WFA

Schützenberger's Theorem 

Theorem: $\text{rank}(H_f) = n \iff f = f_{\mathbf{A}}$ with minimal $\mathbf{A}$ of size n

Remarks ($\Leftarrow$)

Hankel Matrix of a WFA

Schützenberger's Theorem

Theorem: $\text{rank}(H_f) = n \iff f = f_{\mathbf{A}}$ with minimal $\mathbf{A}$ of size n

Remarks ($\Leftarrow$)

Write $F = \alpha_{\mathbf{A}}(\Sigma^\star)^\top \in \mathbb{R}^{\Sigma^\star \times n}$ and $B = \beta_{\mathbf{A}}(\Sigma^\star) \in \mathbb{R}^{n \times \Sigma^\star}$

Note $H_f = F \cdot B$

Then, $\text{rank}(H_f) = n$

Hankel Matrix of a WFA

Schützenberger's Theorem

Theorem: $\text{rank}(H_f) = n \iff f = f_{\mathbf{A}}$ with minimal $\mathbf{A}$ of size n

Remarks ($\Leftarrow$)

Write $F = \alpha_{\mathbf{A}}(\Sigma^\star)^\top \in \mathbb{R}^{\Sigma^\star \times n}$ and $B = \beta_{\mathbf{A}}(\Sigma^\star) \in \mathbb{R}^{n \times \Sigma^\star}$

Note $H_f = F \cdot B$

Then, $\text{rank}(H_f) = n$

Remarks ($\Rightarrow$)

Hankel Matrix of a WFA

Schützenberger's Theorem

Theorem: $\text{rank}(H_f) = n \iff f = f_{\mathbf{A}}$ with minimal $\mathbf{A}$ of size n

Remarks ($\Leftarrow$)

Write $F = \alpha_{\mathbf{A}}(\Sigma^\star)^\top \in \mathbb{R}^{\Sigma^\star \times n}$ and $B = \beta_{\mathbf{A}}(\Sigma^\star) \in \mathbb{R}^{n \times \Sigma^\star}$

Note $H_f = F \cdot B$

Then, $\text{rank}(H_f) = n$

Remarks ($\Rightarrow$)

Assume $\text{rank}(H_f) = n$

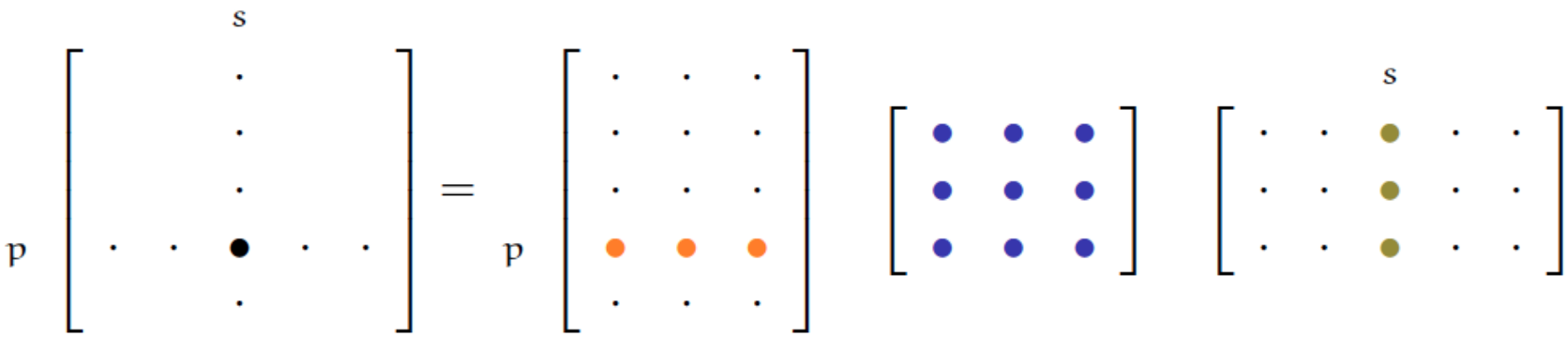
Take *rank factorization* $H_f = F \cdot B$ with $F \in \mathbb{R}^{\Sigma^\star \times n}$ and $B \in \mathbb{R}^{n \times \Sigma^\star}$

A WA that computes f can be constructed using F and B

Low-rank Hankel and Operators

Low-rank Hankel and Operators

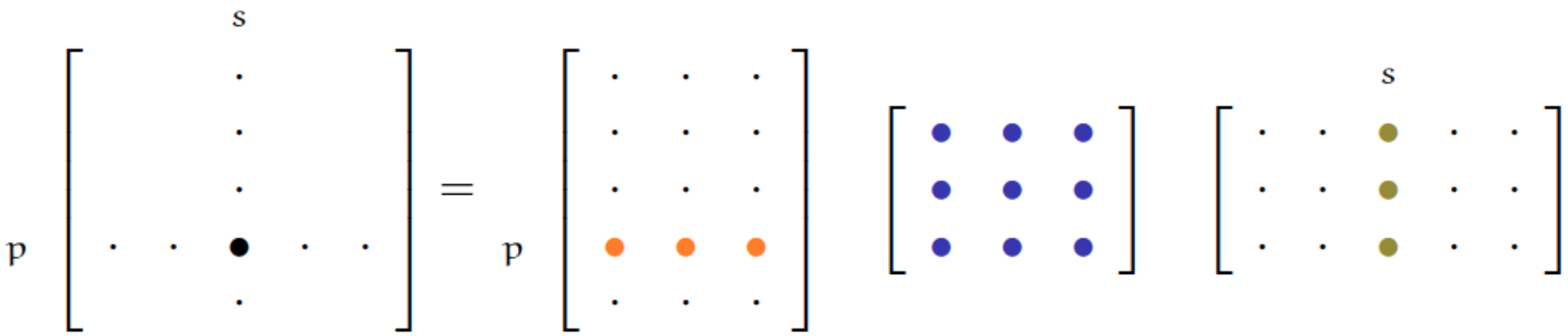
$$H_\sigma \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star} \qquad F \in \mathbb{R}^{\Sigma^\star \times n} \qquad A_\sigma \in \mathbb{R}^{n \times n} \qquad B \in \mathbb{R}^{n \times \Sigma^\star}$$



$$f(p_1 \cdots p_t \cdot \sigma \cdot s_1 \cdots s_{t'}) = \underbrace{\alpha_1^\top A_{p_1} \cdots A_{p_t}}_{\alpha(p)^\top} \; A_\sigma \; \underbrace{A_{s_1} \cdots A_{s_{t'}} \alpha_\infty}_{\beta(s)}$$

Low-rank Hankel and Operators

$$H_\sigma \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star} \qquad F \in \mathbb{R}^{\Sigma^\star \times n} \qquad A_\sigma \in \mathbb{R}^{n \times n} \qquad B \in \mathbb{R}^{n \times \Sigma^\star}$$



$$f(p_1 \cdots p_t \cdot \sigma \cdot s_1 \cdots s_{t'}) = \underbrace{\alpha_1^\top A_{p_1} \cdots A_{p_t}}_{\alpha(p)^\top} \; A_\sigma \; \underbrace{A_{s_1} \cdots A_{s_{t'}} \alpha_\infty}_{\beta(s)}$$

$$H_\sigma = F\, A_\sigma\, B \quad \Longrightarrow \quad A_\sigma = F^+\, H_\sigma\, B^+$$

Spectral Method

Spectral Method

Idea: Use SVD decomposition to obtain a factorization of H

- Choose a basis: sets of prefixes $\mathcal{P}$ and suffixes $\mathcal{S}$
- Obtain H and H_σ over basis $(\mathcal{P}, \mathcal{S})$
- Compute *compact* SVD as $H = U\Lambda V^\top$ with

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

- Construct the WA using $F = U\Lambda$ and $B = V^\top$:
  - $A_\sigma = F^+ H_\sigma V^+$ for all $\sigma \in \Sigma$
  - $\alpha_1^\top = H(\lambda, \cdot)^\top B^+, \qquad \alpha_\infty = F^+ H(\cdot, \lambda)$

Spectral Method

Idea: Use SVD decomposition to obtain a factorization of H

- Choose a basis: sets of prefixes $\mathcal{P}$ and suffixes $\mathcal{S}$
- Obtain H and H_σ over basis $(\mathcal{P}, \mathcal{S})$
- Compute *compact* SVD as $H = U\Lambda V^\top$ with

