Algorithms: AutoCor, Lisa, DiscSegm and CutSegm

Guillaume Guex

August 5, 2020

General notations

- $\mathbf{1}_n$: the (column) vector of ones with length n.
- $\mathbf{I}_n$: the identity matrix of size $(n \times n)$.
- $\circ, \div$: the component-wise multiplication or division.
- $\mathbf{v}^{(a)}, \mathbf{M}^{(a)}$: the component-wise power of a of a vector or a matrix.
- $\mathbf{exp}(\mathbf{v})$ or $\mathbf{Exp}(\mathbf{M})$: the component-wise exponential on a vector or on a matrix.
- $\mathbf{diag}(\mathbf{M})$: the diagonal function taking a matrix and returning the diagonal vector.
- $\mathbf{Diag}(\mathbf{v})$: the diagonal function taking a vector and returning a diagonal matrix.

Algorithms

Algorithm 1: AutoCor algorithm

Input :

- A scalar r defining the maximum range of autocorrelation.
- A $(v \times n)$ presence matrix $\mathbf{P}$, where $p_{ij} = 1$ iif type i is in token position j, 0 otherwise (constructed from corpus).
- A $(v \times v)$ dissimilarity matrix $\mathbf{D}_v$ between types.
- A vector $\boldsymbol{\pi}$ of length v containing type frequencies.

Output:

- The vector $\boldsymbol{\delta}$ of length r containing autocorrelation indices for a token position difference of r.

Compute the global inertia: $\Delta \leftarrow \frac{1}{2}\mathbf{1}_v^\top(\boldsymbol{\pi}\boldsymbol{\pi}^\top \circ \mathbf{D}_v)\mathbf{1}_v$;
for i *from 1 to* r **do**
    Compute the $(n \times n)$ exchange matrix $\mathbf{E}$ with range i;
    Compute the $(v \times v)$ between-types exchange matrix: $\boldsymbol{\mathcal{E}} \leftarrow \mathbf{P}\mathbf{E}\mathbf{P}^\top$;
    Compute the local variance: $\Delta_{\text{loc}} \leftarrow \frac{1}{2}\mathbf{1}_v^\top(\boldsymbol{\mathcal{E}} \circ \mathbf{D}_v)\mathbf{1}_v$;
    Compute the autocorrelation for range i: $\delta_i \leftarrow \frac{\Delta - \Delta_{\text{loc}}}{\Delta}$;
end
return $\boldsymbol{\delta}$;

Algorithm 2: Lisa algorithm

Input :

- A $(n \times n)$ exchange matrix $\mathbf{E}$ between token positions.
- A $(v \times v)$ dissimilarity matrix $\mathbf{D}_v$ between types.
- A $(v \times n)$ presence matrix $\mathbf{P}$, where $p_{ij} = 1$ iif type i is in token position j, 0 otherwise (constructed from corpus).
- A vector $\boldsymbol{\pi}$ of length v containing type frequencies.

Output:

- The vector $\boldsymbol{\delta}$ of length n containing lisa indices for each token.

Compute the token weights vector of length n: $\mathbf{f} \leftarrow \mathbf{E}\mathbf{1}_n$;
Compute the $(n \times n)$ transition matrix between token: $\mathbf{W} \leftarrow \mathbf{E} \div \mathbf{f}\mathbf{1}_n^\top$;
Compute the $(n \times n)$ token dissimilarity matrix: $\mathbf{D}_n \leftarrow \mathbf{P}^\top \mathbf{D}_v \mathbf{P}$;
Compute the $(n \times n)$ centration matrix: $\mathbf{H} \leftarrow \mathbf{I}_n - \mathbf{1}\mathbf{f}^\top$;
Compute the $(n \times n)$ scalar products matrix: $\mathbf{B} \leftarrow -\frac{1}{2}\mathbf{H}\mathbf{D}_n\mathbf{H}^\top$;
Compute the global inertia: $\Delta \leftarrow \frac{1}{2}\mathbf{1}_v^\top(\boldsymbol{\pi}\boldsymbol{\pi}^\top \circ \mathbf{D}_v)\mathbf{1}_v$;
Compute the lisa vector of length n: $\boldsymbol{\delta} \leftarrow \frac{1}{\Delta}\mathbf{diag}(\mathbf{W}\mathbf{B})$;
return $\boldsymbol{\delta}$;

