Multitask Spectral Learning of Weighted Automata

Guillaume Rabusseau, Borja Balle and Joelle Pineau

IVADO - McGill University - Reasoning and Learning Lab

July 16, 2018

MAGNET Seminar - Lille

Learning with Structured Data

Supervised Learning:

Learn $f : \mathcal{X} \to \mathcal{Y}$ from a sample $\{(x_1, y_1), \cdots, (x_N, y_N)\} \subset \mathcal{X} \times \mathcal{Y}$.

Learning with Structured Data

Supervised Learning:

Learn $f : \mathcal{X} \to \mathcal{Y}$ from a sample $\{(x_1, y_1), \cdots, (x_N, y_N)\} \subset \mathcal{X} \times \mathcal{Y}$.

- Classical learning algorithms assume $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}^p$.
- How to handle input/output structured data?

Learning with Structured Data

Supervised Learning:

Learn $f : \mathcal{X} \to \mathcal{Y}$ from a sample $\{(x_1, y_1), \cdots, (x_N, y_N)\} \subset \mathcal{X} \times \mathcal{Y}$.

- Classical learning algorithms assume $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}^p$.
- How to handle input/output structured data?
  - Tensor structured data: Images, videos, spatio-temporal data, ...

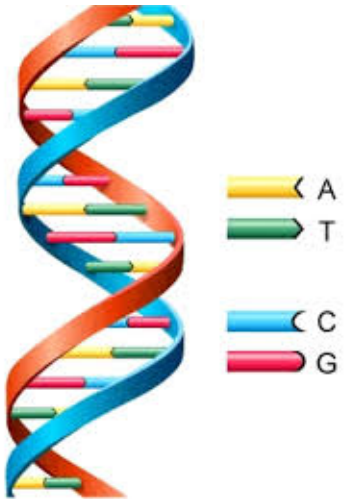
$\in \mathbb{R}^{32\times32\times3} \simeq \mathbb{R}^{3072}$

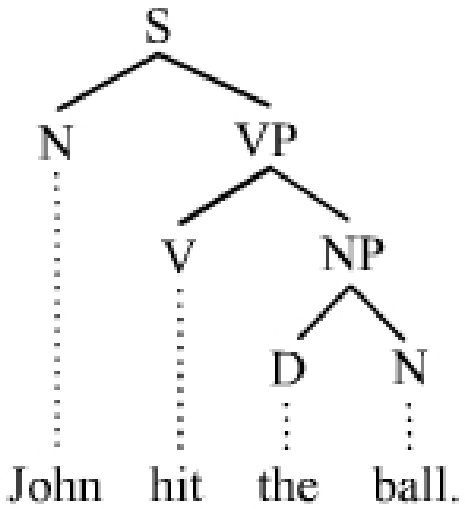
Learning with Structured Data

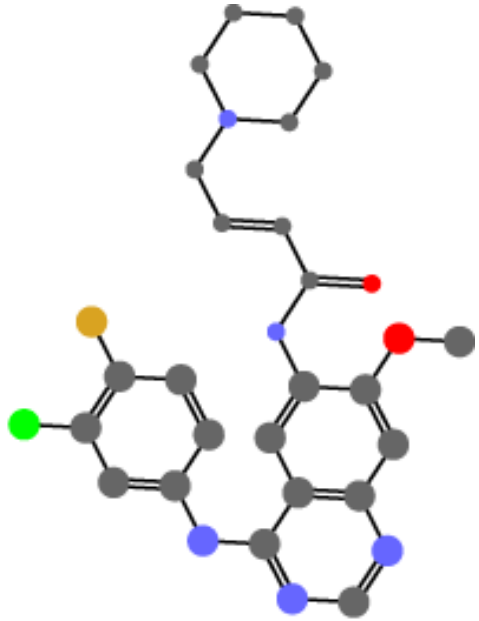
Supervised Learning:

Learn $f : \mathcal{X} \to \mathcal{Y}$ from a sample $\{(x_1, y_1), \cdots, (x_N, y_N)\} \subset \mathcal{X} \times \mathcal{Y}$.

- Classical learning algorithms assume $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}^p$.
- How to handle input/output structured data?
  - Tensor structured data: Images, videos, spatio-temporal data, ...
  - Discrete structured data: strings, trees, graphs, ...







Learning with Structured Data

Supervised Learning:

Learn $f : \mathcal{X} \to \mathcal{Y}$ from a sample $\{(x_1, y_1), \cdots, (x_N, y_N)\} \subset \mathcal{X} \times \mathcal{Y}$.

- Classical learning algorithms assume $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}^p$.
- How to handle input/output structured data?
  - Tensor structured data: Images, videos, spatio-temporal data, ...
  - Discrete structured data: strings, trees, graphs, ...

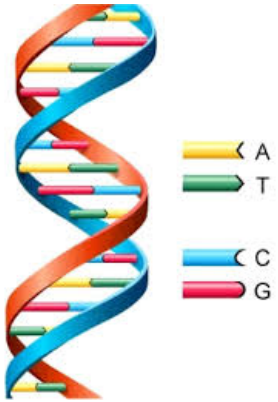
- In both cases, one can leverage linear and tensor algebra to design learning algorithms.

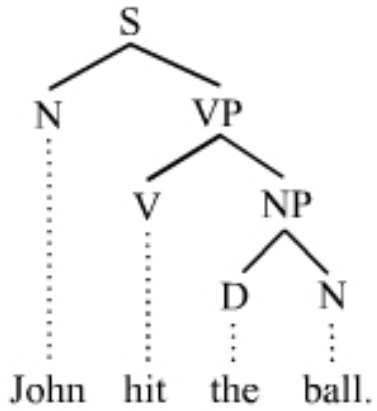
Outline

Weighted Automata (WA) and Spectral Learning

Problem Statement

- How can one learn with structured objects such as strings and trees?

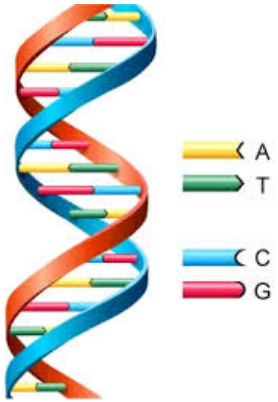


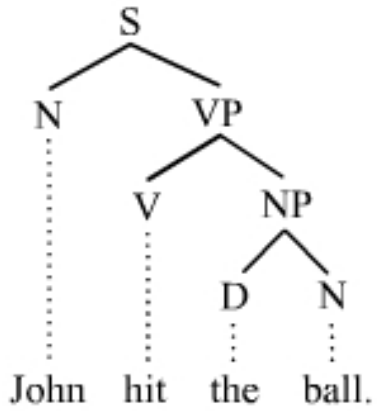


- Intersection of Theoretical Computer Science and Machine Learning...

Problem Statement

- How can one learn with structured objects such as strings and trees?



