

DM2 - Time Series

Similarity

Una *time series* è una collezione di diverse osservazioni, prese in diversi momenti temporali. Si può generalizzare questa nozione anche a dati *sequenziali*: esempio più immagini che formano un video, o più parole che formano un testo. Si ha a che fare con una grande quantità di dati, talvolta ripetuti.

Con le *time series* possiamo estrarre trend, effettuare *clustering* e *motif discovery* (cioè trovare partner interni al dato stesso), trovare delle costanti (*rule discovering*), classificarle. Il minimo comune multiplo fra queste applicazioni è lo **studio della similarità**. Una similarità può essere a **livello formale** (una simile forma) o a **livello strutturale** (una simile informazione veicolata). **Ci concentreremo sulla somiglianza formale.**

Somiglianza strutturale (o *Model bases similarity*)

Per lunghe *time series*, una *shape-based similarity* ci farebbe cadere nella *course of dimensionality* ottenendo risultati poco interessanti. Ci muoviamo allora su un più profondo livello strutturale (*high level structure*). Se ad esempio vogliamo calcolare la distanza strutturale fra due *time series*, dobbiamo realizzare un vettore con i valori che ci interessano (ad esempio la media).

- The basic idea is to:
 1. extract *global* features from the time series,
 2. create a feature vector, and
 3. use it to measure similarity and/or classify
- Example of features:
 - mean, variance, skewness, kurtosis,
 - 1st derivative mean, 1st derivative variance, ...
 - parameters of regression, forecasting, **Markov model**

Talvolta, piuttosto che calcolare la *similarity* ci si concentra sulla *dissimilarity*. La CDM (Compression Based Dissimilarity) può essere così misura di similarity al posto della classica *Euclidean*, "comprimendo" informazioni.

$$d(x, y) = CDM(x, y) = \frac{c(x,y)}{c(x)+c(y)}$$

L'aspetto "temporale" viene meno con la somiglianza strutturale: il valore massimo può ad esempio essere all'inizio o alla fine, ma questa informazione non viene veicolata.

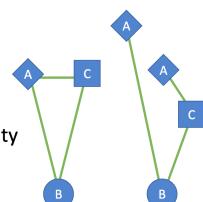
Somiglianza formale

Ricordiamo l'ABC della *distanza*. Se l'ultima proprietà non è rispettata non si tratta di *distance measure*, ma di *metric*. La distanza di Jaccard è una *metric*, si può realizzare l'esempio che invalida l'*Inequalità Triangolare*.

La distanza euclidea con le *time series* non cambia nel suo funzionamento. Abbiamo un vettore con i valori valori provenienti da due *time series* a parità di tempo. Si possono spesso incontrare delle *distorsioni*, dovute a: *Offset Translation*, *Amplitude Scaling*, *Linear Trend* o rumore. È ovviamente prima necessario normalizzare i dati.

Properties in a distance measure.

- $D(A,B) = D(B,A)$ Symmetry
- $D(A,A) = 0$ Constancy
- $D(A,B) = 0$ Iff $A = B$ Positivity
- $D(A,B) \leq D(A,C) + D(B,C)$ Triangular Inequality



$$D(Q, C) \equiv \sqrt{\sum_{i=1}^n (q_i - c_i)^2}$$

- $T1 = < 56, 176, 110, 95 >$
- $T2 = < 36, 126, 180, 80 >$

$$D(T1, T2) = \sqrt{(56-36)^2 + (176-126)^2 + (110-180)^2 + (95-80)^2}$$

ATTENZIONE: La normalizzazione considerata qui **considera i valori di tutta una time series**, ovvero le **righe** del dataset. La *standard scaling normalization* operava invece sulle colonne. Nel caso della *time series* si modificano i dati della *time series* stessa e dunque **non si può effettuare**

una **inverse transformation**, mentre con il classico **standard scaling** è possibile effettuare una **denormalization**.

A lato un esempio di **offset translation**: le due *time series* sono quasi identiche, ma i valori che usano sono "sfasati" (valore medio rispettivamente di 0.2 e 2); **seguono diversi range**. È prima necessario normalizzare, "spingendole" l'una vicina l'altra. In questo caso per normalizzare si sottrae il valore medio.

