Spectral Learning of Weighted Automata: from theory to practice

Rémi Eyraud

QARMA team, Laboratoire d'Informatique et Système, Marseille, France

Journées annuelles du GT Vérification 2018

Context

- 10+ years of research:
  - Premise: [François Denis, Aurélien Lemay, Alain Terlutte. Learning regular languages using RFSAs. 2004]
  - Breakthrough:
    [Raphaël Bailly, François Denis, Liva Ralaivola: Grammatical inference as a principal component analysis problem. 2009]
    and
    [Daniel Hsu, Sham M. Kakade, Tong Zhang. A Spectral Algorithm for Learning Hidden Markov Models. 2009]
  - Readable survey: [Borja Balle, Xavier Carreras, Franco M. Luque, Ariadna Quattoni. Spectral learning of weighted automata - A forward-backward perspective. 2014]
  - To go beyond: [Hadrien Glaude. Méthodes des moments pour l'inférence de systèmes séquentiels linéaires rationnels, PhD thesis, 2016]

Context

- 10+ years of research (lot of researchers - not me)
- 1+ year of programming developments founded by the Laboratoire d'Excellence Archimède (ANR-11-LABX-0033):
  - 2 (part time) research engineers: Denis Arrivault & Dominique Benielli (Archimède Development team)
  - 2 (very part time) researchers: François Denis & myself
  - A first release as a baseline for the SPiCe competition `http://spice.lif.univ-mrs.fr/index.php` (April 2016)
  - Final release as a Scikit-Learn compatible toolbox (version 1.0: October 2016; version 1.2: May 2018)
  - [Denis Arrivault, Dominique Benielli, François Denis, Rémi Eyraud. Scikit-SpLearn: a toolbox for the spectral learning of weighted automata compatible with scikit-learn. 2017]

Outline

Spectral Learning of Weighted Automata (WA)

Scikit SpLearn toolbox

Conclusion and Future developments

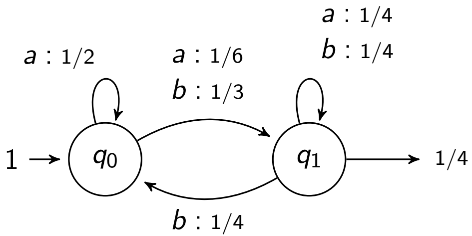
Outline

Spectral Learning of Weighted Automata (WA)

Scikit SpLearn toolbox

Conclusion and Future developments

Linear representation of Weigthed Automata



$$\alpha_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \alpha_\infty = \begin{bmatrix} 0 \\ 1/4 \end{bmatrix}$$

$$M_a = \begin{bmatrix} 1/2 & 1/6 \\ 0 & 1/4 \end{bmatrix} \qquad M_b = \begin{bmatrix} 0 & 1/3 \\ 1/4 & 1/4 \end{bmatrix}$$

WA and linear projection

To compute the weight given to $w = \sigma_1 \dots \sigma_m$:

$$\alpha_0^\top M_w \alpha_\infty = \alpha_0^\top M_{\sigma_1} \dots M_{\sigma_m} \alpha_\infty$$

Example in previous WA: $r(bba) = \alpha_0^\top M_b M_b M_a \alpha_\infty = 5/576$

Let $\alpha^i(w)$ such that $\alpha^0(w) = \alpha_0^\top$ and $\alpha^{i+1}(w) = \alpha^i(w) M_{\sigma_i}$.
The jth component of vector α^i is the sum of the weights of all paths that arrive to the state j given the corresponding prefix.

$\alpha^i(w)$ can be seen as a linear projection into $\mathbb{R}^{nb\_states}$. The automaton is then computing the inner product $\langle \alpha^{|w|}, \alpha_0 \rangle$.

Hankel matrix

$$\mathcal{H}_r = \begin{bmatrix} r(\epsilon \cdot \epsilon) & r(\epsilon \cdot a) & r(\epsilon \cdot b) & r(\epsilon \cdot aa) & r(\epsilon \cdot ab) & \dots \\ r(a \cdot \epsilon) & r(a \cdot a) & r(a \cdot b) & r(a \cdot aa) & r(a \cdot ab) & \dots \\ r(b \cdot \epsilon) & r(b \cdot a) & r(b \cdot b) & r(b \cdot aa) & r(b \cdot ab) & \dots \\ r(aa \cdot \epsilon) & r(aa \cdot a) & r(aa \cdot b) & r(aa \cdot aa) & r(aa \cdot ab) & \dots \\ r(ab \cdot \epsilon) & r(ab \cdot a) & r(ab \cdot b) & r(ab \cdot aa) & r(ab \cdot ab) & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ & & & & & \end{bmatrix}$$

Theorem [Carlyle & Paz,1971; Flies, 1974]:
A rational series $r : \Sigma^* \to \mathbb{R}$ can be defined by a WA iff the rank of its Hankel matrix is finite. In that case this rank is the minimal number of states of any WA that computes r.