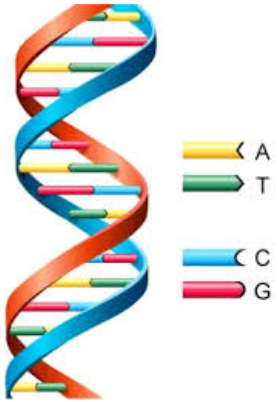


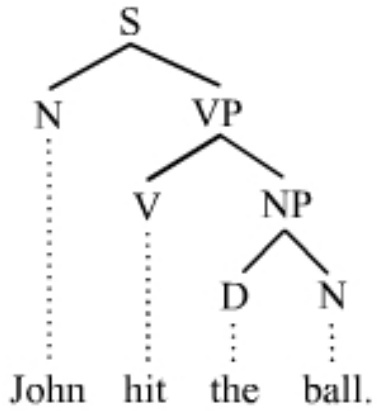
- Intersection of Theoretical Computer Science and Machine Learning...

$\rightarrow$ Weighted Automata: robust model to represent functions defined over structured objects (and in particular probability distributions).

Problem Statement

- How can one learn with structured objects such as strings and trees?





- Intersection of Theoretical Computer Science and Machine Learning...

→ Weighted Automata: robust model to represent functions defined over structured objects (and in particular probability distributions).

- String Weighted Automata (WA): generalize *Hidden Markov Models*, *Predictive State Representations* and closely related to *RNNs*.

String Weighted Automata (WA)

- Σ a finite alphabet (e.g. $\{a, b\}$), Σ^* strings on Σ (e.g. $abba$)
- A WA computes a function $f : \Sigma^* \to \mathbb{R}$

String Weighted Automata (WA)

- Σ a finite alphabet (e.g. $\{a, b\}$), Σ^* strings on Σ (e.g. $abba$)
- A WA computes a function $f : \Sigma^* \to \mathbb{R}$
- Weighted Automaton: $A = (\boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma \in \Sigma}, \boldsymbol{\omega})$ where

$\boldsymbol{\alpha} \in \mathbb{R}^n$ initial weights vector
$\boldsymbol{\omega} \in \mathbb{R}^n$ final weights vector
$\mathbf{A}^\sigma \in \mathbb{R}^{n \times n}$ transition weights matrix for each $\sigma \in \Sigma$

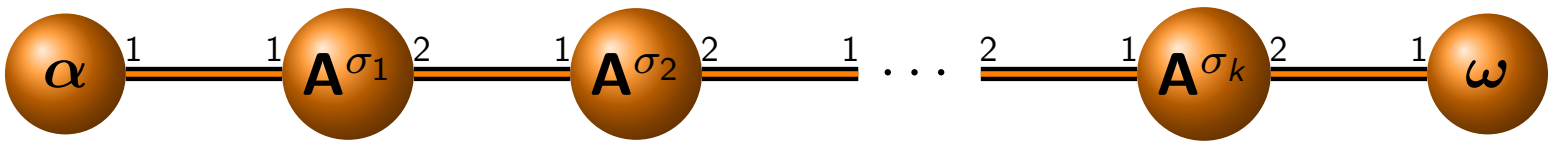
String Weighted Automata (WA)

- Σ a finite alphabet (e.g. $\{a, b\}$), Σ^* strings on Σ (e.g. $abba$)
- A WA computes a function $f : \Sigma^* \to \mathbb{R}$
- Weighted Automaton: $A = (\boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\omega})$ where

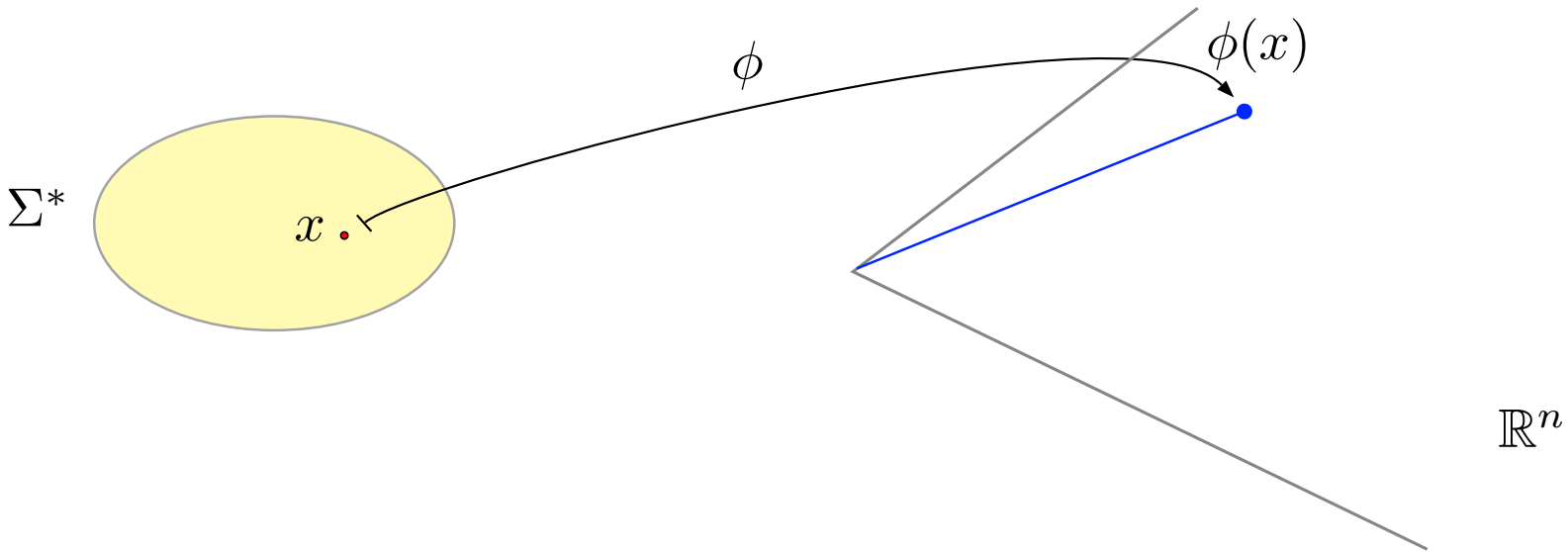
$$\begin{aligned} \boldsymbol{\alpha} &\in \mathbb{R}^n && \text{initial weights vector} \\ \boldsymbol{\omega} &\in \mathbb{R}^n && \text{final weights vector} \\ \mathbf{A}^\sigma &\in \mathbb{R}^{n\times n} && \text{transition weights matrix for each } \sigma \in \Sigma \end{aligned}$$

- A computes a function $f_A : \Sigma^* \to \mathbb{R}$ defined by

$$f_A(\sigma_1\sigma_2\cdots\sigma_k) = \boldsymbol{\alpha}^\top \mathbf{A}^{\sigma_1}\mathbf{A}^{\sigma_2}\cdots\mathbf{A}^{\sigma_k}\boldsymbol{\omega}$$

