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 $f : \mathcal{X} \rightarrow \mathcal{Y}$ 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.
Learning with Structured Data

Supervised Learning:

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

- 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, ...

\[ \in \mathbb{R}^{32 \times 32 \times 3} \simeq \mathbb{R}^{3072} \]
Learning with Structured Data

Supervised Learning:

Learn $f : \mathcal{X} \rightarrow \mathcal{Y}$ 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, ...

![DNA structure](image1)

![Gene structure](image2)
Learning with Structured Data

Supervised Learning:

Learn $f : \mathcal{X} \rightarrow \mathcal{Y}$ 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.
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...

→ 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)

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

- $\Sigma$ a finite alphabet (e.g. $\{a, b\}$), $\Sigma^*$ strings on $\Sigma$ (e.g. $abba$)
- A WA computes a function $f : \Sigma^* \rightarrow \mathbb{R}$
- Weighted Automaton: $A = (\alpha, \{A^\sigma\}_{\sigma \in \Sigma}, \omega)$ where
  - $\alpha \in \mathbb{R}^n$ initial weights vector
  - $\omega \in \mathbb{R}^n$ final weights vector
  - $A^\sigma \in \mathbb{R}^{n \times n}$ transition weights matrix for each $\sigma \in \Sigma$
String Weighted Automata (WA)

- $\Sigma$ a finite alphabet (e.g. $\{a, b\}$), $\Sigma^*$ strings on $\Sigma$ (e.g. $abba$)
- A WA computes a function $f : \Sigma^* \rightarrow \mathbb{R}$
- Weighted Automaton: $A = (\alpha, \{A^\sigma\}_{\sigma \in \Sigma}, \omega)$ where
  - $\alpha \in \mathbb{R}^n$ initial weights vector
  - $\omega \in \mathbb{R}^n$ final weights vector
  - $A^\sigma \in \mathbb{R}^{n \times n}$ transition weights matrix for each $\sigma \in \Sigma$
- $A$ computes a function $f_A : \Sigma^* \rightarrow \mathbb{R}$ defined by
  $$f_A(\sigma_1\sigma_2 \cdots \sigma_k) = \alpha^\top A^{\sigma_1} A^{\sigma_2} \cdots A^{\sigma_k} \omega$$
A WA induces a mapping \( \phi : \Sigma^* \rightarrow \mathbb{R}^n \) (\( \sim \) word embedding)
- A WA induces a mapping \( \phi : \Sigma^* \rightarrow \mathbb{R}^n \) (\( \sim \) word embedding)
- The mapping \( \phi \) is compositional:

\[
\phi(\lambda) = \alpha^\top, \quad \phi(\sigma_1) = \alpha^\top A^{\sigma_1}, \quad \phi(\sigma_1 \sigma_2) = \alpha^\top A^{\sigma_1} A^{\sigma_2} = \phi(\sigma_1) A^{\sigma_2}, \ldots
\]
A WA induces a mapping $\phi : \Sigma^* \rightarrow \mathbb{R}^n$ (\sim word embedding)

The mapping $\phi$ is compositional:

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

The output $f_A(x) = \langle \phi(x), \omega \rangle$ is linear in $\phi(x)$. 
Spectral Learning of Weighted Automata
Hankel matrix

\[ H \in \mathbb{R}^{\Sigma^* \times \Sigma^*} \]

\[ p \cdot s = p' \cdot s' \Rightarrow H(p, s) = H(p', s') \]

\[ f : \Sigma^* \rightarrow \mathbb{R} \]

\[ H_f(p, s) = f(p \cdot s) \]
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 $H$.

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

\[ H_a(p, s) = A(pas) \]

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

\[ \begin{bmatrix}
  \vdots \\
  \vdots \\
  p \\
  \vdots \\
 \end{bmatrix} \begin{bmatrix}
  \vdots \\
  \vdots \\
  A(pas) \\
  \vdots \\
 \end{bmatrix} = \begin{bmatrix}
  \vdots \\
  \vdots \\
  \vdots \\
  \vdots \\
 \end{bmatrix} \begin{bmatrix}
  \vdots \\
  \vdots \\
  \vdots \\
  \vdots \\
 \end{bmatrix} \begin{bmatrix}
  \vdots \\
  \vdots \\
  \vdots \\
  \vdots \\
 \end{bmatrix} \]

\[ H = P S \quad H_a = P A_a S \quad A_a = P^+ H_a S^+ \]
Spectral Learning of Weighted Automata

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

  \[ (H_f)_{p,s} = f(ps) \]

  \textit{Definition:} prefix $p$, suffix $s$ \Rightarrow \begin{pmatrix} H_f \end{pmatrix}_{p,s} = f(ps)

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

  \[ \text{rank}(H_f) < \infty \iff f \text{ can be computed by a WA} \]

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

  \rightarrow \text{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: jointely 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?
  - 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 $\phi(x) = \alpha^T A^x$ through a compositional feature map $\phi : \Sigma^* \rightarrow \mathbb{R}^n$
  2. compute final value $f_A(x) = \langle \phi(x), \omega \rangle$

\[ \phi(x) \quad \phi(x\sigma) \quad f_A(x) \quad \omega \]