Hankel Basis

- Only finite sub-blocks of a Hankel matrix are of interest
- Defined over a basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$
  - $\mathcal{P}$ is a set of rows (prefixes)
  - $\mathcal{S}$ is a set of columns (suffixes)
- $H_{\mathcal{B}}$ is the Hankel matrix restricted to $\mathcal{B}$
- 2 important properties:
  - prefix-close
  - complete

From a Hankel matrix to a WA

[Bailly et al., 2009; Hsu et al.,2009; Balle et al.,2014]:

- Given H a Hankel matrix of a series r and $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ a *complete prefix-close* basis
- For $\sigma \in \Sigma$, let H_σ the sub-block on the basis $(\mathcal{P}\sigma, \mathcal{S})$
- $H_{\mathcal{B}} = PS$ a rank factorization, *i.e.* $P \in \mathcal{R}^{p \times rank(r)}$ and $S \in \mathcal{R}^{rank(r) \times s}$
- Then $\langle \alpha_0, (M_\sigma)_{\sigma \in \Sigma}, \alpha_\infty \rangle$ is a minimal WA for r with
  - $\alpha_0^\top = h_{\epsilon,\mathcal{S}}^\top S^+$
  - $\alpha_\infty = P^+ h_{\mathcal{P},\epsilon}$
  - $M_\sigma = P^+ H_\sigma S^+$

  where $h_{\mathcal{P},\epsilon} \in \mathbb{R}^{\mathcal{P}}$ denotes the p-dimensional vector with coordinates $h_{\mathcal{P},\epsilon}(u) = r(u)$, and $h_{\epsilon,\mathcal{S}}$ the s-dimensional vector with coordinates $h_{\epsilon,\mathcal{S}}(v) = r(v)$

Hankel matrix variants

- The *prefix Hankel matrix*: $H^p(u, v) = r(uv\Sigma^*) = r_p(uv)$ for any $u, v \in \Sigma^*$. Rows are indexed by prefixes and columns by factors (substrings).
- The *suffix Hankel matrix*: $H^s(u, v) = r(\Sigma^* uv) = r_s(uv)$ for any $u, v \in \Sigma^*$. Rows are indexed by factors and columns by suffixes.
- The *factor Hankel matrix*: $H^f(u, v) = r(\Sigma^* uv\Sigma^*) = r_f(uv)$ for any $u, v \in \Sigma^*$. In this matrix both rows and columns are indexed by factors.

Theorem [Balle et al, 2014; Gybels et al., 2014]:
The ranks of r_p, r_s, and r_f are all equal to the rank of r.

Spectral learning of WA

- Fix a Hankel variant, a basis, and a rank value
- Estimate the corresponding Hankel sub-block(s) using the training data (positive examples only)
- Compute the truncated singular value decomposition (SVD) (gives you a rank factorization)
- Generate the corresponding WA

Some theoretical results

[Hsu et al., 2009] With high probability:

$$||H_{\mathcal{B}} - \hat{H}_{\mathcal{B}}||_F \leq \mathcal{O}(\frac{1}{\sqrt{m}})$$

where m is the number of examples and $\hat{H}_{\mathcal{B}}$ the *empirical Hankel sub-block*.

[Bailly et al., 2009] $\hat{H}_{\mathcal{B}}$ is of full rank with probability one.

[Balle & Mohri, 2018] The Rademacher complexity of the class of WA with n states is bounded.

Extension

- Spectral Learning of Weighted Tree Automata: [Bailly et al., 2010; Rabusseau et al., 2015]
- Spectral Learning of Graph Weighted Models: [Rabusseau, 2018]
- Multitask Spectral Learning of Weighted Automata [Rabusseau et al., 2017]
- A priori basis selection [Quattoni et al., 2017]
- Nonlinear Weighted Finite Automata [Li et al., 2017]

Outline

Spectral Learning of Weighted Automata (WA)

Scikit SpLearn toolbox

Conclusion and Future developments

Toolbox environment

- Scikit-Learn: a toolbox with main machine learning algorithms, widely used.