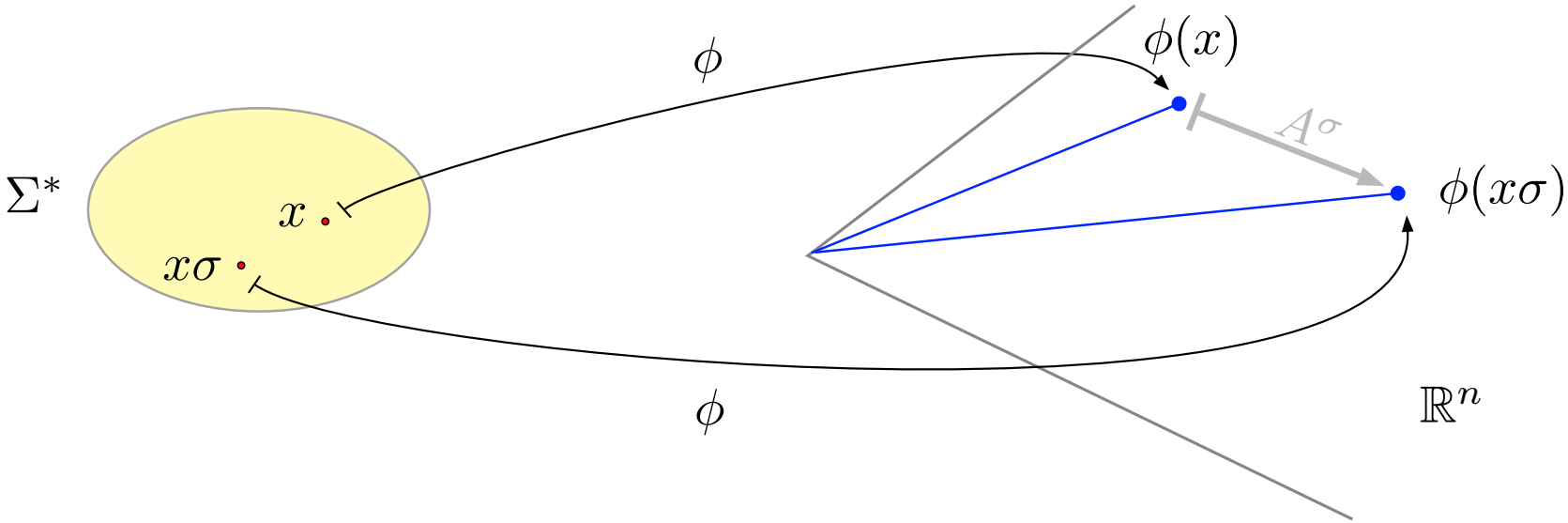


Weighted Automata and Representation Learning



- A WA induces a mapping $\phi : \Sigma^* \to \mathbb{R}^n$ ($\sim$ word embedding)

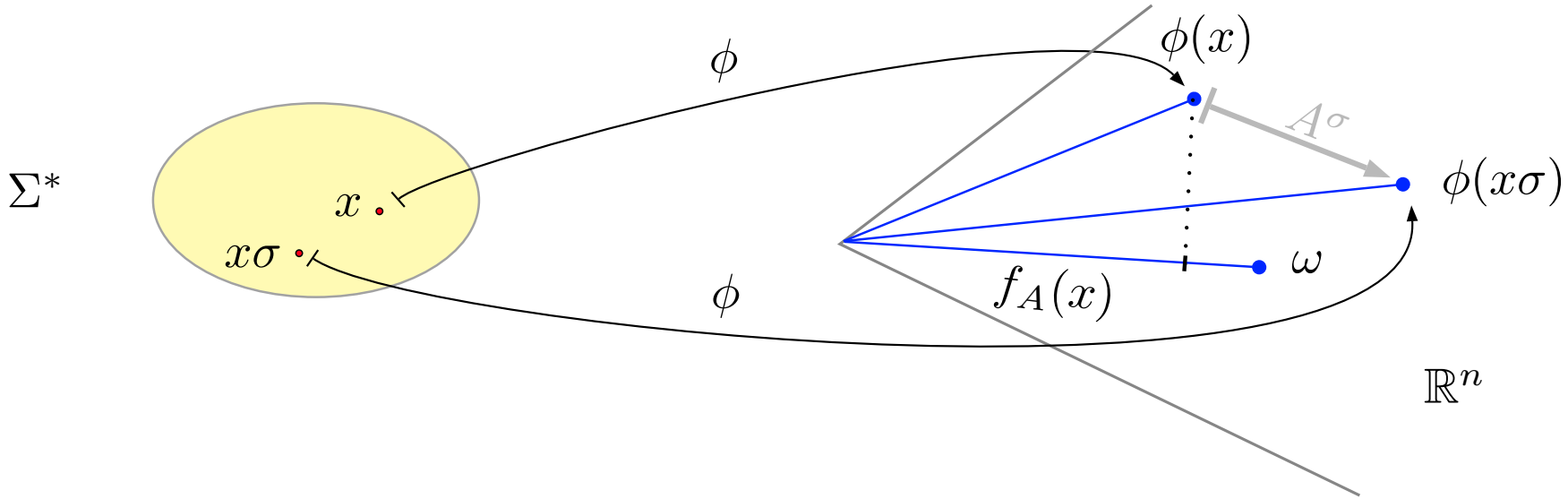
Weighted Automata and Representation Learning



- A WA induces a mapping $\phi : \Sigma^* \to \mathbb{R}^n$ ($\sim$ word embedding)
- The mapping ϕ is compositional:

$$\phi(\lambda) = \boldsymbol{\alpha}^\top, \;\; \phi(\sigma_1) = \boldsymbol{\alpha}^\top \mathbf{A}^{\sigma_1}, \;\; \phi(\sigma_1\sigma_2) = \boldsymbol{\alpha}^\top \mathbf{A}^{\sigma_1}\mathbf{A}^{\sigma_2} = \phi(\sigma_1)\mathbf{A}^{\sigma_2}, \;\; \ldots$$

Weighted Automata and Representation Learning



- A WA induces a mapping $\phi : \Sigma^* \to \mathbb{R}^n$ ($\sim$ word embedding)
- The mapping ϕ is compositional:

$$\phi(\lambda) = \boldsymbol{\alpha}^\top, \;\; \phi(\sigma_1) = \boldsymbol{\alpha}^\top \mathbf{A}^{\sigma_1}, \;\; \phi(\sigma_1\sigma_2) = \boldsymbol{\alpha}^\top \mathbf{A}^{\sigma_1}\mathbf{A}^{\sigma_2} = \phi(\sigma_1)\mathbf{A}^{\sigma_2}, \;\; \ldots$$

- The output $f_A(x) = \langle \phi(x), \boldsymbol{\omega} \rangle$ is linear in $\phi(x)$.