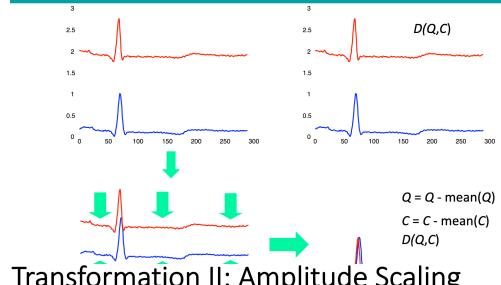
Nel caso di **amplitude scaling** si considera anche la **standard deviation**, quando oltre al diverso offset si ha una diversa **amplitude**, o **variance** (la linea blu è più piatta della verde). Si ricorre all'**amplitude scaling** nel caso di *time series* con **different variability**, che vanno su e giù in modo diverso.

$$Q = (Q - \text{mean}(Q)) / \text{std}(Q)$$

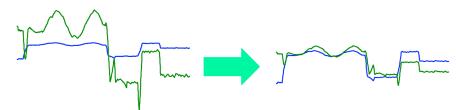
$$C = (C - \text{mean}(C)) / \text{std}(C)$$

$$D(Q,C)$$

Transformation I: Offset Translation

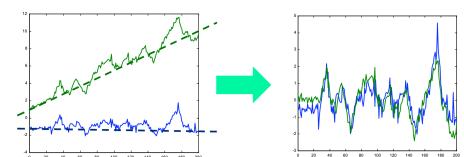


Transformation II: Amplitude Scaling

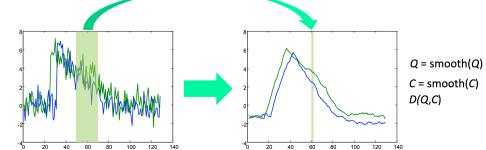


Transformation III: Linear Trend

- Removing linear trend: fit the best fitting straight line to the time series, then subtract that line from the time series.



The intuition behind removing noise is to average each datapoints value with its neighbors.



Due *time series* possono essere molto simili ma seguire un diverso *trend*, che si dovrà quindi linearizzare. A lato, linearizzazione del trend, offset translation e amplitude scaling.

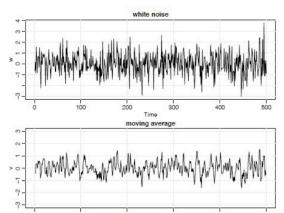
Per rimuovere il **rumore** si applicano delle funzioni di **smoothing** a specifiche parti della *time series*. Un esempio di funzione di smoothing è la **media mobile** (*moving average*).

La media mobile è calcolata fissando un punto, poi calcolando la media dei successivi W punti, punto iniziale compreso (nell'immagine: si parte da 20, si calcola la media di $20+24+22$). In questo modo si avranno valori "missing" all'inizio e alla fine. Spesso si ripete il valore subito prima/dopo (22, 24.3), o si calcola la media degli ultimi valori (es: $24+24.3/2$). Approccio semplice ed efficiente, deve essere però valutato bene il W .

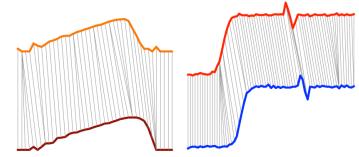
Moving Average

- Noise can be removed by a **moving average** (MA) that smooths the TS.
- Given a window of length w and a TS t , the MA is applied as follows
- $t_i = \frac{1}{w} \sum_{j=i-w/2}^{i+w/2} t_j$ for $i = 1, \dots, n$
- For example, if $w=3$ we have
- $t_i = \frac{1}{3} (t_{i-1} + t_i + t_{i+1})$

time	value	ma
t1	20	-
t2	24	22.0
t3	22	24.0
t4	26	24.3
t5	25	-



Fixed Time Axis. Sequences are aligned "one to one". Greatly suffers from the misalignment in data. Euclidean.