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

- Construct the WA using $F = U\Lambda$ and $B = V^\top$:
  - $A_\sigma = F^+ H_\sigma V^+$ for all $\sigma \in \Sigma$
  - $\alpha_1^\top = H(\lambda, \cdot)^\top B^+, \qquad \alpha_\infty = F^+ H(\cdot, \lambda)$

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 $\hat{A}_\sigma \approx A_\sigma$

Spectral Method

Choice of Preffix-Suffix Basis $(\mathcal{P}, \mathcal{S})$

- Should satisfy that $\text{rank}(H_{\Sigma^\star \times \Sigma^\star}) = \text{rank}(H_{\mathcal{P} \times \mathcal{S}})$
  - $\mathcal{P} = \mathcal{S} = \Sigma \cup \{\lambda\}$ (Hsu et al COLT-09)
  - all prefixes/suffixes in the training data
  - most frequent ones (Balle et al MLJ-13)
  - feature-based equivalence classes (Cohen et al NAACL-13)

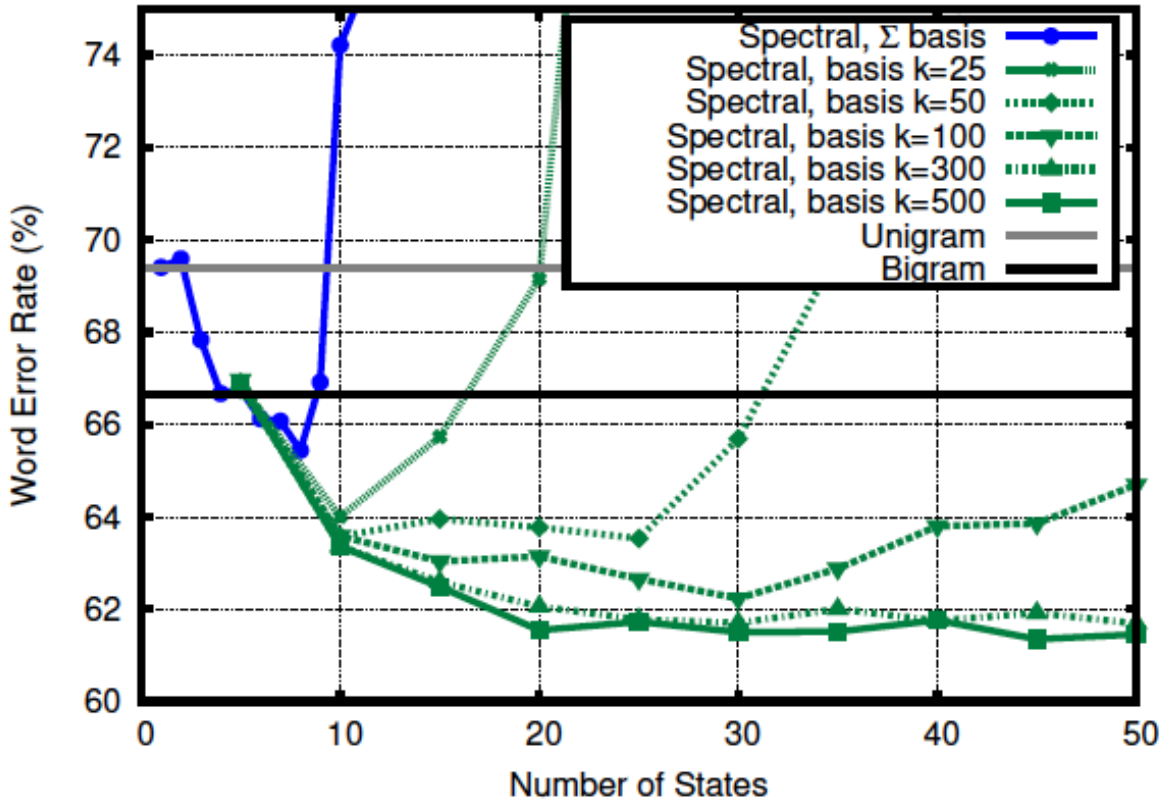
Observable Statistics in H and H_σ

- The WA can be recovered from
  - String probabilities $\mathbb{P}[x]$
  - Prefix probabilities $\mathbb{P}[x\Sigma^*]$ (Hsu et al COLT-09)
  - Substring expectations $\mathbb{E}[|w|_x]$
- See (Balle et al MLJ-13) for details

Parameter of the method

- A dimension n: controls complexity of the WA family

Experiment: PoS-Tag Sequence Models

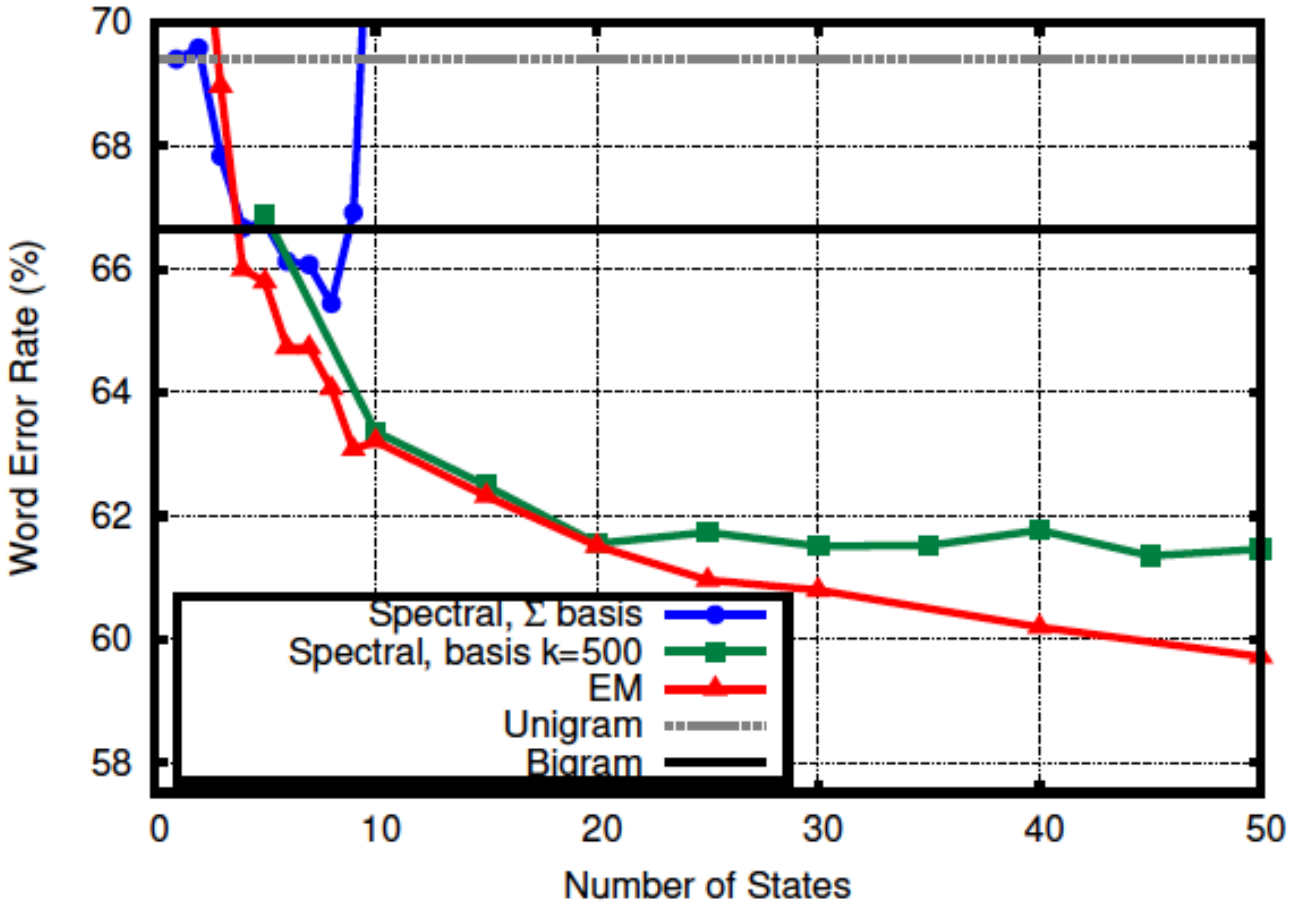


PTB sequences of simplified PoS tags (Petrov et al 2010)

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

Classical Spectral: Summary

Classical Spectral: Summary

Learning Setting:

- Considers stochastic functions
- Assumes there is a model that generates sequences
- We can sample from this model

Classical Spectral: Summary

Learning Setting:

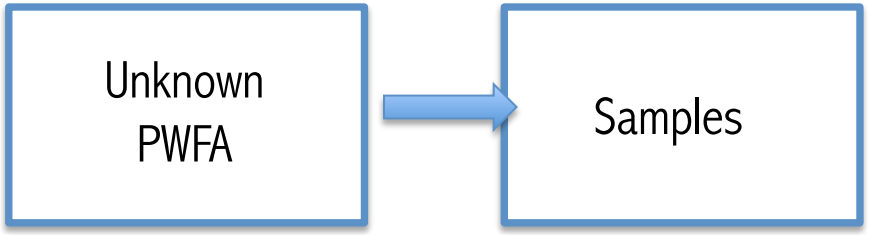
- Considers stochastic functions
- Assumes there is a model that generates sequences
- We can sample from this model

Unknown PWFA

Classical Spectral: Summary

Learning Setting:

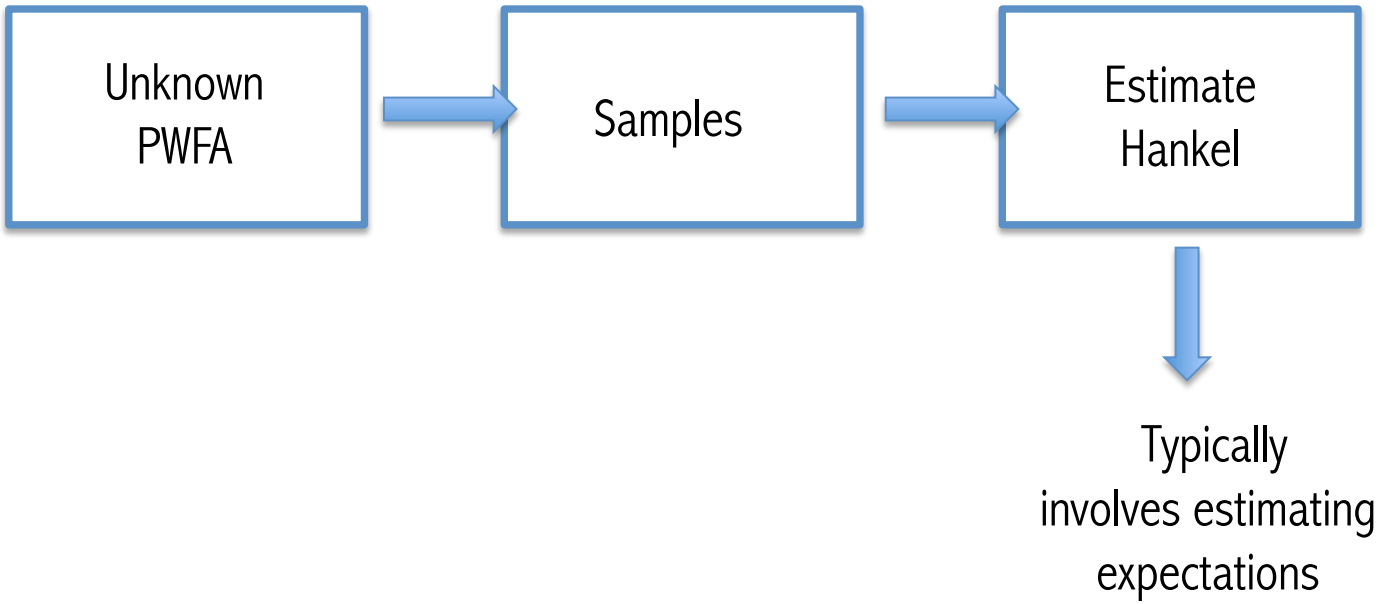
- Considers stochastic functions
- Assumes there is a model that generates sequences
- We can sample from this model



Classical Spectral: Summary

Learning Setting:

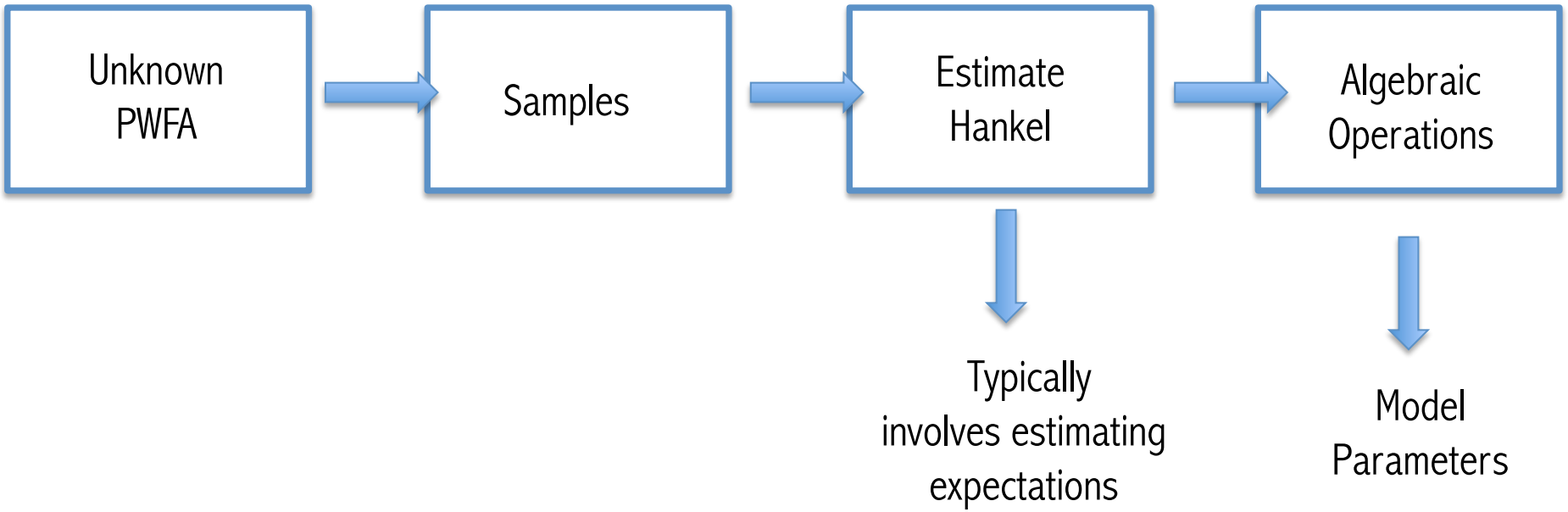
- Considers stochastic functions
- Assumes there is a model that generates sequences
- We can sample from this model



Classical Spectral: Summary

Learning Setting:

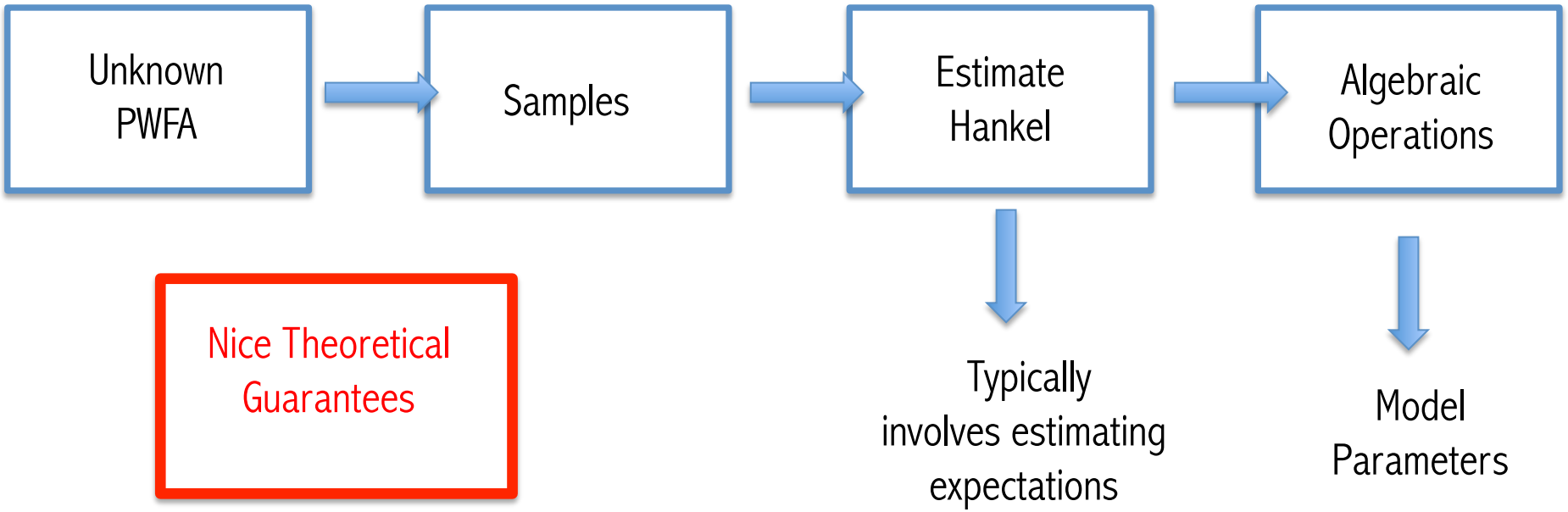
- Considers stochastic functions
- Assumes there is a model that generates sequences
- We can sample from this model



Classical Spectral: Summary

Learning Setting:

- Considers stochastic functions
- Assumes there is a model that generates sequences
- We can sample from this model



Classical Spectral: Limitations

Classical Spectral: Limitations

- Necessary statistics are hard to estimate
- In some cases we can not directly estimate the Hankel from samples

  eg. Learning a function that is not a distribution

  Learning a PCFG when we don't have derivations

  Learning transducers when we don't have alignments

Classical Spectral: Limitations

- Necessary statistics are hard to estimate
- In some cases we can not directly estimate the Hankel from samples:

  eg. Learning a function that is not a distribution
  Learning a PCFG when we don't have derivations
  Learning transducers when we don't have alignments

It would be better not to have to assume that samples are generated by some model !!