Spectral Learning of Weighted Automata

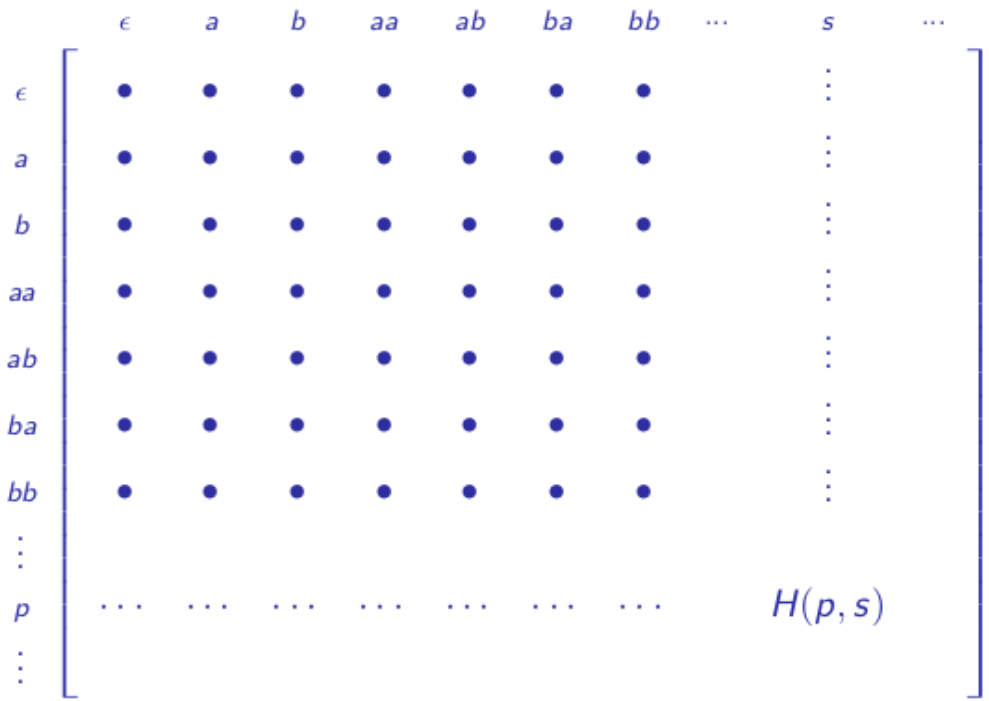
Hankel matrix

$$H \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$$

$$p \cdot s = p' \cdot s' \Rightarrow H(p,s) = H(p',s')$$

$$f : \Sigma^\star \longrightarrow \mathbb{R}$$

$$H_f(p,s) = f(p \cdot s)$$

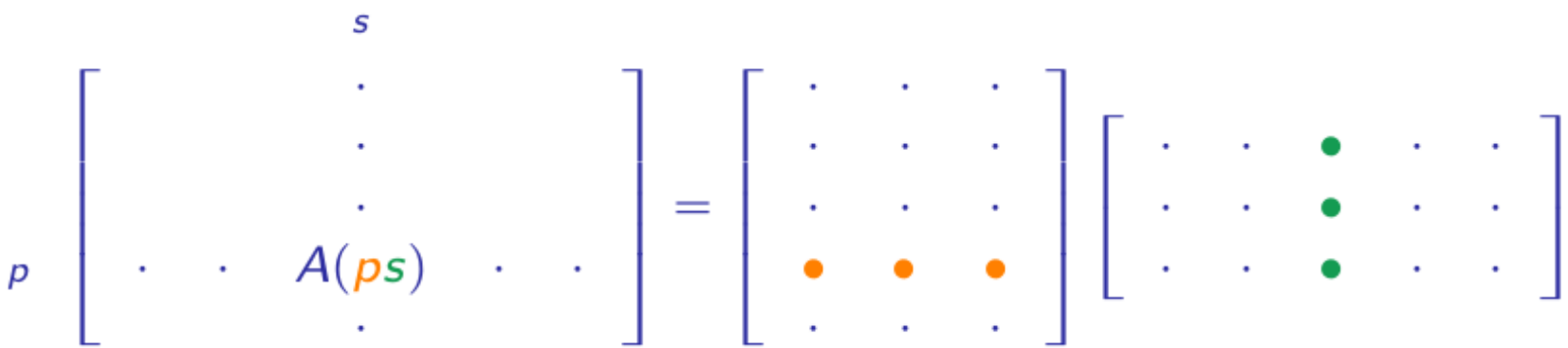


slide credits: Borja Balle

Hankel matrix and WA

Theorem (Fliess '74)
The rank of a *real* Hankel matrix H equals the minimal number of states of a WFA recognizing the weighted language of H

$$A(p_1 \cdots p_t s_1 \cdots s_{t'}) = \alpha^\top A_{p_1} \cdots A_{p_t} A_{s_1} \cdots A_{s_{t'}} \beta$$



slide credits: Borja Balle

Hankel matrix: spectral learning

$$H_a(p,s) = A(pas)$$

$$A(p_1 \cdots p_t a s_1 \cdots s_{t'}) = \alpha^\top A_{p_1} \cdots A_{p_t} A_a A_{s_1} \cdots A_{s_{t'}} \beta$$

$$H = P\ S \qquad H_a = P\ A_a\ S \qquad A_a = P^+\ H_a\ S^+$$

slide credits: Borja Balle

Spectral Learning of Weighted Automata

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

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

- Fundamental theorem [Carlyle and Paz, 1971; Fliess 1974]:

  $\text{rank}(\mathbf{H}_f) < \infty \iff f$ can be computed by a WA

- Proof is constructive $\Rightarrow$ Spectral Learning of WA:
  1. Estimate a sub-block of $\mathbf{H}_f$ from training data
  2. Low rank decomposition $\mathbf{H} \simeq \mathbf{PS}$
  3. Build WA $\hat{A}$ using $\mathbf{H}, \mathbf{P}$ and $\mathbf{S}$.

$\rightarrow$ Efficient and consistent learning algorithms for weighted automata [Hsu et al., 2009; Bailly et al. 2009; Balle et al., 2014, ...].

Multitask Learning of Weighted Automata

Multitask Learning

- Multitask learning: jointly learn multiple related functions.
  - learn to predict rain level, min and max temperatures and sun hours,
  - predict the next word in sentences in French, Spanish and Catalan.
- This work: *Multitask learning of functions defined over sequences.*

