Scikit Spectral Learning (SpLearn): a toolbox for the spectral learning of weighted automata

Denis Arrivault [1] Dominique Benielli [1] François Denis [2]
Rémi Eyraud [2]

[1]LabEx Archimède, Aix-Marseille University, France

[2]QARMA team, Laboratoire d'Informatique Fondamentale de Marseille, France

ICGI 2016 (Delft)

Context

- A one year project founded by the Laboratoire d'Excellence Archiméde (ANR-11-LABX-0033)
- 2 (part time) research engineers
- 2 (very part time) researchers
- A first release as a baseline for the SPiCe competition (April 1st 2016)
- Final release as a ScikitLearn-like toolbox (October 5th 2016)

Outline

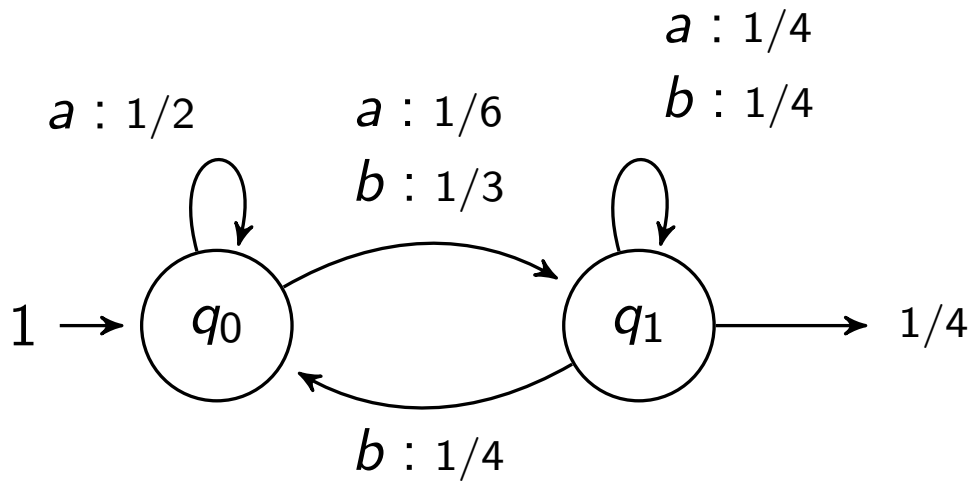
Spectral Learning of Weighted Automata (WA)

Scikit SpLearn toolbox

Conclusion and Future developments

Outline

Linear representation of Weigthed Automata



$$I = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad T = \begin{bmatrix} 0 \\ 1/4 \end{bmatrix}$$

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

$$r(bba) = I^\top M_b M_b M_a T = 5/576$$

Hankel matrix

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

- Only finite sub-blocks 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 $\mathcal{B}$

Hankel matrix variants

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

From a Hankel matrix to a WA

[Balle et al., 2014]:

- Given H a Hankel matrix of a series r and $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ a *complete* 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
- Then $\langle I, (M_\sigma)_{\sigma\in\Sigma}, T\rangle$ is a minimal WA for r with
  - $I^\top = h_{\epsilon,\mathcal{S}}^\top S^+$
  - $T = P^+ h_{\mathcal{P},\epsilon}$
  - $M_\sigma = P^+ H_\sigma S^+$

  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 $h_{\epsilon,\mathcal{S}}(v) = r(v)$

Spectral learning of WA

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

Outline

Toolbox environment

- 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:
  `https://pythonhosted.org/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 and returns a sample in Scikit-Learn format.

```
>>> from splearn.datasets.base import load_data_sample
>>> train = load_data_sample("1.pautomac.train")
>>> train.nbEx
20000
>>> train.nbL
4
```

Splearn-array

Inherit from python `numpy ndarray` object

```
>>> train.data
Splearn_array([[ 5.,  4.,  1., ..., -1., -1., -1.],
       [ 4.,  4.,  7., ..., -1., -1., -1.],
       [ 2.,  4.,  4., ..., -1., -1., -1.],
       ...,
       [ 4.,  1.,  3., ..., -1., -1., -1.],
       [ 0.,  6.,  5., ..., -1., -1., -1.],
       [ 4.,  0., -1., ..., -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)

Estimator: Spectral

Usage:

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

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(lcolumns=5, lrows=5, 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.loss(test.data), est.score(test.data)
(23.234189560218198, -23.234189560218198)
>>> est.nb_trigram()
61
```

SpLearn use case (cont'd)

```
>>> targets = open("1.pautomac_solution.txt", "r")
>>> targets.readline()
'1000\n'
>>> target_proba = [float(line[:-1]) for line in targets]
>>> est.loss(test.data, y=target_proba)
2.6569772687614514e-05
>>> est.score(test.data, y=target_proba)
46.56212657907001
```

SpLearn and Scikit methods

- Cross-validation

```
>>> from sklearn import cross_validation as c_v
>>> c_v.cross_val_score(est, train.data, cv = 5)
array([-17.74749858, -17.63678657, -17.60412108,
       -17.43726243, -17.73316833])
>>> c_v.cross_val_score(est, test.data, target_proba,
                        cv = 5)
array([  16.48311708,   56.46485233,  111.20384957,
        89.13625474,   28.84640423])
```

SpLearn and Scikit methods

- Gridsearch

```
>>> from sklearn import grid_search as g_s
>>> param = {'version': ['suffix','prefix'],
             'lcolumns': [5, 6, 7], 'lrows': [5, 6, 7]}
>>> grid = g_s.GridSearchCV(est, param, cv = 5)
>>> grid.fit(train.data)
>>> grid.best_params_
{'version': 'prefix', 'lcolumns': 5, 'lrows': 6}
>>> grid.best_score_
-17.636386233284796
```

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

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

Future developments

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

Any comment (and help) welcomed!

Time comparison between sp2learn and splearn