Classical Spectral: Limitations

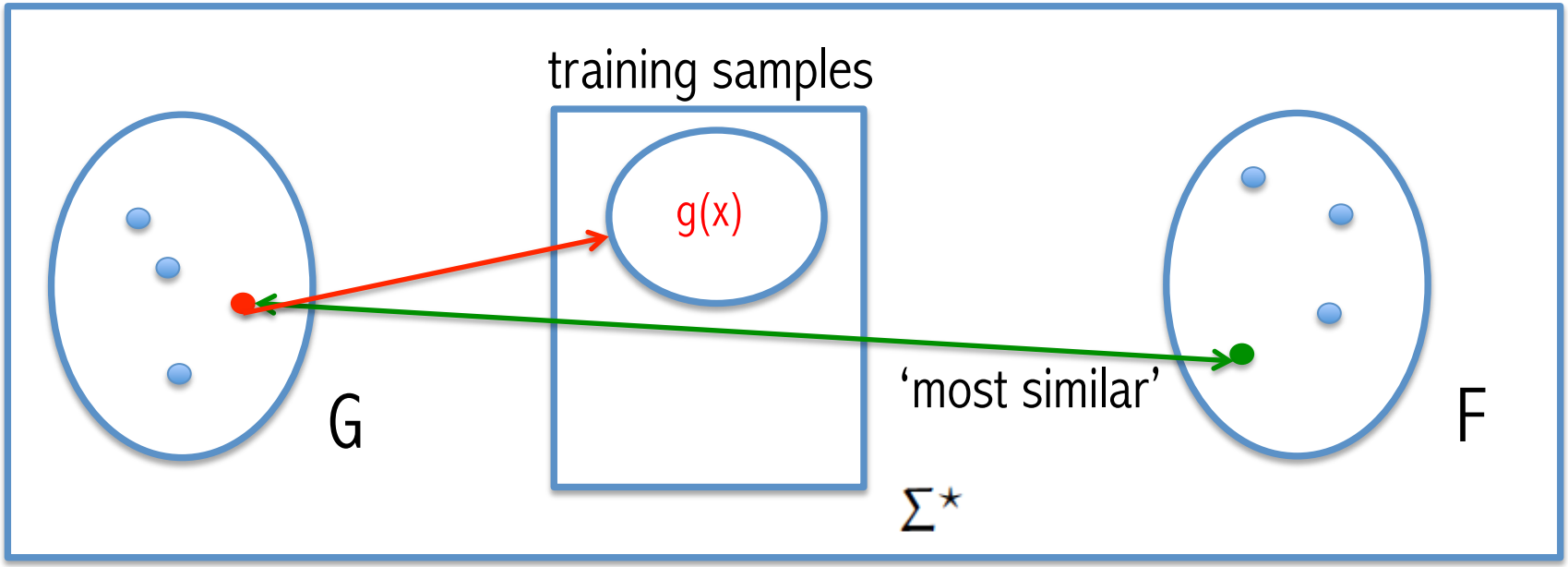
- Necessary statistics are hard to estimate
- In some cases we can not directly estimate the Hankel from samples:

  eg. Learning a function that is not a distribution

  Learning a PCFG when we don't have derivations

  Learning transducers when we don't have alignments

It would be better not to have to assume that samples are generated by some model !!



Outline

- Spectral Learning of WFA
- **Hankel-Based Optimization for Structured Prediction**

Outline

- Spectral Learning of WFA
- **Hankel-Based Optimization for Structured Prediction**

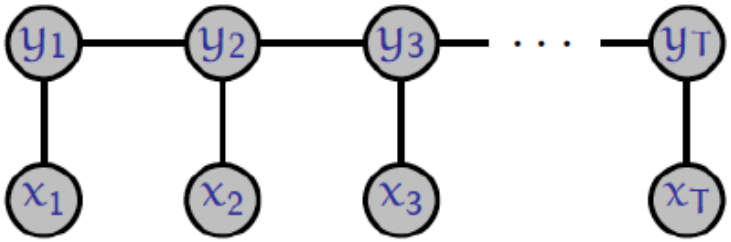
Outline

- Spectral Learning of WFA
- **Hankel-Based Optimization for Structured Prediction** → Sequence Tagging

Sequence Tagging

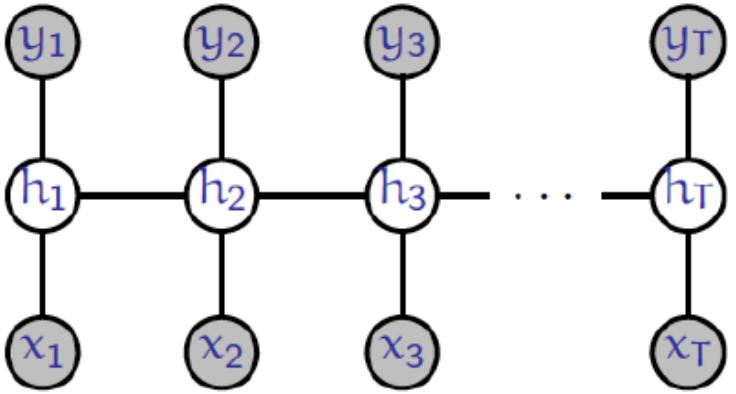
output:	h	l	p	-	x	p	a	t	x	m	x	s
input:	h	i	p	p	o	p	o	t	a	m	u	s

Fully Observable Models



+ Making predictions is tractable
+ Learning is convex
– Performance crucially depends on features

Latent-variable Models



+ Hidden layer provides more expressivity
– Making predicitons is not tractable
– Learning is non-convex (this paper)

Learning Structured Predictors with Latent Variables

Desiderata:

- Expressive scoring functions
- Tractable prediction function
- Effective regularizer
- *Convex training procedure*

Main Idea: Change of Representation + Relaxation

- Problem Formulation
  - Scoring functions computed by Input-Output OOMs
  - Piecewise Prediction and Loss Function

- Solving the Learning Problem
  - Spectral trick:
    optimize over parameters of $f \rightarrow$ optimize low-rank matrix H
  - Relax low-rank constraint using nuclear norm of H
  - Recover parameters of f from H using the spectral method

Scoring Functions Computed by IO-OOM

Latent Score $\theta(x, y, h)$:

$$\alpha(h_0) \prod_{t=1}^{T} A_{y_t}^{x_t}(h_{t-1}, h_t)\ \beta(h_T)$$

Scoring Function $F_A(x, y)$:

$$\sum_h \theta(x, y, h) = \alpha^T\ A_{y_1}^{x_1} \dots A_{y_T}^{x_T}\ \beta$$

- Model: $A : \langle \alpha, \beta, \{A_b^a\} \rangle$
- Number of states: n
- Initital Weights: $\alpha \in \mathbb{R}^n$
- Final Weights: $\beta \in \mathbb{R}^n$
- Observable Operators $A_b^a \in \mathbb{R}^{n \times n}$

- Expressive Function Family $\rightarrow$ e.g. it includes HMM
- Making Predictions (i.e. maximizing $F_A(x, y)$) $\rightarrow$ NP-hard