- $\phi$ is compositional: $\phi(x\sigma)^T = \phi(x)^T A^\sigma$. 
WAs as Linear Models in a Feature Space

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

- $\phi$ is compositional: $\phi(x\sigma)^\top = \phi(x)^\top A^\sigma$.
- $\phi$ is minimal if $V = \text{span}(\{\phi(x)\}_{x \in \Sigma^*}) \subset \mathbb{R}^n$ 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 $\phi(x) = \alpha^\top A^x$ through a compositional feature map $\phi: \Sigma^* \rightarrow \mathbb{R}^n$
  2. compute final value $f_A(x) = \langle \phi(x), \omega \rangle$

- $\phi$ is compositional: $\phi(x\sigma)^\top = \phi(x)^\top A^\sigma$.
- $\phi$ is minimal if $V = \text{span}(\{\phi(x)\}_{x \in \Sigma^*}) \subset \mathbb{R}^n$ is of dimension $n$.
  $\Rightarrow \phi : x \mapsto \alpha^\top A^x$ is minimal if and only if $(\alpha, \{A^\sigma\}_{\sigma \in \Sigma}, \omega)$ is minimal.
A Notion of Relatedness between Functions on Sequences

Relatedness between WAs: to which extent two WAs can share a joint feature map \( \phi \):

\[
\begin{align*}
f_1(x) &= \langle \phi(x), \omega_1 \rangle \\
f_2(x) &= \langle \phi(x), \omega_2 \rangle
\end{align*}
\]
A Notion of Relatedness between Functions on Sequences

Relatedness between WAs: to which extent two WAs can share a joint feature map $\phi$:

$$f_1(x) = \langle \phi(x), \omega_1 \rangle \quad f_2(x) = \langle \phi(x), \omega_2 \rangle$$

- Let $f_1, f_2 : \Sigma^* \rightarrow \mathbb{R}$ of rank $n_1$ and $n_2$. with feature maps
  $\phi_1 : \Sigma^* \rightarrow \mathbb{R}^{n_1}$ and $\phi_2 : \Sigma^* \rightarrow \mathbb{R}^{n_2}$.
- $\phi = \phi_1 \oplus \phi_2 : \Sigma^* \rightarrow \mathbb{R}^{n_1+n_2}$ is a joint feature map for $f_1$ and $f_2$:
  $$f_1(x) = \langle \phi(x), \omega_1 \oplus 0 \rangle \quad \text{and} \quad f_2(x) = \langle \phi(x), 0 \oplus \omega_2 \rangle$$
A Notion of Relatedness between Functions on Sequences

Relatedness between WAs: to which extent two WAs can share a joint feature map $\phi$:

$$f_1(x) = \langle \phi(x), \omega_1 \rangle \quad f_2(x) = \langle \phi(x), \omega_2 \rangle$$

- 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 $f_2$:

$$f_1(x) = \langle \phi(x), \omega_1 \oplus 0 \rangle \quad \text{and} \quad f_2(x) = \langle \phi(x), 0 \oplus \omega_2 \rangle$$

but it may not be minimal.

$\rightarrow$ 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 $\phi$:

$$f_1(x) = \langle \phi(x), \omega_1 \rangle \quad f_2(x) = \langle \phi(x), \omega_2 \rangle$$

- 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 $f_2$:

$$f_1(x) = \langle \phi(x), \omega_1 \oplus \mathbf{0} \rangle \quad \text{and} \quad f_2(x) = \langle \phi(x), \mathbf{0} \oplus \omega_2 \rangle$$

but it may not be minimal.

→ 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.
A $d$-dimensional vector-valued weighted finite automaton (vv-WA) with $n$ states is a tuple $A = (\alpha, \{A^\sigma\}_{\sigma \in \Sigma}, \Omega)$ where

- $\alpha \in \mathbb{R}^n$ is the initial weights vector
- $\Omega \in \mathbb{R}^{n \times d}$ is the matrix of final weights
- $A^\sigma \in \mathbb{R}^{n \times n}$ is the transition matrix for each $\sigma \in \Sigma$.

A vv-WA computes a function $\vec{f}_A : \Sigma^* \rightarrow \mathbb{R}^d$ defined for each word $x = x_1 x_2 \cdots x_k \in \Sigma^*$ by

$$ \vec{f}_A(x_1 x_2 \cdots x_k) = \alpha^\top A^{x_1} A^{x_2} \cdots A^{x_k} \Omega = \alpha^\top A^x \Omega. $$
Vector-Valued WA

- A $d$-dimensional vector-valued weighted finite automaton (vv-WA) with $n$ states is a tuple $A = (\alpha, \{A^\sigma\}_{\sigma \in \Sigma}, \Omega)$ where
  - $\alpha \in \mathbb{R}^n$ is the initial weights vector
  - $\Omega \in \mathbb{R}^{n \times d}$ is the matrix of final weights
  - $A^\sigma \in \mathbb{R}^{n \times n}$ is the transition matrix for each $\sigma \in \Sigma$.