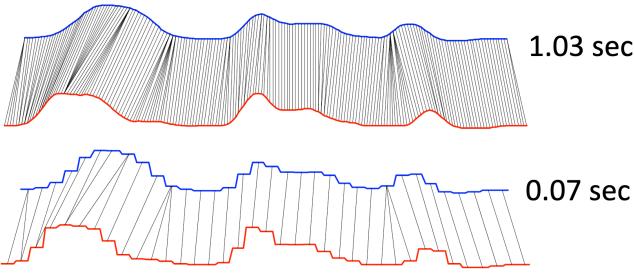


Warped Time Axis. Nonlinear alignments are possible. Can correct misalignments in data. Dynamic Time Warping.

Un'altra possibile distanza è il **Dynamic Time Warping**. Talvolta le *time series* possono essere simili, ma a un certo momento temporale procedere a differenti velocità. Si crea così uno "sfasamento" e la distanza euclidea non riuscirebbe a calcolare (linearmente) la distanza minima. Inoltre, poiché gli assi *are warping* si può calcolare la DTW di valori di dimensionalità differente, senza normalizzare

Il calcolo del DTW è molto oneroso in termine di risorse. Per questo motivo si ricorre a due tecniche di semplificazione:

- 1) Approssimazione, che vedremo in seguito (fig. 1);
- 2) *Global Constraints* (fig. 2). Si specifica un range di celle che si analizzano, per non calcolare l'intera matrice. Un Warping Width compreso fra 2 e 5 è solitamente sufficiente.

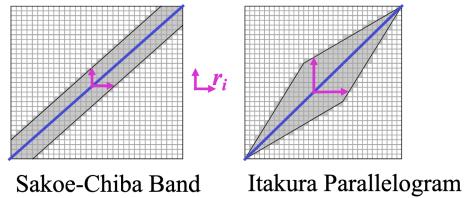


1.03 sec

0.07 sec

A global constraint constrains the indices of the warping path $w_k = (i,j)_k$ such that $j-r \leq i \leq j+r$, where r is a term defining allowed range of warping for a given point in a sequence.

r can be considered as a *window* that reduces the number of calculus.



Sakoe-Chiba Band

Itakura Parallelogram

Approximation and Clustering

L'**approssimazione** di una *time series* è una particolare forma di *Dimensionality Reduction* per questo tipo di dati. Non va confusa con la **compressione**: in questo caso **non necessariamente lo spazio compresso è comprensibile**, mentre con l'approssimazione sì.

Vari metodi per comprimere le slide sono sufficienti. Nel codice questi metodi vanno applicati dopo la normalizzazione. Ricordiamo qui che:

Discrete Fourier Transform e **Discrete Wavelet Transform**: Rappresentano la TS come combinazione di seno/coseno o wavelet, mantenendo N coefficienti (la metà nel caso di DFT). Molti coefficienti di Fourier hanno piccola amplitudine e sono eliminati. Quelli di Wavelet riescono a rappresentare *local subsection of the data*.

Singular Value Decomposition (SVD): La TS è rappresentata come *combinazione lineare di eigenwaves*. Solo le prime includono la maggior parte della varianza del segnale, per cui solo le prime sono mantenute.

Piecewise Linear Approximation (PLA): La TS è rappresentata come sequenza di linee. Di ciascuna sappiamo la **lunghezza** e la **altezza SX** (quella DX può essere inferita dal segmento successivo). Il dilemma del metodo è la selezione del K dei segmenti – si deve scegliere fra compattezza e precisione. **Pro: Comprime i dati, filtra il rumore, contesti non Euclidei.**

Piecewise Aggregate Approximation (PAA): Divide la TS in N segmenti di **pari lunghezza** (adopera quindi anche una *dimensionality reduction*). Di ciascuno sappiamo il valore medio. **Pro: Veloce da calcolare, contesti non Euclidei, distanza Euclidea pesata.**

Adaptive Piecewise Constant Approximation (APCA): Variante della PAA. Ogni segmento ha due valori: lunghezza, valore medio. **Aree ad alta attività avranno tanti segmenti, aree a bassa attività pochi.** **Pro: Come sopra, ma più precisa!**

Symbolic Aggregate Approximation (SAX): La TS è convertita con la PAA in parti uguali. Lo spazio è poi diviso in k parti equiprobabili (un segmento avrà la stessa probabilità di finire in ciascuna regione). **Questo k sarà il numero di lettere dell'alfabeto**, ciascuna delle quali sarà assegnata ad ogni regione (es.: 3 => a, b, c).