Piecewise Prediction and Loss for IO-OOM

Approximation: $F_A^k(x, y)$:

$$\sum_{t=1}^{T-(k-1)} F_A(x_{t:t+k-1}, y_{t:t+k-1})$$

- Factor size: k
- Sum k–grams
- Task loss: $l(y, z)$
  e.g. hamming distance

Loss $L_k(x, y, F_A)$:

$$\max_z [F_A^k(x, z) - F_A^k(x, y) + l(y, z))$$

- Prediction and Loss Function $\rightarrow$ computed in $O(T|Y|^k)$
  using the Viterbi Algorithm

Discrete Regularizer for IO-OOM

Learning Problem:

$$\underset{A \in \mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{m} L_k(x^i, y^i, F_A) + \tau |A|)$$

- Function class (IO-OOM): $\mathcal{F}$
- Training Examples: $\langle x^i, y^i \rangle$
- Loss Function: L_k
- Regularizer $\rightarrow$ number of states: $|A|$
- Trade-off constant: τ

- $k \geqslant 2 \rightarrow$ Non-convex dependence of L_k on parameters of A
- L_k involves polynomials of order $k + 3$

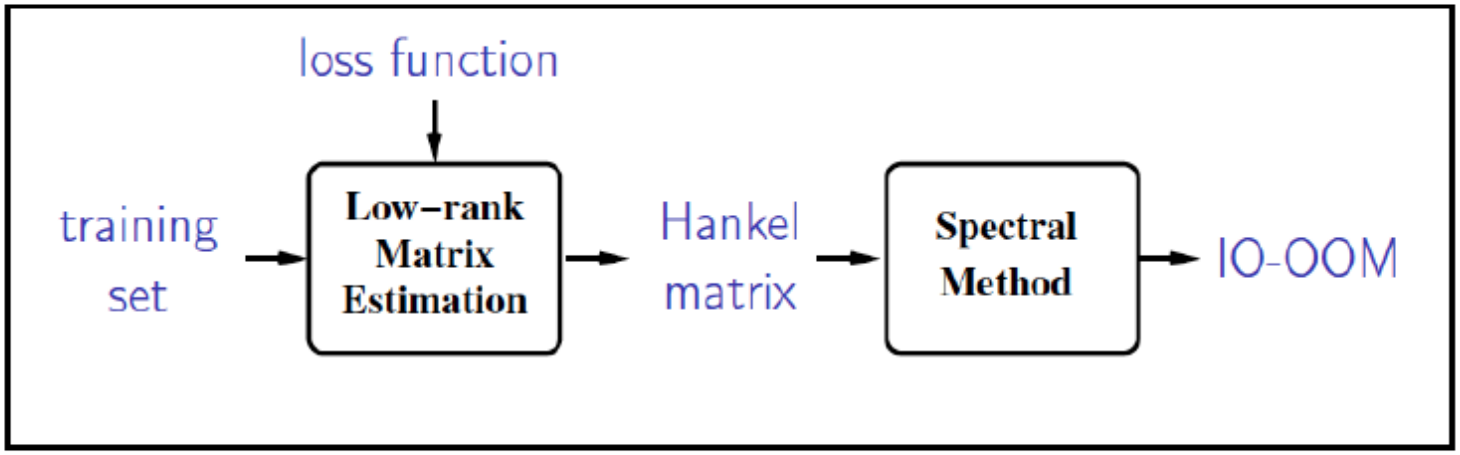
Optimization Strategy

- L_k convex on values of A $\rightarrow$ optimization over $(X \times Y)^k$ values
- Three challenges
  1. Table of values $\rightarrow$ must correspond to valid IO-OOM
  2. Regularizer over table $\rightarrow$ must correspond to #states of IO-OOM
  3. Recover parameters of A from this table

Optimization Strategy

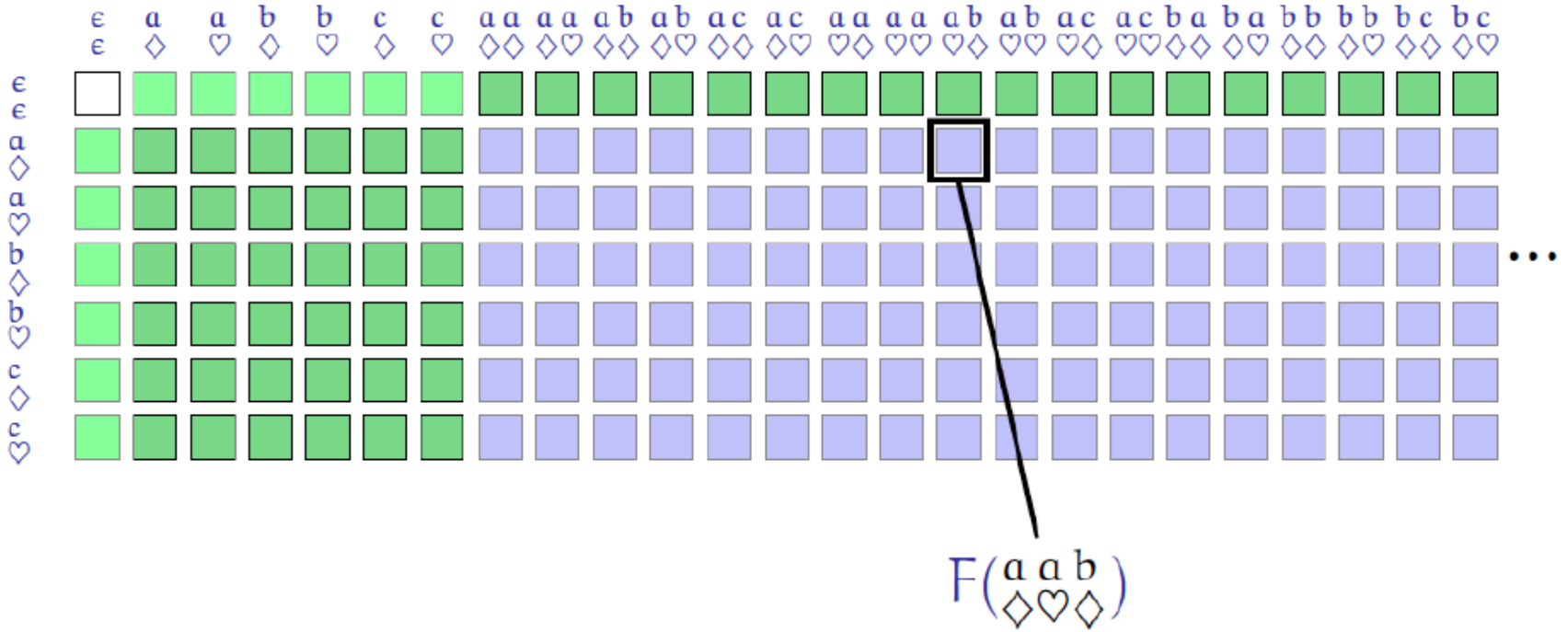
- L_k convex on values of A $\rightarrow$ optimization over $(X \times Y)^k$ values
- Three challenges
  1. Table of values $\rightarrow$ must correspond to valid IO-OOM
  2. Regularizer over table $\rightarrow$ must correspond to #states of IO-OOM
  3. Recover parameters of A from this table