Multitask Learning

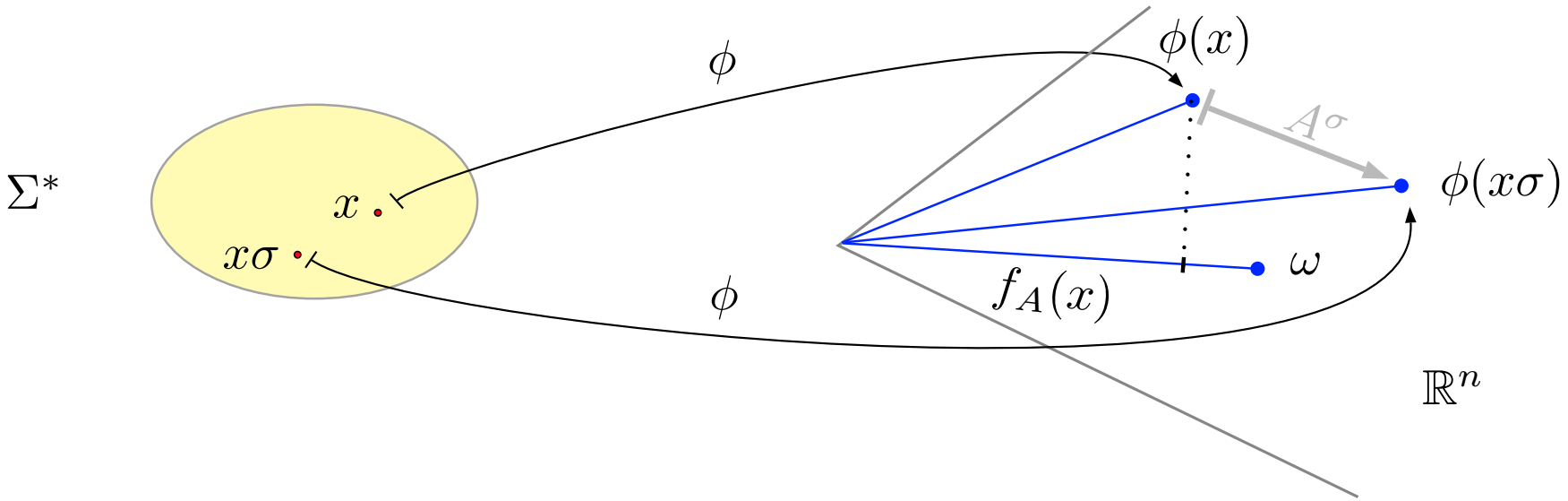
- Multitask learning: jointly learn multiple related functions.
  - learn to predict rain level, min and max temperatures and sun hours,
  - predict the next word in sentences in French, Spanish and Catalan.
- This work: *Multitask learning of functions defined over sequences.*

- Which notion of relatedness?

  $\rightarrow$ Tasks share a *joint representation space*.
- How to extend the spectral learning algorithm to leverage such relatedness?

WAs as Linear Models in a Feature Space

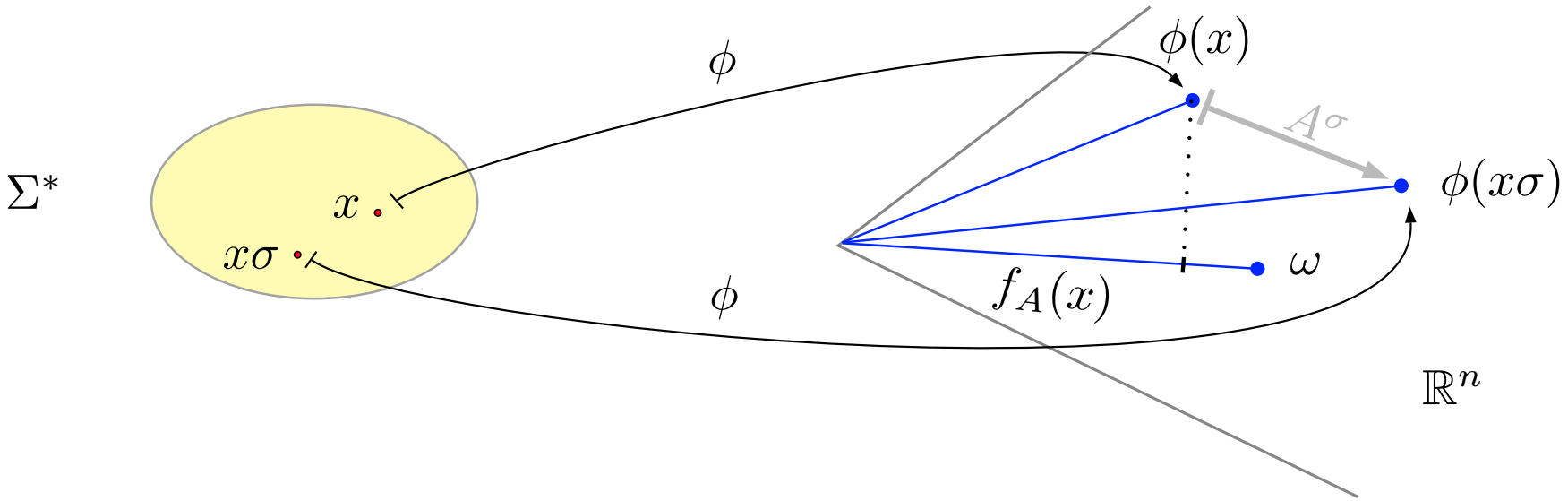
- Computation of a WA A on $x \in \Sigma^*$:
  1. map x to feature vector $\phi(x) = \boldsymbol{\alpha}^\top \mathbf{A}^x$ through a compositional feature map $\phi : \Sigma^* \to \mathbb{R}^n$
  2. compute final value $f_A(x) = \langle \phi(x), \boldsymbol{\omega} \rangle$



- ϕ is compositional: $\phi(x\sigma)^\top = \phi(x)^\top \mathbf{A}^\sigma$.

WAs as Linear Models in a Feature Space

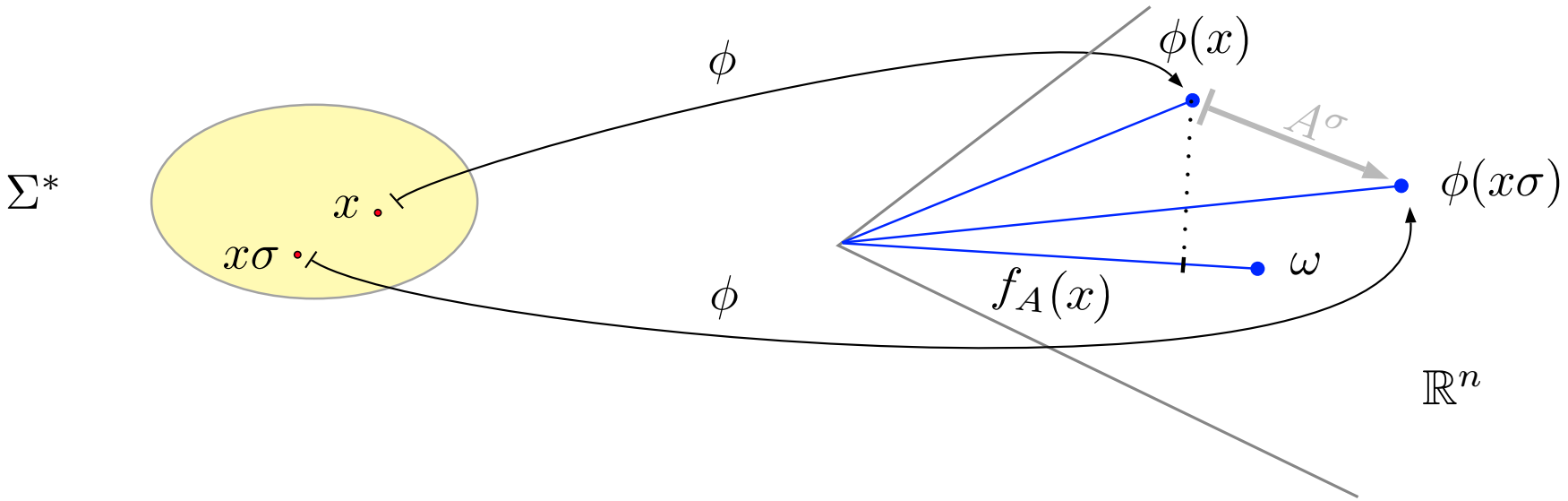
- Computation of a WA A on $x \in \Sigma^*$:
  1. map x to feature vector $\phi(x) = \boldsymbol{\alpha}^\top \mathbf{A}^x$ through a compositional feature map $\phi : \Sigma^* \to \mathbb{R}^n$
  2. compute final value $f_A(x) = \langle \phi(x), \boldsymbol{\omega} \rangle$



- ϕ is compositional: $\phi(x\sigma)^\top = \phi(x)^\top \mathbf{A}^\sigma$.
- ϕ is minimal if $V = \mathrm{span}(\{\phi(x)\}_{x \in \Sigma^*}) \subset \mathbb{R}^n$ is of dimension n.

