Simplex and S-map Algorithms

Yair Daon

December 4, 2017

Courant Institute, New York University
New York, NY 10012, USA

Abstract

Pseudo-code for the simplex projection algorithm [1] and the S-map algorithm [2]. Algorithms are presented for the simple case of predicting one variable using its own time series.

1 Notation

- E denotes the embedding dimension.
- k denotes the number of nearest neighbors we use. For the simplex method, the default is $k = E + 1$ but for the S-map method it can be much larger.
- T_p denotes how many time-steps into the future we are trying to predict.
- $X \in \mathbb{R}$ denotes a (potentially long) time series.
- $y \in \mathbb{R}^E$ is a vector of lagged observations for which we want to make a prediction — in the simplest case where all components of the vector are single time step lags, y_1 represents the current value, y_2 is the value one time step prior and y_E is the value $E - 1$ time steps prior.
- $\theta \geq 0$ is the tuning parameter in the S-map method.
- $X_t^E = (X_t, X_{t-1}, \ldots, X_{t-E+1})' \in \mathbb{R}^E$ denotes the lagged embedding vectors.
- $\|v\|$ is an unspecified norm of v. We do not specify which norm to use and that choice is left to the user / reader.
- $\|v\|_2^2 = \sum_i v_i^2$ is the squared L2-norm (squared Euclidean distances).
- Entries of matrices and vectors are indexed in the standard linear algebraic fashion, starting at 1 (like the R standard) and not at 0 (like the C/C++ and python standard).

2 Helper Methods

2.1 Nearest neighbors

I will not write implementation of the nearst neighbors method, just present its description. The method will be used with the signature presented in algorithm 1.

The input variables X, y and k are defined in section 1. The method returns a list of indices $N = \{N_1, \dots, N_k\}$ such that

$$\|X^E_{N_i} - y\| \leq \|X^E_{N_j} - y\| \text{ if } 1 \leq i \leq j \leq k,$$

Algorithm 1 Find Nearest neighbors

```
procedure NearNeighbor(y, X, k)
```

2.2 Least Squares

A least squares method finds x that minimizes the error in the solution of an over-determined linear system (more equations than variables). Below, $A \in \mathbb{R}^{p\times q}, p > q$ and $b \in \mathbb{R}^p$ and the least squares problem is to find

$$\hat{x} := \arg\min_{x\in\mathbb{R}^q} \|Ax - b\|_2^2.$$

This problem can be solved using a Singular Value Decompostion (SVD), as outlined in algorithm 2.

Algorithm 2 Least Squares via SVD

```
procedure LeastSquares(A, b)                ▷ Assume A ∈ ℝ^{p×q}, p > q.
    U, S, V ← SVD(A)                        ▷ Thus, A = USV'
    S^{inv} ← zeros(q, p)                   ▷ The zero matrix in ℝ^{q×p}
    for i = 1, ..., q do
        if S_{ii} > 10^{-5} S_{11} then     ▷ Note that 10^{-5} is arbitrary
            S^{inv}_{ii} ← 1/S_{ii}
    x ← V S^{inv} U' b
    return x
```

3 Simplex Projection

Ignoring ties in distances, minimal distances, minimal weights and other potential hazards, the following algorithm performs Simplex projection to predict T_p time-steps ahead.

Algorithm 3 Simplex Projection [1]

procedure SimplexPrediction(y, X, E, k, T_p)
  $N \leftarrow$ NearNeighbor(y, X, k) ▷ Find k nearest neighbors.
  $d \leftarrow \|X^E_{N_1} - y\|$ ▷ Define the distance scale.
  for $i = 1, \ldots, k$ **do**
    $w_i \leftarrow \exp(-\|X^E_{N_i} - y\|/d)$ ▷ Compute weights.
  $\hat{y} \leftarrow \sum_{i=1}^{k} \left(w_i X_{N_i+T_p}\right) / \sum_{i=1}^{k} w_i$ ▷ prediction = average of predicitions.
  return $\hat{y}$

4 S-map

Ignoring ties in distances, minimal distances, minimal weights and other potential hazards, the following algorithm uses the S-map method to predict T_p time-steps ahead.

Algorithm 4 S-map [2]

procedure SmapPrediction(y, X, E, k, T_p, θ)
  $N \leftarrow$ NearNeighbor(y, X, k) ▷ Find NN to use for prediciton.
  $d \leftarrow \frac{1}{k} \sum_{i=1}^{k} \|X^E_{N_i} - y\|$ ▷ Sum of distances.
  for $i = 1, \ldots, k$ **do**
    $w_i \leftarrow \exp(-\theta\|X^E_{N_i} - y\|/d)$ ▷ Compute weights.
  $W \leftarrow \text{diag}(w_i)$ ▷ Reweighting matrix.
  $A \leftarrow \begin{bmatrix} 1 & X_{N_1} & X_{N_1-1} & \ldots & X_{N_1-E+1} \\ 1 & X_{N_2} & X_{N_2-1} & \ldots & X_{N_2-E+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & X_{N_k} & X_{N_k-1} & \ldots & X_{N_k-E+1} \end{bmatrix}$ ▷ Design matrix.
  $A \leftarrow WA$ ▷ Weighted design matrix.
  $b \leftarrow \begin{bmatrix} X_{N_1+T_p} \\ X_{N_2+T_p} \\ \vdots \\ X_{N_k+T_p} \end{bmatrix}$ ▷ Response vector.
  $b \leftarrow Wb$ ▷ Weighted response vector.
  $\hat{c} \leftarrow \arg\min_c \|Ac - b\|_2^2$ ▷ Least squares, can be solved via algorithm 2.
  $\hat{y} \leftarrow \hat{c}_0 + \sum_{i=1}^{E} \hat{c}_i y_i$ ▷ Using the local linear model $\hat{c}$ for prediction.
  return $\hat{y}$

Note that k, the number of nearest neighbors used for prediciton, can be very large compared to the embedding dimension E. Since $A \in \mathbb{R}^{k\times(1+E)}$, this means that A is "tall and skinny" and the system $Ac = b$ is *over-determined* (it has more equations than variables). This means (typically) that there does not exist any unique c that solves said system. This is why we seek a least-squares solution instead.

References

[1] George Sugihara and Robert M. May. Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series. *Nature*, 344:734–741, 1990.

[2] G Sugihara. Nonlinear forecasting for the classification of natural time series. *Philosophical Transactions: Physical Sciences and Engineering*, 348(1688):477–495, 1994.