Solution: the Spectral Trick



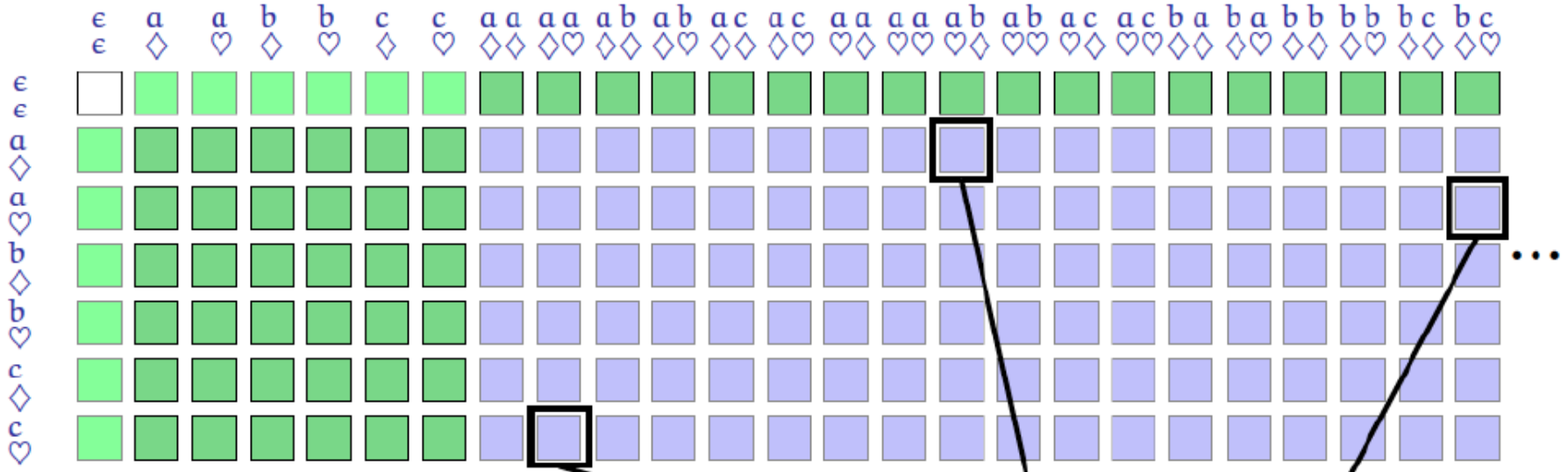
IO-OOM and Hankel Matrices

$$X = \{a, b, c\} \quad Y = \{\diamondsuit, \heartsuit\}$$



IO-OOM and Hankel Matrices

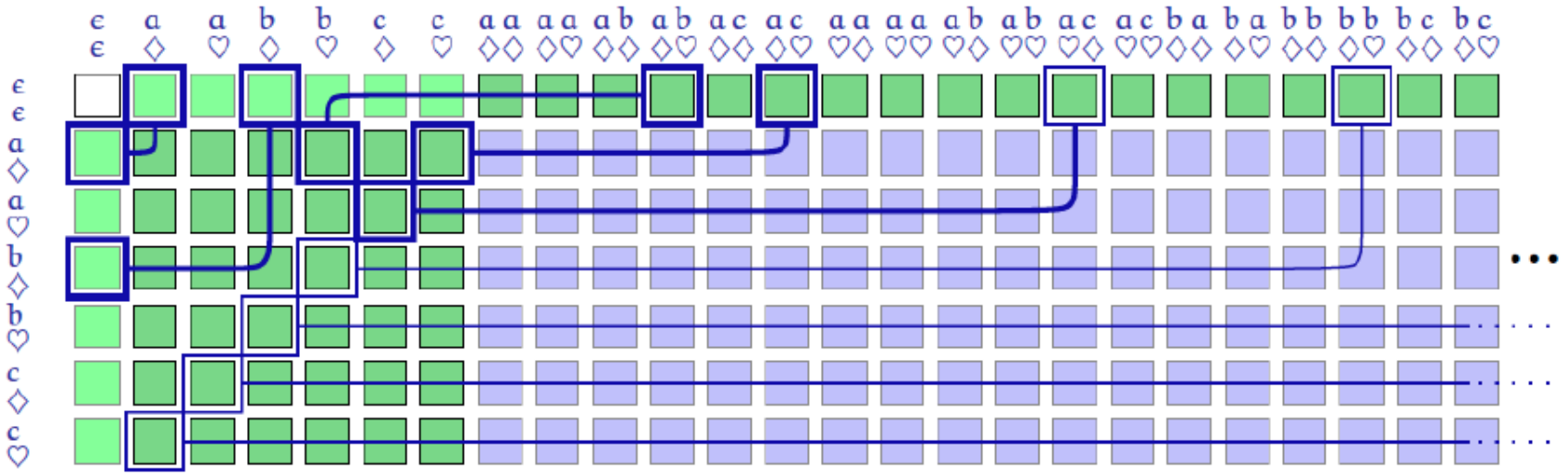
$$X = \{a, b, c\} \quad Y = \{\diamondsuit, \heartsuit\}$$



$$F_{k=3}\left(\begin{matrix} c & a & a & b & c \\ \heartsuit & \diamondsuit & \heartsuit & \diamondsuit & \heartsuit \end{matrix}\right) = F\left(\begin{matrix} c & a & a \\ \heartsuit & \diamondsuit & \heartsuit \end{matrix}\right) + F\left(\begin{matrix} a & a & b \\ \diamondsuit & \heartsuit & \diamondsuit \end{matrix}\right) + F\left(\begin{matrix} a & b & c \\ \heartsuit & \diamondsuit & \heartsuit \end{matrix}\right)$$

IO-OOM and Hankel Matrices

$$X = \{a, b, c\} \quad Y = \{\diamondsuit, \heartsuit\}$$



Hankel Structure:

- Equality constraints
- Low-rank constraints

Fundamental Theorem:

F is realized by an n-state IO-OOM

$\Longleftrightarrow$

H has rank at most n for every basis

Max-Margin Completion of Hankel Matrices

Optimization with rank regularization:

$$\underset{H \in \mathbb{H}(P,S)}{\operatorname{argmin}} \sum_{i=1}^{m} L_k(x^i, y^i, H) + \tau \operatorname{rank}(H)$$

Convex relaxation:

$$\underset{H \in \mathbb{H}(P,S)}{\operatorname{argmin}} \sum_{i=1}^{m} L_k(x^i, y^i, H) + \tau ||H||_*$$

- Set of Hankel Matrices over some basis: $\mathbb{H}(P, S)$
- Rank regularizer: $\operatorname{rank}(H)$
- Nuclear norm relaxation: $||H||_*$

- Optimization almost equivalent $\rightarrow$ we search over IO-OOM that can be recovered from $H \in \mathbb{H}(P, S)$
- Once we solve for H we can recover parameters using the spectral technique

Estimation of Hankel Matrices via Convex Optimization

FOBOS Algorithm: Minimization of $L(H) + \tau ||H||_*$

- Initialize: $H_0 = 0$
- while $t \leqslant MaxIter$ do
  - Set G_t to a subgradient of $L(H)$ at H_t
  - Set $H_{t+0.5} = H_t - \frac{c}{\sqrt{t}} G_t$
  - Calculate the SVD of $H_{t+0.5} = U\Sigma V^\top$
  - Define a diagonal matrix Σ' such that $\sigma_i^{'} = max[\sigma_i - \nu_t \tau, 0]$
  - set $H_{t+1} = U\Sigma' V^\top$

  end while

Spectral Recovery

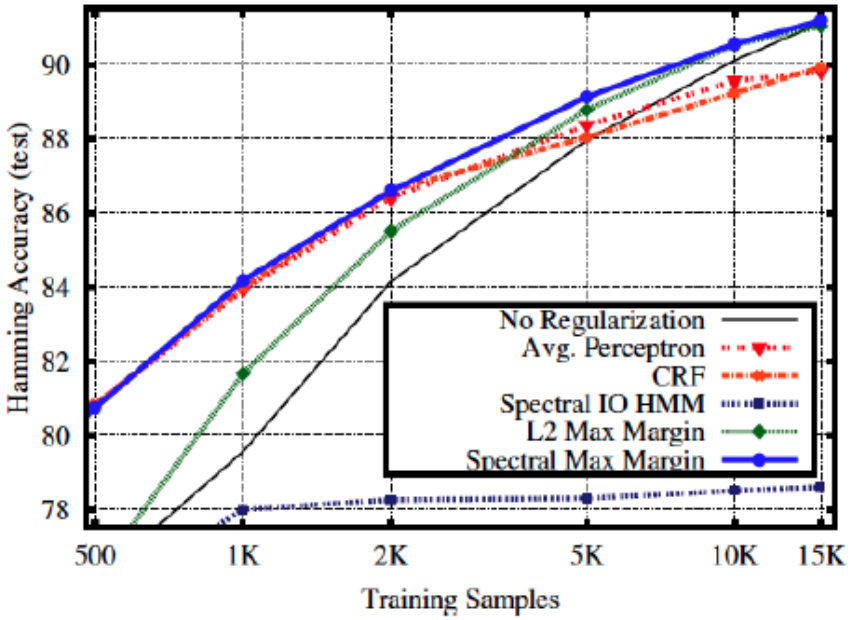
using the method by (Hsu et al. 2009)