ESERCIZIO: Calcolo del DTW

Lezione 14 parte 2: da minuto 26 inizio spiegazione su come calcolare il DTW.

Lezione 15 parte 1: fino a minuto 20 esercizio nel concreto. Nuovamente a minuto 31

Slide 27: Matrice di calcolo completa

<http://didawiki.cli.di.unipi.it/lib/exe/fetch.php/>

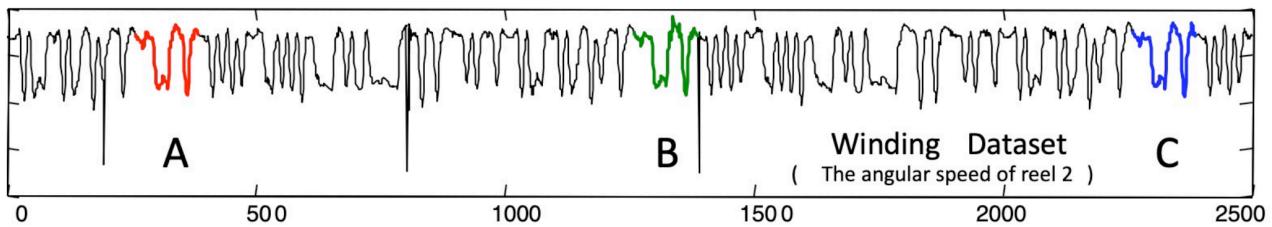
ESERCIZIO: Approssimazione

Presente nelle slide ma **non verrà chiesto all'esame**

Per il clustering: Guidotti è più interessato ad usare diversi tipi di cluster (quelli inclusi del codice, quelli classici di SKLearn di DM1, diverse metriche) che ai risultati nudi e crudi. Ritiene inoltre interessante fare il clustering dopo aver applicato un *pair wise approximation algoritm*, notando la diversa posizione dei centroidi.

Motifs

Un *motif* è un pattern ricorrente presente in una singola *time series*. Trovare *motifs* è un prerequisito per il *pattern mining*, classificare trovando *typical prototypes of each class* o trovare eventuali anomalie.



Come trovare dei motif? Una possibile è il "metodo forza bruta", ma computazionalmente molto esoso. Un metodo più efficiente (ma che **non sarà usato nel progetto**) si basa sul SAX e si ispira alla bioinformatica.

- 1) Trasformiamo la time series con il SAX (ottenendo quindi una serie di lettere) impostando a lettere e una sax window di 4;
- 2) Si oscurano delle colonne random. In quelle rimaste si contano quante volte la stessa combinazione di lettere appare. Si aggiorna quindi la matrice con il conto;
- 3) Si prendono altre colonne random. Si aggiorna nuovamente la matrice.

1	a	c	b	a		1			
2	b	c	a	b		2			
:	:	:	:	:		1	3		
58	a	c	c	a		27	2	1	
:	:	:	:	:		3	2	2	1
985	b	c	c	c		0	1	2	1
	1	2	3	4		1	2	:	58
								:	985

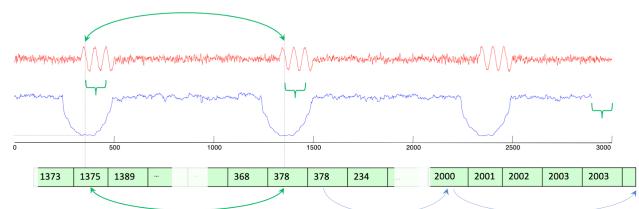
Quante colonne si considerano? Più si considerano, più l'algoritmo è preciso (ma computazionalmente esoso). Considerandone di meno, si perde in precisione. Inoltre il risultato è molto legato ai parametri dell'approssimazione.

