Scikit-SpLearn: a toolbox for the spectral learning of weighted automata compatible with scikit-learn

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

[1]LabEx Archimède, Aix-Marseille Université, France
[2]QARMA team, Laboratoire d'Informatique Fondamentale de Marseille, Aix-Marseille Université, France

Introduction

Scikit-SpLearn is a Python toolbox for the spectral learning of weighted automata from a set of strings, licensed under Free BSD, and compatible with the well-known scikit-learn toolbox.

Weighted automata

- Automata Representation : $\langle \Sigma, Q, \iota, \tau, \phi \rangle$ where
  - Σ is a finite set of symbols; Σ^* is the set of all strings over Σ.
  - Q is a finite set of states
  - $\iota : Q \mapsto \mathbb{R}$ is the initial function
  - $\tau : Q \mapsto \mathbb{R}$ is the final function
  - $\phi : Q \times \Sigma \times Q \mapsto \mathbb{R}$ is the transition function

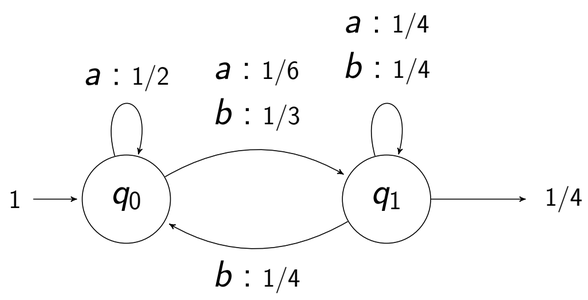

Figure 1: A weighted automaton.

- Linear Representation : $\langle \Sigma, n, I, (M_\sigma)_{\sigma\in\Sigma}, T \rangle$ where
  - n is the dimension of the representation
  - $I \in \mathbb{R}^n$ is the initial vector
  - $T \in \mathbb{R}^n$ is the final vector
  - $M_\sigma \in \mathbb{R}^{n\times n}$ is the transition matrix associated with σ.

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

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

Figure 2: The equivalent linear representation.

- A WA A computes a function $r_A : \Sigma^* \to \mathbb{R}$ as follows:
  - The weight of a path $(q_0, \sigma_1, q_1)\dots(q_{n-1}, \sigma_n, q_n)$ is equal to the product $\iota(q_0)\phi(q_0, \sigma_1, q_1)\dots\phi(q_{n-1}, \sigma_n, q_n)\tau(q_n)$.
  - The weight $r_A(w)$ of a string w is the sum of the weights of the paths that read w.
  - It can be computed by $r_A(w) = I^\top M_{\sigma_1}\dots M_{\sigma_n} T$.
  - Example:

$$\begin{aligned} r_A(ab) &= 1\times 1/2\times 1/3\times 1/4 + 1\times 1/6\times 1/4\times 1/4 \\ &= I^\top M_a M_b T = 5/96. \end{aligned}$$

- Hankel matrices
  - Let $\mathcal{P}, \mathcal{S} \subseteq \Sigma^*$ and $r : \Sigma^* \to \mathbb{R}$. For $x \in \Sigma \cup \{\epsilon\}$, the associated Hankel matrices $H_x \in \mathbb{R}^{|\mathcal{P}|\times|\mathcal{S}|}$ are defined by $H_x(u, v) = r(uxv)$ for any $(u, v) \in \mathcal{P}\times\mathcal{S}$.
  - **Fundamental property:** If r is computed by a WA and if $\mathcal{P}, \mathcal{S}$ are "large enough", then the rank of H_ϵ is equal to the dimension of r and a representation of r can be computed from the matrices H_x.
  - Example: let $\mathcal{P} = \mathcal{S} = \{\epsilon, a\}$. It is sufficient to know H_ϵ, H_a, H_b, i.e. $r(\epsilon), r(a), r(b), r(aa), r(ab), r(ba), r(aaa), r(aba)$ to recover A.
  - **Spectral formulas:** let $H_\epsilon = P \cdot S$ be a rank n factorization of H_ϵ, and let

$$I^\top = H_\epsilon[0, :]^\top S^+, \qquad T = P^+ H_\epsilon[:, 0], \qquad M_\sigma = P^+ H_\sigma S^+$$

  Then $\langle \Sigma, n, I, (M_\sigma)_{\sigma\in\Sigma}, T \rangle$ is a linear representation of r [3].

Spectral learning

- Spectral learning algorithm
  - From a sample of strings $\mathcal{S}$, compute estimates $\widehat{H}_x$ of H_x by replacing $r(w)$ by $\widehat{r}(w) = \frac{|\mathcal{S}_w|}{|\mathcal{S}|}$.
  - Estimate the rank n
  - Compute an n SVD of $\widehat{H}_\epsilon$ to get a factorization: $\widehat{H}_\epsilon = \widehat{P} \cdot \widehat{S}$.
  - Plug these values in the spectral formulas.
- Improvements: learn first a variant of r
  - the prefix variant: $r_p(w) = \sum_{s\in\Sigma^*} r(ws)$ from $\hat{H}^p_x(u, v) = \hat{r}_p(uxv)$,
  - the suffix variant: $r_s(w) = \sum_{p\in\Sigma^*} r(pw)$ from $\hat{H}^s_x(u, v) = \hat{r}_s(uxv)$,
  - the factor variant: $r_f(w) = \sum_{s,p\in\Sigma^*} r(pws)$ from $\hat{H}^f_x(u, v) = \hat{r}_f(uxv)$.