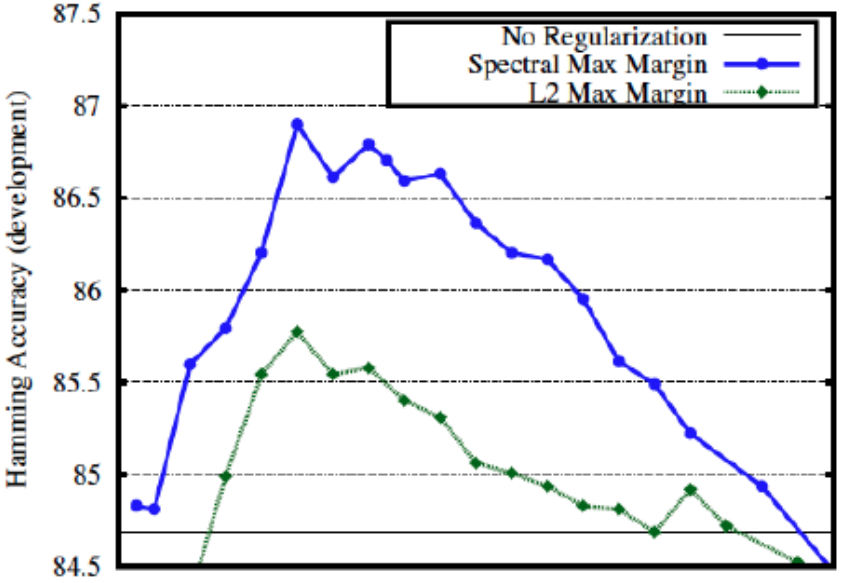
- Spectral Algorithm for IO-OOM
  - Assume F is realized by a minimal n-state IO-OOM A
  - We are given a basis (P, S) such that H has rank n
  - We are given corresponding H_b^a
  - To recover parameters of A:
    - Perform SVD to get $H = U\Sigma V^\top$
    - Define $A_b^a = (HV)^+ H_b^a V$
- Typical spectral algorithms assume that we can estimate H
- In contrast, we regard H as an optimization variable in a loss minimization procedure

Experiments

- Task: Phonetic Transcription (UCI "Nettalk" Dataset)

@	p	-	L	-		h	I	p	-	x	p	a	t	x	m	x	s
a	p	p	I	e		h	i	p	p	o	p	o	t	a	m	u	s





New Regularization for Factorized Linear Models

New Regularization for Factorized Linear Models

Feature Function

$$\phi_{abc;ABC}(x, y, t) \Rightarrow \begin{array}{ll} 1 & \text{if } \begin{array}{l} y_{t-1}y_t y_{t+1} = ABC \\ x_{t-1}x_t x_{t+1} = abc \end{array} \\ 0 & \text{otherwise} \end{array}$$

feature vector

$$\phi(x, y, t) \in \mathbb{R}^{|X|^3 \times |Y|^3}$$

parameter vector

$$w \in \mathbb{R}^{|X|^3 \times |Y|^3}$$

New Regularization for Factorized Linear Models

Feature Function

$$\phi_{abc;ABC}(x,y,t) \Rightarrow \begin{array}{l} 1 \text{ if } \begin{array}{l} y_{t-1}y_t y_{t+1} = ABC \\ x_{t-1}x_t x_{t+1} = abc \end{array} \\ 0 \text{ otherwise} \end{array}$$

feature vector

$$\phi(x,y,t) \in \mathbb{R}^{|X|^3 \times |Y|^3}$$

parameter vector

$$w \in \mathbb{R}^{|X|^3 \times |Y|^3}$$

Scoring function

$$\lambda(x,y) = \sum_t w^t \phi(x,y,t) \Rightarrow \sum_t w(x_{t-1}, x_t, x_{t+1}; y_{t-1}, y_t, y_{t+1})$$

New Regularization for Factorized Linear Models

Feature Function

$$\phi_{abc;ABC}(x,y,t) \rightarrow \begin{cases} 1 \text{ if } \begin{matrix} y_{t-1}y_t y_{t+1} = ABC \\ x_{t-1}x_t x_{t+1} = abc \end{matrix} \\ 0 \text{ otherwise} \end{cases}$$

feature vector

$$\phi(x,y,t) \in \mathbb{R}^{|X|^3 \times |Y|^3}$$

parameter vector

$$w \in \mathbb{R}^{|X|^3 \times |Y|^3}$$

Scoring function

$$\lambda(x,y) = \sum_t w^t \phi(x,y,t) \rightarrow \sum_t w(x_{t-1}, x_t, x_{t+1}; y_{t-1}, y_t, y_{t+1})$$

Think of parameter as a vector → l_1 l_2 regularization

New Regularization for Factorized Linear Models

Feature Function

$$\phi_{abc;ABC}(x,y,t) \rightarrow \begin{cases} 1 \text{ if } \begin{matrix} y_{t-1}y_t y_{t+1} = ABC \\ x_{t-1}x_t x_{t+1} = abc \end{matrix} \\ 0 \text{ otherwise} \end{cases}$$

feature vector

$$\phi(x,y,t) \in \mathbb{R}^{|X|^3 \times |Y|^3}$$

parameter vector

$$w \in \mathbb{R}^{|X|^3 \times |Y|^3}$$

Scoring function

$$\lambda(x,y) = \sum_t w^t \phi(x,y,t) \rightarrow \sum_t w(x_{t-1}, x_t, x_{t+1}; y_{t-1}, y_t, y_{t+1})$$

Think of parameter as a vector → l_1 l_2 regularization

Directly from multiclass prediction

New Regularization for Factorized Linear Models

Feature Function

$$\phi_{abc;ABC}(x,y,t) \rightarrow \begin{cases} 1 \text{ if } y_{t-1}y_t y_{t+1} = ABC \\ \quad\; x_{t-1}x_t x_{t+1} = abc \\ 0 \text{ otherwise} \end{cases}$$

feature vector

$$\phi(x,y,t) \in \mathbb{R}^{|X|^3 \times |Y|^3}$$

parameter vector

$$w \in \mathbb{R}^{|X|^3 \times |Y|^3}$$

Scoring function

$$\lambda(x,y) = \sum_t w^t \phi(x,y,t) \rightarrow \sum_t w(x_{t-1}, x_t, x_{t+1}; y_{t-1}, y_t, y_{t+1})$$

Think of parameter as a vector → l_1 l_2 regularization

Directly from multiclass prediction

W Think of parameter as a matrix → Spectral regularization

New Regularization for Factorized Linear Models

Feature Function

$$\phi_{abc;ABC}(x,y,t) \rightarrow \begin{cases} 1 \text{ if } \begin{matrix} y_{t-1}y_t y_{t+1} = ABC \\ x_{t-1}x_t x_{t+1} = abc \end{matrix} \\ 0 \text{ otherwise} \end{cases}$$

feature vector

$$\phi(x,y,t) \in \mathbb{R}^{|X|^3 \times |Y|^3}$$

parameter vector

$$w \in \mathbb{R}^{|X|^3 \times |Y|^3}$$

Scoring function

$$\lambda(x,y) = \sum_t w^t \phi(x,y,t) \rightarrow \sum_t w(x_{t-1}, x_t, x_{t+1}; y_{t-1}, y_t, y_{t+1})$$

Think of parameter as a vector → l_1 l_2 regularization

Directly from multiclass prediction

W Think of parameter as a matrix → Spectral regularization

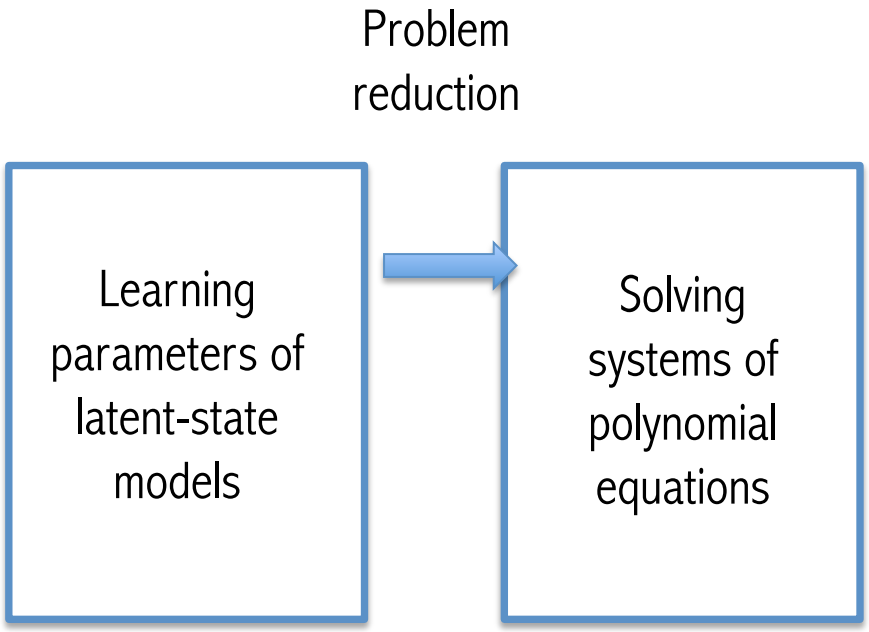
Specifically designed for structured prediction