Algorithm 3: DiscSegm algorithm

Input :

- A $(n \times n)$ exchange matrix $\mathbf{E}$ between token positions.
- A $(v \times v)$ dissimilarity matrix $\mathbf{D}_v$ between types.
- A $(v \times n)$ presence matrix $\mathbf{P}$, where $p_{ij} = 1$ iif type i is in token position j, 0 otherwise (constructed from corpus).
- A vector $\boldsymbol{\pi}$ of length v containing type frequencies.
- An integer m defining the number of groups.
- Hyperparmaters $\alpha > 0$, $\beta > 0$ and $\kappa \in [0, 1]$.

Output:

- The $(n \times m)$ membership matrix $\mathbf{Z}$ with fuzzy memberships of tokens into each group.

Compute the token weights vector of length n: $\mathbf{f} \leftarrow \mathbf{E}\mathbf{1}_n$;
Compute the $(n \times n)$ transition matrix between token: $\mathbf{W} \leftarrow \mathbf{E} \div \mathbf{f}\mathbf{1}_n^\top$;
Compute the $(n \times n)$ token dissimilarity matrix: $\mathbf{D}_n \leftarrow \mathbf{P}^\top \mathbf{D}_v \mathbf{P}$;
Initialize $\widetilde{\mathbf{Z}}$ with random strictly positive values;
Normalize $\mathbf{Z}$: $\mathbf{Z} \leftarrow \widetilde{\mathbf{Z}} \div (\widetilde{\mathbf{Z}}\mathbf{1}_m\mathbf{1}_n^\top)$;
while $\mathbf{Z}$ *has not converged* **do**
    Compute the group weights vector of length m: $\boldsymbol{\rho} \leftarrow (\mathbf{Z} \circ \mathbf{f}\mathbf{1}_m^\top)^\top \mathbf{1}_n$;
    Compute the $(n \times m)$ token distribution among groups matrix: $\mathbf{F} \leftarrow (\mathbf{Z} \div \mathbf{1}_n\boldsymbol{\rho}^\top) \circ \mathbf{f}\mathbf{1}_m^\top$;
    Compute the $(n \times m)$ uncentered token-centroid distance matrix: $\widetilde{\boldsymbol{\mathcal{D}}} \leftarrow \mathbf{D}_n^\top \mathbf{F}$;
    Compute the vector of group inertia of length m: $\boldsymbol{\delta} \leftarrow \frac{1}{2}\mathbf{diag}(\widetilde{\boldsymbol{\mathcal{D}}}^\top \mathbf{F})$;
    Compute the $(n \times m)$ token-centroid distance matrix: $\boldsymbol{\mathcal{D}} \leftarrow \widetilde{\boldsymbol{\mathcal{D}}} - \mathbf{1}_n\boldsymbol{\delta}^\top$;
    Compute the Dirichlet form vector or length m: $\boldsymbol{\epsilon} \leftarrow (\mathbf{E}\mathbf{Z}^{(2)})^\top \mathbf{1}_n - \mathbf{diag}(\mathbf{Z}^\top \mathbf{E}\mathbf{Z})$;
    Compute the $(n \times m)$ matrix $\mathbf{H}$: $\mathbf{H} \leftarrow \beta\boldsymbol{\mathcal{D}} + \alpha\mathbf{1}_n(\boldsymbol{\rho}^{(-\kappa)})^\top \circ (\mathbf{Z} - \mathbf{W}\mathbf{Z}) - \frac{\alpha\kappa}{2}\mathbf{1}_n(\boldsymbol{\rho}^{(-\kappa-1)} \circ \boldsymbol{\epsilon})^\top$;
    Compute the new unormalized membership matrix: $\widetilde{\mathbf{Z}} \leftarrow \mathbf{1}_n\boldsymbol{\rho}^\top \circ \mathbf{Exp}(-\mathbf{H})$;
    Normalize the new membership matrix: $\mathbf{Z}$: $\mathbf{Z} \leftarrow \widetilde{\mathbf{Z}} \div (\widetilde{\mathbf{Z}}\mathbf{1}_m\mathbf{1}_n^\top)$;
end
return $\mathbf{Z}$;