WAs as Linear Models in a Feature Space

- Computation of a WA A on $x \in \Sigma^*$:
  1. map x to feature vector $\phi(x) = \boldsymbol{\alpha}^\top \mathbf{A}^x$ through a compositional feature map $\phi : \Sigma^* \to \mathbb{R}^n$
  2. compute final value $f_A(x) = \langle \phi(x), \boldsymbol{\omega} \rangle$



- ϕ is compositional: $\phi(x\sigma)^\top = \phi(x)^\top \mathbf{A}^\sigma$.
- ϕ is minimal if $V = \mathrm{span}(\{\phi(x)\}_{x \in \Sigma^*}) \subset \mathbb{R}^n$ is of dimension n.

$\Rightarrow$ $\phi : x \mapsto \boldsymbol{\alpha}^\top \mathbf{A}^x$ is minimal if and only if $(\boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma \in \Sigma}, \boldsymbol{\omega})$ is minimal.

A Notion of Relatedness between Functions on Sequences

Relatedness between WAs: to which extent two WAs can share a joint feature map ϕ:

$$f_1(x) = \langle \phi(x), \boldsymbol{\omega}_1 \rangle \qquad f_2(x) = \langle \phi(x), \boldsymbol{\omega}_2 \rangle$$

A Notion of Relatedness between Functions on Sequences

Relatedness between WAs: to which extent two WAs can share a joint feature map ϕ:

$$f_1(x) = \langle \phi(x), \boldsymbol{\omega}_1 \rangle \qquad f_2(x) = \langle \phi(x), \boldsymbol{\omega}_2 \rangle$$

- Let $f_1, f_2 : \Sigma^* \to \mathbb{R}$ of rank n_1 and n_2. with feature maps $\phi_1 : \Sigma^* \to \mathbb{R}^{n_1}$ and $\phi_2 : \Sigma^* \to \mathbb{R}^{n_2}$.
- $\phi = \phi_1 \oplus \phi_2 : \Sigma^* \to \mathbb{R}^{n_1+n_2}$ is a joint feature map for f_1 and f_2:

$$f_1(x) = \langle \phi(x), \boldsymbol{\omega}_1 \oplus \mathbf{0} \rangle \text{ and } f_2(x) = \langle \phi(x), \mathbf{0} \oplus \boldsymbol{\omega}_2 \rangle$$

A Notion of Relatedness between Functions on Sequences

Relatedness between WAs: to which extent two WAs can share a joint feature map ϕ:

$$f_1(x) = \langle \phi(x), \boldsymbol{\omega}_1 \rangle \qquad f_2(x) = \langle \phi(x), \boldsymbol{\omega}_2 \rangle$$

- Let $f_1, f_2 : \Sigma^* \to \mathbb{R}$ of rank n_1 and n_2. with feature maps $\phi_1 : \Sigma^* \to \mathbb{R}^{n_1}$ and $\phi_2 : \Sigma^* \to \mathbb{R}^{n_2}$.
- $\phi = \phi_1 \oplus \phi_2 : \Sigma^* \to \mathbb{R}^{n_1+n_2}$ is a joint feature map for f_1 and f_2:

$$f_1(x) = \langle \phi(x), \boldsymbol{\omega}_1 \oplus \mathbf{0} \rangle \text{ and } f_2(x) = \langle \phi(x), \mathbf{0} \oplus \boldsymbol{\omega}_2 \rangle$$

  but it may not be minimal.

$\to$ there may exist another feature map of dimension $n < n_1 + n_2$.

A Notion of Relatedness between Functions on Sequences

Relatedness between WAs: to which extent two WAs can share a joint feature map ϕ:

$$f_1(x) = \langle \phi(x), \boldsymbol{\omega}_1 \rangle \qquad f_2(x) = \langle \phi(x), \boldsymbol{\omega}_2 \rangle$$

- Let $f_1, f_2 : \Sigma^* \to \mathbb{R}$ of rank n_1 and n_2. with feature maps $\phi_1 : \Sigma^* \to \mathbb{R}^{n_1}$ and $\phi_2 : \Sigma^* \to \mathbb{R}^{n_2}$.
- $\phi = \phi_1 \oplus \phi_2 : \Sigma^* \to \mathbb{R}^{n_1+n_2}$ is a joint feature map for f_1 and f_2:

$$f_1(x) = \langle \phi(x), \boldsymbol{\omega}_1 \oplus \mathbf{0} \rangle \text{ and } f_2(x) = \langle \phi(x), \mathbf{0} \oplus \boldsymbol{\omega}_2 \rangle$$

  but it may not be minimal.

$\to$ there may exist another feature map of dimension $n < n_1 + n_2$.

- The smaller n is, the more related f_1 and f_2 are.

Vector-Valued WA

- A d-dimensional vector-valued weighted finite automaton (vv-WA) with n states is a tuple $A = (\boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\Omega})$ where
  - $\boldsymbol{\alpha} \in \mathbb{R}^n$ is the initial weights vector
  - $\boldsymbol{\Omega} \in \mathbb{R}^{n\times d}$ is the matrix of final weights
  - $\mathbf{A}^\sigma \in \mathbb{R}^{n\times n}$ is the transition matrix for each $\sigma \in \Sigma$.
- A vv-WA computes a function $\vec{f}_A : \Sigma^* \to \mathbb{R}^d$ defined for each word $x = x_1 x_2 \cdots x_k \in \Sigma^*$ by

$$\vec{f}_A(x_1 x_2 \cdots x_k) = \boldsymbol{\alpha}^\top \mathbf{A}^{x_1} \mathbf{A}^{x_2} \cdots \mathbf{A}^{x_k} \boldsymbol{\Omega} = \boldsymbol{\alpha}^\top \mathbf{A}^{x} \boldsymbol{\Omega}.$$

Vector-Valued WA

- A d-dimensional vector-valued weighted finite automaton (vv-WA) with n states is a tuple $A = (\boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\Omega})$ where
  - $\boldsymbol{\alpha} \in \mathbb{R}^n$ is the initial weights vector
  - $\boldsymbol{\Omega} \in \mathbb{R}^{n\times d}$ is the matrix of final weights
  - $\mathbf{A}^\sigma \in \mathbb{R}^{n\times n}$ is the transition matrix for each $\sigma \in \Sigma$.
- A vv-WA computes a function $\vec{f}_A : \Sigma^* \to \mathbb{R}^d$ defined for each word $x = x_1 x_2 \cdots x_k \in \Sigma^*$ by

$$\vec{f}_A(x_1 x_2 \cdots x_k) = \boldsymbol{\alpha}^\top \mathbf{A}^{x_1} \mathbf{A}^{x_2} \cdots \mathbf{A}^{x_k} \boldsymbol{\Omega} = \boldsymbol{\alpha}^\top \mathbf{A}^{x} \boldsymbol{\Omega}.$$

$\Rightarrow$ Rank of $\vec{f} = [f_1, f_2] : \Sigma^* \to \mathbb{R}^2$ equal dimension of a minimal joint feature map for f_1 and f_2.