Spectral Trick:
Change of Representation

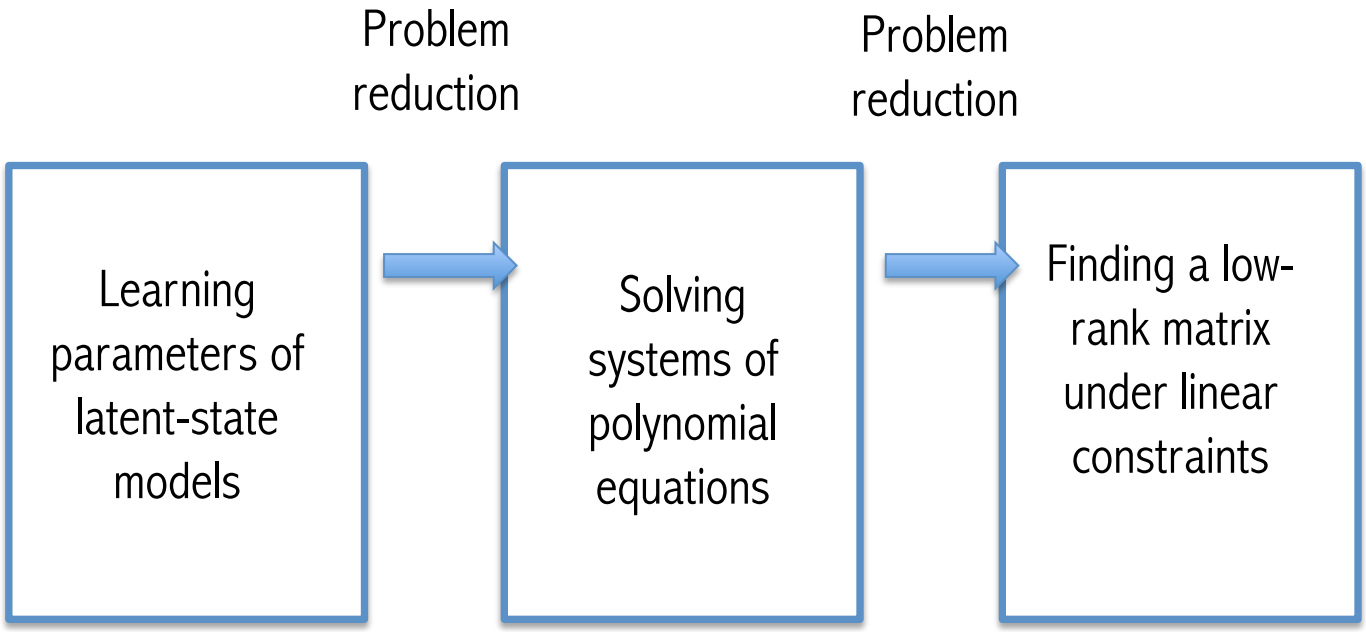
Spectral Trick:
Change of Representation

Learning parameters of latent-state models

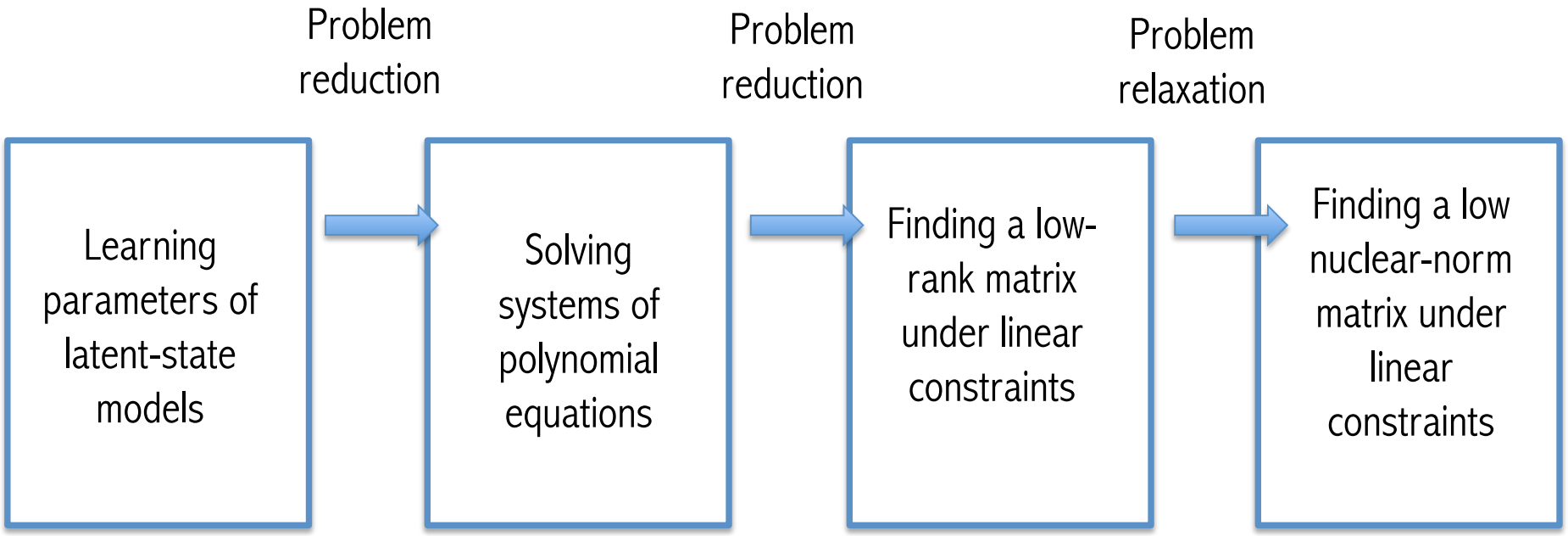
Spectral Trick: Change of Representation



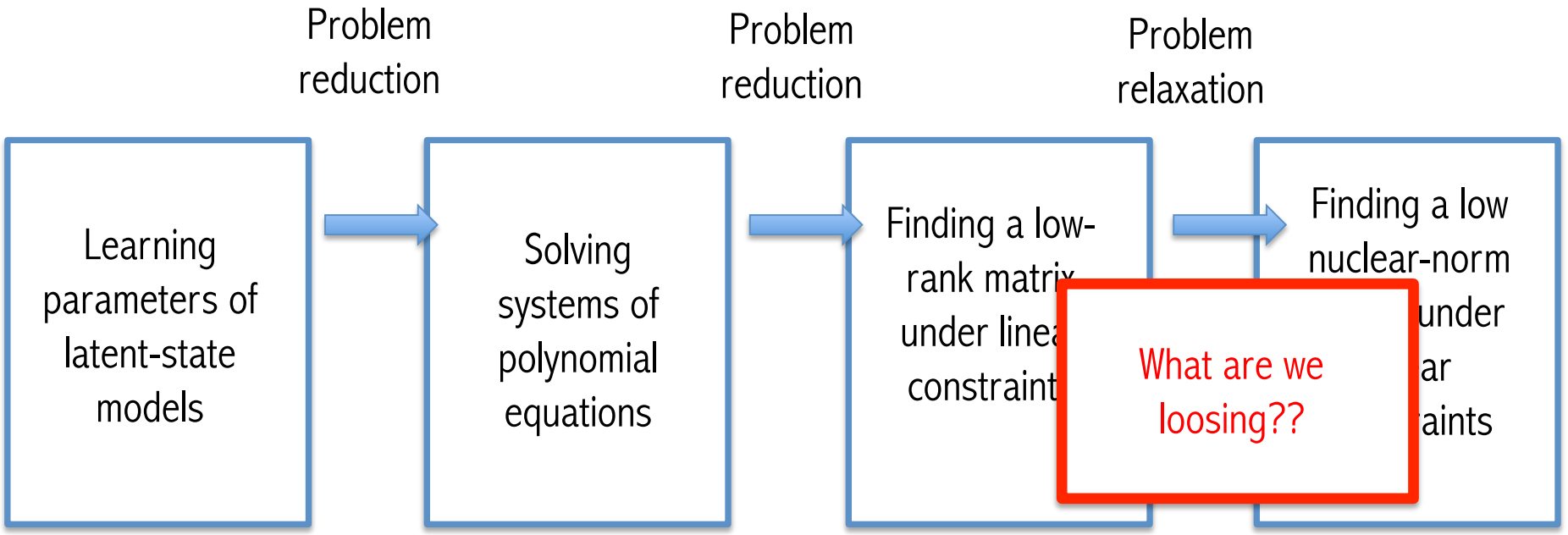
Spectral Trick: Change of Representation



Spectral Trick: Change of Representation



Spectral Trick:
Change of Representation



Take-home message: Fundamental ideas behind spectral learning have a wide range of applicability for structured prediction

Thanks!