- A vv-WA computes a function $\vec{f}_A : \Sigma^* \rightarrow \mathbb{R}^d$ defined for each word $x = x_1x_2 \cdots x_k \in \Sigma^*$ by

$$\vec{f}_A(x_1x_2 \cdots x_k) = \alpha^\top A^{x_1}A^{x_2} \cdots A^{x_k}\Omega = \alpha^\top A^x\Omega.$$ 

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

Guillaume Rabusseau

Multitask spectral learning of WA

July 16, 2018 17 / 28
Vector-Valued WA

- A $d$-dimensional vector-valued weighted finite automaton ($\text{vv-WA}$) with $n$ states is a tuple $A = (\alpha, \{A^\sigma\}_{\sigma \in \Sigma}, \Omega)$ where
  - $\alpha \in \mathbb{R}^n$ is the initial weights vector
  - $\Omega \in \mathbb{R}^{n \times d}$ is the matrix of final weights
  - $A^\sigma \in \mathbb{R}^{n \times n}$ is the transition matrix for each $\sigma \in \Sigma$.

- A vv-WA computes a function $\vec{f}_A : \Sigma^* \rightarrow \mathbb{R}^d$ defined for each word $x = x_1 x_2 \cdots x_k \in \Sigma^*$ by
  \[
  \vec{f}_A(x_1 x_2 \cdots x_k) = \alpha^\top A^{x_1} A^{x_2} \cdots A^{x_k} \Omega = \alpha^\top A^x \Omega.
  \]

$\Rightarrow$ Rank of $\vec{f} = [f_1, f_2] : \Sigma^* \rightarrow \mathbb{R}^2$ 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:

\[
\begin{align*}
  f_1(x) &= 0.5|x|_a + 0.5|x|_b \\
  f_2(x) &= 0.3|x|_b - 0.6|x|_c \\
  f_3(x) &= |x|_c
\end{align*}
\]
Example

Consider the following count functions:

\[
\begin{align*}
    f_1(x) &= 0.5 |x|_a + 0.5 |x|_b \\
    f_2(x) &= 0.3 |x|_b - 0.6 |x|_c \\
    f_3(x) &= |x|_c
\end{align*}
\]

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 \( \vec{f} : \Sigma^* \rightarrow \mathbb{R}^d \)

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

**Theorem** [Vector-Valued Fliess Theorem] For any \( \vec{f} : \Sigma^* \rightarrow \mathbb{R}^d \), \( \text{rank}(\vec{f}) = \text{rank}(\mathcal{H}(1)) \), where \( \mathcal{H}(1) = [\mathcal{H}_{,:1,:} \quad \mathcal{H}_{,:2,:} \quad \cdots \quad \mathcal{H}_{,:d,:}] \) 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 $\vec{f} : \Sigma^* \rightarrow \mathbb{R}^d$

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

**Theorem** [Vector-Valued Fliess Theorem] For any $\vec{f} : \Sigma^* \rightarrow \mathbb{R}^d$, $\text{rank}(\vec{f}) = \text{rank}(\mathcal{H}_{(1)})$, 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 $\mathcal{H}_{(1)} = \mathbf{PS}_{(1)}$ with $\mathbf{P} \in \mathbb{R}^{\Sigma^* \times n}$ and $\mathbf{S} \in \mathbb{R}^{n \times d \times \Sigma^*}$.
  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 $A = (\alpha, \{A^\sigma\}_{\sigma \in \Sigma}, \Omega)$ where $\alpha^\top = \mathbf{P}_{\lambda,:}, \Omega = \mathbf{S}_{:,\lambda},$ and $A^\sigma = \mathbf{P}^\dagger \mathcal{H}^\sigma_{(1)}(\mathbf{S}_{(1)})^\dagger$ is a minimal vv-WA for $\vec{f}.$
Experiments
Experiments

- We compare MT-SL with classical spectral learning (SL).

**Evaluation metrics:**

- Perplexity per character: $\text{perp}(h) = 2^{-\frac{1}{M}} \sum_{x \in T} \log(h(x))$ 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).
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

$ds = 10, d_T = 0$

$ds = 10, d_T = 5$

$ds = 10, d_T = 10$

Guillaume Rabusseau  Multitask spectral learning of WA  July 16, 2018  24 / 28
Real Data

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

⇒ 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)

<table>
<thead>
<tr>
<th>Training size</th>
<th>100</th>
<th>500</th>
<th>1000</th>
<th>5000</th>
<th>all available data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Perplexity</strong></td>
<td>7.0744 (±7.76)</td>
<td>3.6666 (±5.22)</td>
<td>3.2879 (±5.17)</td>
<td>3.4187 (±5.57)</td>
<td>3.1574 (±5.48)</td>
</tr>
<tr>
<td><strong>WER</strong></td>
<td>1.4919 (±2.37)</td>
<td>1.3786 (±2.94)</td>
<td>1.2281 (±2.62)</td>
<td>1.4964 (±2.70)</td>
<td>1.4932 (±2.77)</td>
</tr>
</tbody>
</table>

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

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

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

**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?