Vector-Valued WA

- A d-dimensional vector-valued weighted finite automaton (vv-WA) with n states is a tuple $A = (\boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\Omega})$ where
  - $\boldsymbol{\alpha} \in \mathbb{R}^n$ is the initial weights vector
  - $\boldsymbol{\Omega} \in \mathbb{R}^{n\times d}$ is the matrix of final weights
  - $\mathbf{A}^\sigma \in \mathbb{R}^{n\times n}$ is the transition matrix for each $\sigma \in \Sigma$.
- A vv-WA computes a function $\vec{f}_A : \Sigma^* \to \mathbb{R}^d$ defined for each word $x = x_1 x_2 \cdots x_k \in \Sigma^*$ by

$$\vec{f}_A(x_1 x_2 \cdots x_k) = \boldsymbol{\alpha}^\top \mathbf{A}^{x_1} \mathbf{A}^{x_2} \cdots \mathbf{A}^{x_k} \boldsymbol{\Omega} = \boldsymbol{\alpha}^\top \mathbf{A}^{x} \boldsymbol{\Omega}.$$

$\Rightarrow$ Rank of $\vec{f} = [f_1, f_2] : \Sigma^* \to \mathbb{R}^2$ equal dimension of a minimal joint feature map for f_1 and f_2.

$\Rightarrow$ $\max\{\text{rank}(f_1), \text{rank}(f_2)\} \leq \text{rank}([f_1, f_2]) \leq \text{rank}(f_1) + \text{rank}(f_2)$.

Example

- Consider the following count functions:

$$\begin{cases} f_1(x) = 0.5|x|_a + 0.5|x|_b \\ f_2(x) = 0.3|x|_b - 0.6|x|_c \\ f_3(x) = |x|_c \end{cases}$$

Example

- Consider the following count functions:

$$\begin{cases} f_1(x) = 0.5|x|_a + 0.5|x|_b \\ f_2(x) = 0.3|x|_b - 0.6|x|_c \\ f_3(x) = |x|_c \end{cases}$$

- We have
  - $\text{rank}(f_2) = 4 = \text{rank}([f_2, f_3])$
  - $\text{rank}([f_1, f_3]) = 6 = \text{rank}(f_1) + \text{rank}(f_3)$
  - $\text{rank}(f_1) = \text{rank}(f_2) < \text{rank}([f_1, f_2]) < \text{rank}(f_1) + \text{rank}(f_2)$

Spectral Learning of Vector-Valued Weighted Automata

Spectral Learning of vv-WAs

- Hankel tensor $\mathcal{H} \in \mathbb{R}^{\Sigma^* \times d \times \Sigma^*}$ associated with a function $\vec{f} : \Sigma^* \to \mathbb{R}^d$

$$\mathcal{H}_{u,:,v} = \vec{f}(uv) \quad \text{for all} \quad u, v \in \Sigma^*.$$

Theorem [Vector-Valued Fliess Theorem] *For any* $\vec{f} : \Sigma^* \to \mathbb{R}^d$, $\text{rank}(\vec{f}) = \text{rank}(\mathcal{H}_{(1)})$, *where* $\mathcal{H}_{(1)} = [\mathcal{H}_{:,1,:} \;\; \mathcal{H}_{:,2,:} \;\; \cdots \;\; \mathcal{H}_{:,d,:}]$ *is the flattening of the Hankel tensor.*

Spectral Learning of vv-WAs

- Hankel tensor $\mathcal{H} \in \mathbb{R}^{\Sigma^* \times d \times \Sigma^*}$ associated with a function $\vec{f} : \Sigma^* \to \mathbb{R}^d$

$$\mathcal{H}_{u,:,v} = \vec{f}(uv) \quad \text{for all} \quad u, v \in \Sigma^*.$$

Theorem [Vector-Valued Fliess Theorem] *For any* $\vec{f} : \Sigma^* \to \mathbb{R}^d$, $\text{rank}(\vec{f}) = \text{rank}(\mathcal{H}_{(1)})$, *where* $\mathcal{H}_{(1)} = [\mathcal{H}_{:,1,:} \;\; \mathcal{H}_{:,2,:} \;\; \cdots \;\; \mathcal{H}_{:,d,:}]$ *is the flattening of the Hankel tensor.*

- Spectral learning of vv-WAs. A vv-WA computing $\vec{f}$ can be recovered from any rank n factorization of $\mathcal{H}_{(1)}$:
  1. Let $\mathcal{H}_{(1)} = \mathbf{P}\mathcal{S}_{(1)}$ with $\mathbf{P} \in \mathbb{R}^{\Sigma^* \times n}$ and $\mathcal{S} \in \mathbb{R}^{n \times d \times \Sigma^*}$.
  2. For each $\sigma \in \Sigma$, let $\mathcal{H}^\sigma \in \mathbb{R}^{\Sigma^* \times d \times \Sigma^*}$ be defined by $\mathcal{H}^\sigma_{u,:,v} = \vec{f}(u\sigma v)$ for all $u, v \in \Sigma^*$.
  3. The vv-WA $A = (\boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma \in \Sigma}, \boldsymbol{\Omega})$ where $\boldsymbol{\alpha}^\top = \mathbf{P}_{\lambda,:}$, $\boldsymbol{\Omega} = \mathcal{S}_{:,:,\lambda}$, and $\mathbf{A}^\sigma = \mathbf{P}^\dagger \mathcal{H}^\sigma_{(1)} (\mathcal{S}_{(1)})^\dagger$ is a minimal vv-WA for $\vec{f}$.