- Written in Python 3.5 (compatible 2.7)
- Easy installation:

  `pip install scikit-splearn`
- Sources easily downloadable (Free BSD license): `https://pypi.python.org/pypi/scikit-splearn`
- Detailed documentation and more: `http://pageperso.lis-lab.fr/~remi.eyraud/scikit-splearn/`

Content

4 classes:

- Automaton: a linear representation of WA, including useful methods (e.g. numerically stable PA minimization)
- Datasets.base: to load samples
- Hankel: for Hankel matrices, with a bunch of tools
- Spectral: main class, with functions `fit`, `predict`, `score` and many other

Load data

Function `load_data_sample` loads from a file with usual GI format and returns a sample in Scikit-Learn format.

```
>>> from splearn.datasets.base import load_data_sample
>>> train = load_data_sample("3.pautomac.train")
>>> train.nbEx
20000
>>> train.nbL
4
```

Splearn-array

Inherit from python `numpy` `ndarray` object

```
>>> train.data
SplearnArray([[ 3.,  0.,  3., ..., -1., -1., -1.],
              [ 3.,  3., -1., ..., -1., -1., -1.],
              [ 3.,  2.,  0., ..., -1., -1., -1.],
              ...,
              [ 3.,  1.,  3., ..., -1., -1., -1.],
              [ 3.,  3., -1., ..., -1., -1., -1.],
              [ 3.,  1.,  3., ..., -1., -1., -1.]])
```

Contains also the dictionaries `train.data.sample`, `train.data.pref`, `train.data.suff`, and `train.data.fact` (empty at that moment).

Estimator: Spectral

- Inherit from BaseEstimator (sklearn.base)
- parameters:
  - `rank`: the value for the rank factorization
  - `version`: the variant of Hankel matrix to use
  - `sparse`: if `True`, uses a sparse representation for the Hankel matrix
  - `partial`: if `True`, computes only a specified sub-block of the Hankel matrix
  - `lrows` and `lcolumns`: if `partial` is `True`, either integers corresponding to the max length of elements to consider, or list of strings to use for the Hankel matrix
  - `smooth_method`: `'none'` or `'trigram'` (so far)
  - `full_svd_calculation`: random or full SVD computation
  - `mode_quiet`

Estimator: Spectral

Usage:

```
>>> from splearn.spectral import Spectral
>>> est = Spectral()
>>> est.get_params()
{'rank': 5, 'version': 'classic', 'lrows': 7,
    'lcolumns': 7, 'partial': True, 'sparse': True,
    'full_svd_calculation': False,
    'smooth_method': 'none', 'mode_quiet': False}
>>> est.set_params(lrows=5, lcolumns=5,
                   smooth_method='trigram',
                   version='factor')
Spectral(full_svd_calculation=False, lcolumns=5,
    lrows=5, mode_quiet=False, partial=True, rank=5,
    smooth_method='trigram', sparse=True,
    version='factor')
```

Estimator: Spectral

Main methods:

- **fit**(self, X, y=None)
- **predict**(self, X)
- **predict_proba**(self,X)
- **loss**(self, X, y=None)
- **score**(self, X, y=None, scoring="perplexity")
- **nb_trigram**(self)

SpLearn use case

```
>>> est.fit(train.data)
Start Hankel matrix computation
End of Hankel matrix computation
Start Building Automaton from Hankel matrix
End of Automaton computation
Spectral(full_svd_calculation=False, lcolumns=5, lrows=5,
     mode_quiet=False,partial=True, rank=5,
     smooth_method='trigram', sparse=True, version='factor
>>> test = load_data_sample("3.pautomac.test")
>>> est.predict(test.data)
array([3.23849562e-02, 1.24285813e-04, ...
...])
>>> est.nb_trigram()
80
```

SpLearn use case (cont'd)

```
>>> #Create y vector for supervised evaluation
>>> targets = open("3.pautomac_solution.txt", "r")
>>> targets.readline()   #get rid of nb lines
>>> target_proba = [float(line[:-1]) for line in targets]
>>>
>>> # Compute the means of squared differences
>>> est.loss(test.data, y=target_proba)
2.162725190444073e-05
>>> # Compute the perplexity
>>> est.score(test.data, y=target_proba)
71.49521987246547
```

SpLearn and Scikit methods

- Cross-validation

```
>>> from sklearn.model_selection import cross_val_score
>>> est.set_params(mode_quiet=True)
>>> scores = cross_val_score(est, train.data, cv=5)
>>> scores
array([-10.11871728, -10.44673223, -10.36855581,
   -10.39396116, -10.34336961])
>>> scores = cross_val_score(est, test.data,
                             target_proba, cv=5)
>>> scores
array([31.52112125, 80.45998967, 87.53014326,
       73.43037055, 73.30544451])
```