Un'alternativa è quindi il **Matrix Profile**, che analizza la *time series subsequences* dopo *subsequences*, cercando appunto la *subsequences* più simile. Si crea così una *time series "parallela"* (**matrix profile**) che indice le distanze e il **matrix profile index** che indice l'indice delle subsequences più simili. **Quando la matrix profile è al minimo, la distanza è al minimo e quindi a colpo d'occhio possiamo vedere dove si localizzano i motif.**

The MP index allows to find the nearest neighbor to any subsequence in constant time.
Note that the pointers in the matrix profile index are not necessarily symmetric.

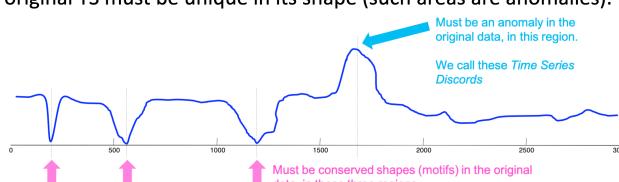
If A points to B, then B may or may not point to A

The classic TS motif: the two smallest values in the MP must have the same value, and their pointers must be mutual.

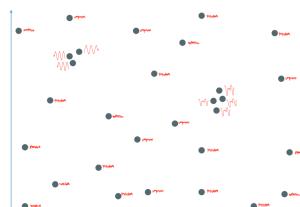


Se le "valli" nella matrix profile sono i motif, i "monti" sono i **discord**, cioè le anomalie.

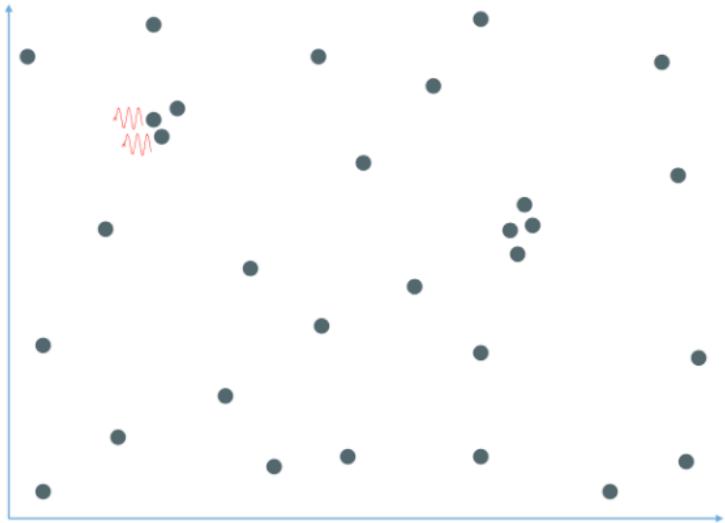
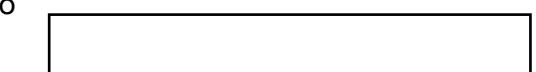
For relatively high values, you know that the subsequence in the original TS must be unique in its shape (such areas are anomalies).



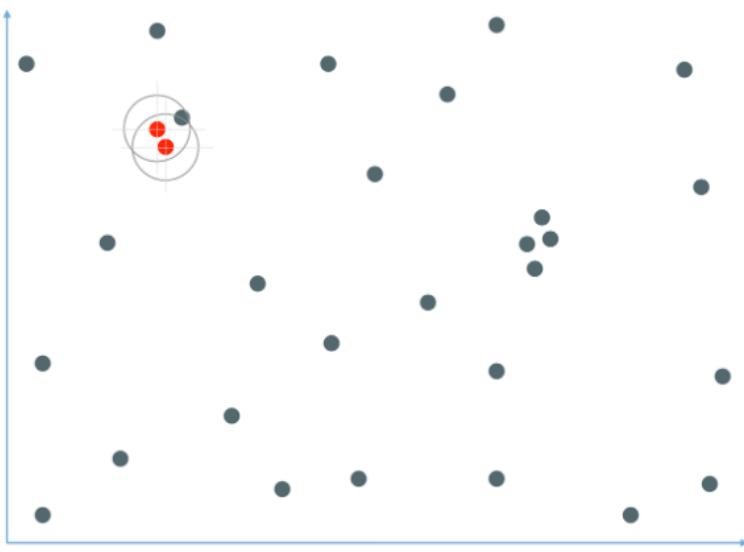
È anche possibile "mappare" la *matrix profile* in uno spazio. In questo caso, le regioni più dense dello spazio corrispondono alle "valli" della matrix profile. Questa