Experiments

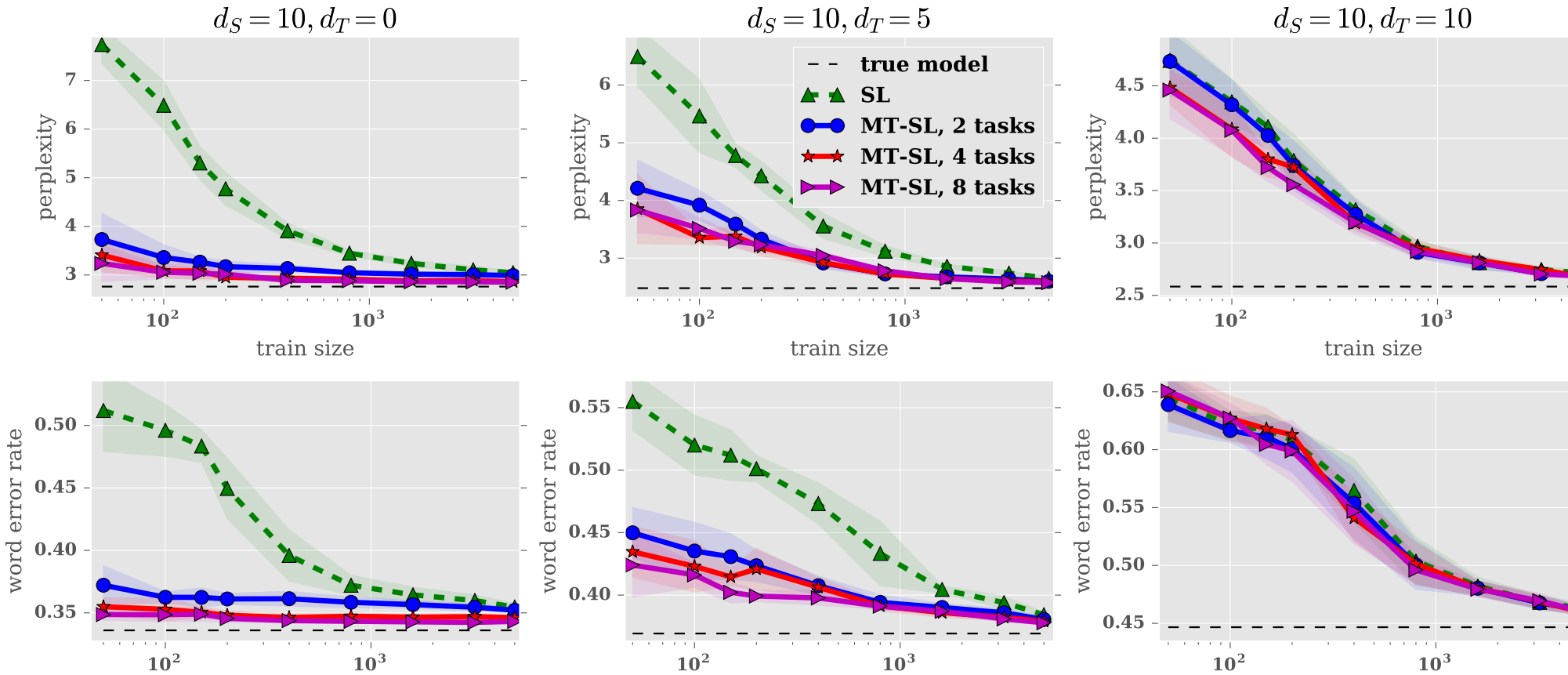
Experiments

- We compare MT-SL with classical spectral learning (SL).
- Evaluation metrics:
  - Perplexity per character: $\mathrm{perp}(h) = 2^{-\frac{1}{M}\sum_{x\in T}\log(h(x))}$ where M is the number of symbols in the test set T.
  - Word error rate (WER): proportion of mis-predicted symbols averaged over all prefixes in the test set (when the most likely symbol is predicted).

Synthetic Data

- Randomly generated stochastic WAs following the PAutomaC competition process [Verwer et al., 2012].
- Related WAs: joint feature space of dimension $d_S = 10$ and task specific space of dimension d_T (i.e. $\text{rank}(f_i) = d_S + d_T$ and $\text{rank}(\vec{f}) = \text{rank}([f_1, \cdots, f_m]) = d_S + md_T$).
- Training sample drawn from target task f_1 and training samples of size $5,000$ for tasks $f_2, \cdots, f_m$.

Synthetic Data



Real Data

- Universal Dependencies treebank [Nivre et al., 2016]: sentences from 33 languages labeled with 17 PoS tags.

$\Rightarrow$ Samples drawn from 33 distributions over strings on an alphabet of size 17.

- For each language, $(80\%, 10\%, 10\%)$-split between training, validation and test sets.
- Two ways of selecting related tasks:
  1. use all other languages
  2. select the 4 closest languages w.r.t. the distance between the (top-50) left singular subspaces of the Hankel matrices.

Real Data (cont'd)

Training size	100	500	1000	5000	all available data
	Related tasks: all other languages				
Perplexity	7.0744 (±7.76)	3.6666 (±5.22)	3.2879 (±5.17)	3.4187 (±5.57)	3.1574 (±5.48)
WER	1.4919 (±2.37)	1.3786 (±2.94)	1.2281 (±2.62)	1.4964 (±2.70)	1.4932 (±2.77)
	Related tasks: 4 closest languages				
Perplexity	6.0069 (±6.76)	4.3670 (±5.83)	4.4049 (±5.50)	2.9689 (±5.87)	2.8229 (±5.90)
WER	2.0883 (±3.26)	1.5175 (±2.87)	1.2961 (±2.57)	1.3080 (±2.55)	1.2160 (±2.31)

Table: Average relative improvement over all languages (in %) of MT-SL vs. SL on the UNIDEP dataset (e.g. for perplexity we report $100 \cdot (p_{\mathrm{SL}} - p_{\mathrm{MT-SL}})/p_{\mathrm{SL}}$).

- Cherry picked example: on the Basque task with a training set of size 500, the WER was reduced from $\sim 77\%$ for SL to $\sim 71\%$ using all other languages as related tasks, and to $\sim 68\%$ using the 4 closest tasks (Finnish, Polish, Czech and Indonesian).

Real Data (cont'd)

Target task	4 closest tasks w.r.t. subspace distance (closest first)			
Basque	Finnish	Polish	Czech	Indonesian
Croatian	Estonian	Slovenian	Czech	Finnish
French	Italian	Spanish	German	English
Hungarian	Danish	Ancient Greek	German	Portuguese
Gothic	Old Church Slavonic	Latin	Ancient Greek	Finnish
Italian	English	French	Spanish	Dutch
Japanese	Hindi	Persian	Arabic	Tamil
Latin	Old Church Slavonic	Ancient Greek	Gothic	Finnish
Swedish	Danish	Norwegian	Finnish	Estonian

Table: Some related tasks used in the UNIDEP experiment.

Conclusion

Conclusion

- Multitask extension of the spectral learning algorithm.
  - A bit of theoretical analysis and experiment details in the paper.
- "Novel" model of vector-valued weighted automata.

Conclusion

- Multitask extension of the spectral learning algorithm.
  - A bit of theoretical analysis and experiment details in the paper.
- "Novel" model of vector-valued weighted automata.

- Potential applications in reinforcement learning.
- Extension to weighted tree automata should be easy.

Conclusion

- Multitask extension of the spectral learning algorithm.
  - A bit of theoretical analysis and experiment details in the paper.
- "Novel" model of vector-valued weighted automata.

- Potential applications in reinforcement learning.
- Extension to weighted tree automata should be easy.

Thank you! Questions?

Joakim Nivre, Zeljko Agić, Lars Ahrenberg, et al. Universal dependencies 1.4, 2016. URL `http://hdl.handle.net/11234/1-1827`. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Sicco Verwer, Rémi Eyraud, and Colin De La Higuera. Results of the pautomac probabilistic automaton learning competition. In **ICGI**, pages 243–248, 2012.