  All these variants can be computed by a WA and have the same rank as r.

  Then compute $\widehat{r}$ from $\widehat{r}_*$.

Toolbox SpLearn

The Scikit-SpLearn toolbox is made of 5 Python classes and implements several variants of the **spectral learning algorithm**.

- easy installation, the pip command: *pip install scikit-splearn*
- extremely tunable: wide range of parameters allowed

An earlier version [4] (scikit-learn not compatible) released for the SPiCe competition [5].

Spectral Class: Estimator

The class **Spectral** inherits from `BaseEstimator` of `sklearn.base`. At the initialization of a class instance, specified parameters:

- `rank` is the estimated value for the rank factorization.
- `version` indicates which variant of the Hankel matrix is going to be used (possible values are `classic` for $\hat{H}_x$, `prefix` for $\hat{H}^p_x$, `suffix` for $\hat{H}^s_x$, and `factor` for $\hat{H}^f_x$).
- `partial` indicates whether all the elements have to be taken into account in the Hankel matrix.
- `lrows` and `lcolumns` can either be lists of strings that form the basis $\mathcal{B}$ of the sub-block that is going to be considered, or integers corresponding to the maximal length of elements of the basis. Need to be specified only when `partial` is activated.
- `smooth_method` can be `'none'` (by default) or `'trigram'` if one wants to replace potential non-positive weights given by the learned WA by corresponding 3-gram values.
- `mode_quiet` is a boolean indicating whether a run will indicate the different steps followed successively.

Spectral Learning Usage

Data Loading

```
>>> from splearn.datasets.base
    ...   import load_data_sample
>>> train =
 ...   load_data_sample("1.pautomac.train")
>>> train.nbEx
20000
>>> train.nbL
4
>>> train.data
Splearn_array([[ 5., 4., 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.]])
>>> train.data.sample, train.data.pref,
...      train.data.suff, train.data.fact
({}, {}, {}, {})  # dict.filled after fit
```

Spectral instance

```
>>> from splearn.spectral import Spectral
>>> est = Spectral()
>>> est.set_params(lrows=5, lcolumns=5,
 smooth_method='trigram', version='factor',
 mode_quiet=True)
Spectral(lcolumns=5, lrows=5, partial=True,
       rank=5, smooth_method='trigram',
       sparse=True, version='factor',
       mode_quiet=True)
```

Fit and Predict methods

```
>>> est.fit(train.data)
Spectral(lcolumns=5, lrows=5,
    partial=True, rank=5,
    smooth_method='trigram', sparse=True,
    version='factor', mode_quiet=True)
>>> test =
...   load_data_sample("1.pautomac.test")
>>> est.predict(test.data)
array([3.2384956e-02, 1.2428581e-04,...  ])
```

Loss and Score method

```
>>> est.loss(test.data)
23.234189560218198
>>>  est.score(test.data)
-23.234189560218198
>>> est.nb_trigram()
61
>>> targets =
 ...   open("1.pautomac_solution.txt", "r")
>>> target_proba =
 ...   [float(r[:-1]) for r in targets]
>>> est.loss(test.data, y=target_proba)
2.6569772687614514e-05
>>> est.score(test.data, y=target_proba)
46.56212657907001
```

Compatible with scikit-learn

```
>>> 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])
>>> 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
```

Download

Documentations and Technical manual:

- http://pageperso.lif.univ-mrs.fr/~remi.eyraud/scikit-splearn/

Python site, PyPi:

- https://pypi.python.org/pypi/scikit-splearn

Bibliography

[1] Hsu D., Kakade S., Zhang T. *Conference on Computational Learning Theory*. A Spectral Algorithm for Learning Hidden Markov Models. 2009.

[2] Bailly R., Denis F., Ralaivola L. *International Conference on Machine Learning*. Grammatical inference as a principal component analysis problem. 2009.

[3] Balle B., Carreras X., Luque F., Quattoni A. *Machine Learning* Vol. 96, Spectral learning of weighted automata. 2014.

[4] D. Arrivault, D. Benielli, F. Denis, R. Eyraud *Proceedings of the International Conference on Grammatical Inference.*. Sp2Learn: A Toolbox for the Spectral Learning of Weighted Automata. 2016.

[5] Balle B., Eyraud R., Luque F. M., Quattoni A., Verwer S. *Proceedings of the International Conference on Grammatical Inference*. Results of the Sequence PredIction ChallengE (SPiCe): a Competition on Learning the Next Symbol in a Sequence. 2016.