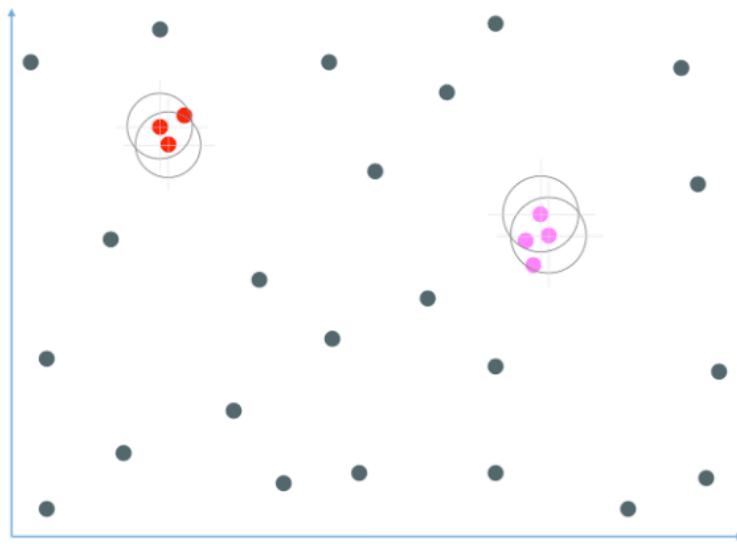
rappresentazione può aiutare a trovare i *top-k motifs*. Nella pratica è una forma di clustering sia gerarchico che basato



- We need a parameter R.
- $1 < R <$ (small number, say 3)
- Lets make R = 2 for now.
- We begin by finding the nearest pair of points, the *motif pair*....
- This the pair of subsequences corresponding to lowest pair of values in the MP



- We find the nearest pair of points are D1 apart.
- Lets draw a circle, $D_1 \times R$, around both points.
- Any points that are within either of these circles, are added to this motif, in this case just one.
- The Top-1 motif has three members, it is done.

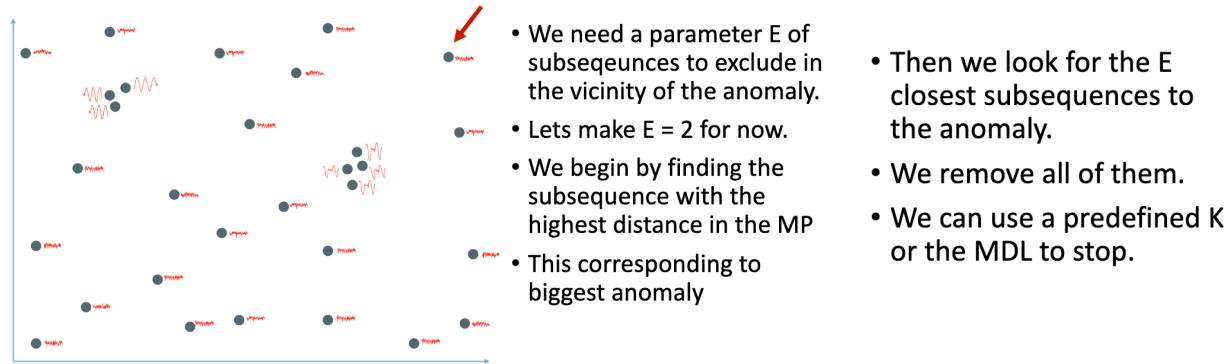


- Now lets find the Top-2 motif. We find the *nearest pair of points*, excluding anything from the top motif.
- The nearest pair of points are D2 apart.
- Lets draw a circle $D_2 \times R$, around both points.
- Any points that are within either of these circles, is added to this motif, in this case there are two for a total of four items in the Top-2 Motif

sulla densità.