SpLearn and Scikit methods

- Gridsearch

```
>>> from sklearn.model_selection import GridSearchCV
>>> param = {'version': ['suffix','prefix'],
             'lcolumns': [5, 6, 7], 'lrows': [5, 6, 7]}
>>> grid = GridSearchCV(est, param)
>>> grid.fit(train.data)
GridSearchCV(cv=None, error_score='raise',
       estimator=Spectral(...),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'version': ['suffix', 'prefix'],
          'lcolumns': [5, 6, 7], 'lrows': [5, 6, 7]},
       pre_dispatch='2*n_jobs', refit=True,
       return_train_score='warn', scoring=None,
       verbose=0)
>>> grid.best_params_
{'lcolumns': 5, 'lrows': 7, 'version': 'prefix'}
```

- And all other (not contractual...) Scikit-learn methods

More than a learning toolbox

- Lots of tools to play with weighted automata:
  - A numerically stable and parametrized minimization algorithm
  - Possibilities of saving or downloading an automaton
  - Visualization methods
  - Prefix/Suffix/Factor/Next symbol transformation
  - Test for absolute convergence
  - ...
- Data treatments
- Results analysis

New in version 1.2: modularity for learning

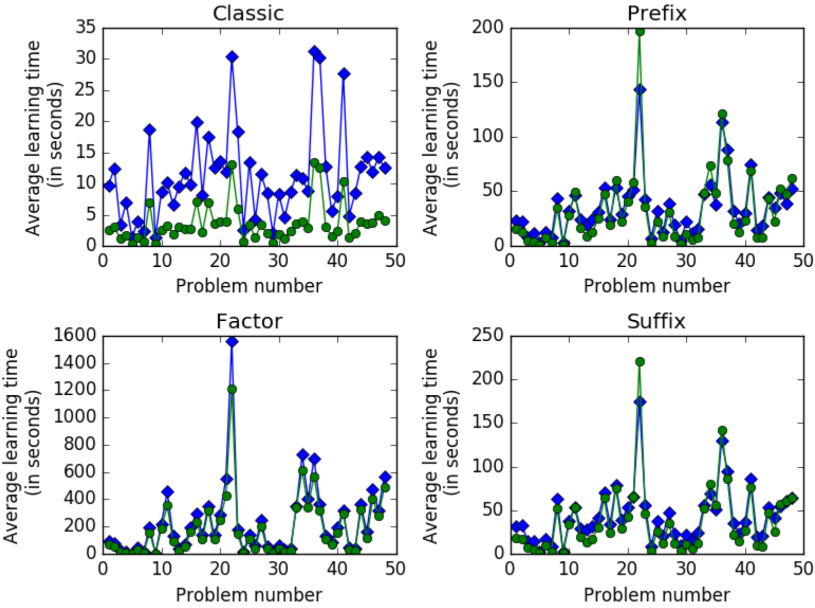
- Decomposition of the spectral learning algorithm:
  - `polulate_dictionnaries(Spectral.self, X)`: creates the needed dictionaries (prefixes/suffixes/factors of needed sizes)
  - `Hankel(sample_instance,` *`+ parameters of fit`*`)`: creates the needed blocks of Hankel
  - `to_automaton(Hankel.self, rank, mode_quiet)`: creates the WA from the Hankel blocks.
- Different uses: for instance, to evaluate a black-box

Outline

Conclusion

- Tested (unitary, 95% coverage)
- Used on all 48 PAutomaC data (results in the article)
    - `rank` between 2 and 40
    - `lrows` and `lcolumns` between 2 and 6
    - for all 4 Hankel matrix variants
    - a total of 28 000+ runs

Time comparison between sp2learn and splearn



Future developments

- Data generation tools
- Basis selection function(s)
- Other scoring functions (WER, KL, NDCG ...)
- Real smoothing methods (Baum-Welch?)
- Other Method of Moments algorithms
- Moving to tree automata

Any comment (and help) welcomed!

Some advertisement to finish: 2 upcoming events

- LearnAut 2018:
  - mid-FLoC workshop
  - July 13th, Oxford, UK
  - Early registration deadline: June 6th
  - Nice program, including 4 invited talks:
    - Doina Precup (McGill University & DeepMind, Canada)
    - Alexander Clark (King's College London, UK)
    - Kousha Etessami (University of Edinburg, UK)
    - George Argyros (Columbia University, USA)
  - `https://learnaut2018.wordpress.com/`
- ICGI 2018
  - 14th International Conference in Grammatical Inference
  - September 5-7 2018, Wrocław, Poland
  - submission deadline: June 15
  - preliminary works also welcomed
  - `http://icgi2018.pwr.edu.pl/`