D_1 è fissato *a priori*, ad esempio = 10, o **estrando lo dal dataset** (la distanza fra i due punti) Nelle interazioni successive aumenta sempre xR . Se $R = 2$: $D_2 = 20$, $D_3 = 30$ etc.. Quando ci si ferma? Si decide *a priori*, o secondo la *Minimum Description Length*.

Per quanto riguarda le anomalie, si procede in modo analogo osservando però i punti più lontani dagli altri (che rappresentano i "monti" nella matrix profile). Si parte dal punto più isolato (in figura con la freccia), cioè che ha la *highest distance*, poi si elimina insieme agli E punti più vicini.

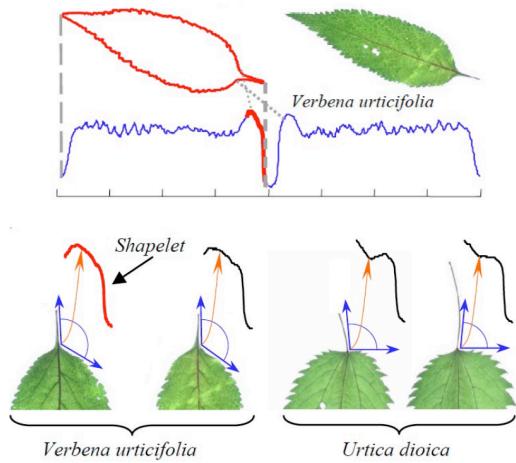


In conclusione la *matrix profile* è un **modello di rappresentazione di una singola time series**, da cui si possono poi estrarre *motifs* e anomalie. Il parametro più difficile da settare è $l'm$ che definisce la window della *time series*: non deve essere troppo piccolo, o la porzione non è rappresentativa, né troppo grande, o è difficile trovare set di punti uguali. Come fare con set di punti molto grandi? Si deve ricorrere a tecniche di approssimazione o al DTW, ma il punto di una matrix profile è proprio *non* ricorrere a tecniche del genere e confrontare le *time series* non manipolate, per non far dipendere i *motif* trovati da un'approssimazione.

Classification

Per la classificazione delle *time series* si ricorre solitamente a Deep Neural Network o metodi ensabled. Nella pratica, si può ottenere la stessa performance con uno **Shapelet-based Classifier**, rappresentando la *time series* as a vector of distances with representative subsequences (cioè gli *shapelets*) e dando in pasto questo vettore ai classificatori.

Ma cos'è uno shapelet? L'idea di base è che **ogni immagine può essere rappresentata come time series guardando al bordo**. La parte in rosso dell'immagine è il discriminante fra i due tipi di foglie, ovvero uno shapelet. **Uno shapelet è una sub-seguenza di una TS che è maximally representative of a class** (un solo shapelet può essere non sufficiente).



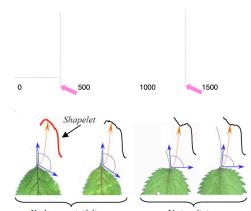
- A **motif** is a repeated pattern/subsequence in a given TS.

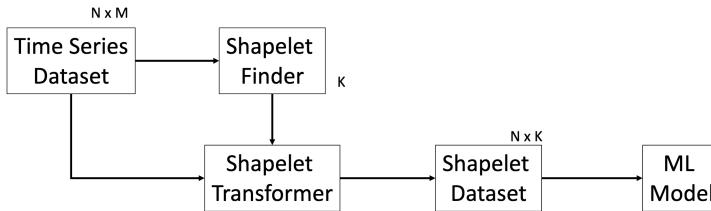
Motif = Pattern di una singola TS.

Shapelet = Pattern di più TS, che ne determinano la classe.

- A **shapelet** is a pattern/subsequence which is maximally representative of a class with respect to a given dataset of TSs.

Nella pratica, potremmo avere 4 TS, da cui estraiamo in tutti 3 shapelet, 3 "frammenti rappresentativi della classe". Poi "scorriamo" ciascun shapelet lungo la TS, per cercare dove combacia (se combacia), ottenendo così la *distanza* fra





shapelet e TS. In modo poco intuitivo, qui per "distanza" si intende quanto uno shapelet combacia.

- Shapelets are TS subsequences which are maximally representative of a class.
- Shapelets can provide interpretable results, which may help domain practitioners better understand their data.
- Shapelets can be significantly more accurate/robust because they are *local features*, whereas most other state-of-the-art TS classifiers consider *global features*.

Grazie agli shapelet è anche possibile non lavorare più con una *time series* ma con queste distanze. In questo senso possiamo parlare degli *shapelet* come una "dimensionality reduction" delle TS.

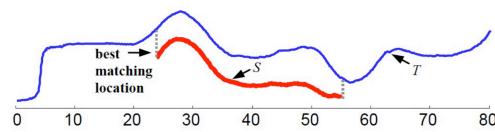
Non c'è un singolo modo per estrarre gli shapelet. Noi ne vedremo solo uno - il metodo forza bruta. Si setta una finestra di lunghezza "m" e muovendola si estraggono diverse sub-sequenze, tutte della stessa lunghezza, da più TS, ciascuna con già una classe (nell'immagine sotto, quadrati rossi e cerchi blu). La procedura si può effettuare anche con più "m". Si calcola poi la *distanza* per ciascun "candidate shapelet".

Distance from the TS to the subsequence $\text{SubsequenceDist}(T, S)$ is a distance function that takes time series T and subsequence S as inputs and returns a nonnegative value d , which is the distance from T to S .

$$\text{SubsequenceDist}(T, S) = \min(\text{Dist}(S, S')), \text{ for } S' \in S_T^{(S)}$$

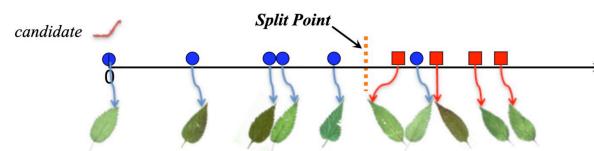
where $S_T^{(S)}$ is the set of all possible subsequences of T

Intuitivamente, it is the distance between S and its best matching location in T .

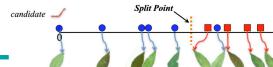


Testing The Utility of a Candidate Shapelet

- Arrange the TSs in the dataset D based on the distance from the candidate.
- Find the optimal split point that maximizes the information gain (same as for Decision Tree classifiers)
- Pick the candidate achieving best utility as the shapelet



Entropy



- A TS dataset D consists of two classes, A and B.
- Given that the proportion of objects in class A is $p(A)$ and the proportion of objects in class B is $p(B)$,
- The **Entropy** of D is: $I(D) = -p(A)\log(p(A)) - p(B)\log(p(B))$.
- Given a strategy that divides the D into two subsets D_1 and D_2 , the information remaining in the dataset after splitting is defined by the weighted average entropy of each subset.
- If the fraction of objects in D_1 is $f(D_1)$ and in D_2 is $f(D_2)$,
- The total entropy of D after splitting is $\hat{I}(D) = f(D_1)I(D_1) + f(D_2)I(D_2)$.

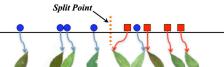
Information Gain

- Given a certain split strategy sp which divides D into two subsets D_1 and D_2 , the entropy before and after splitting is $I(D)$ and $\hat{I}(D)$.

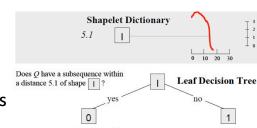
• The **information gain** for this splitting rule is:

$$\begin{aligned} \text{Gain}(sp) &= I(D) - \hat{I}(D) = \\ &= I(D) - f(D_1)I(D_1) + f(D_2)I(D_2). \end{aligned}$$

- We use the distance from T to a shapelet S as the splitting rule sp .



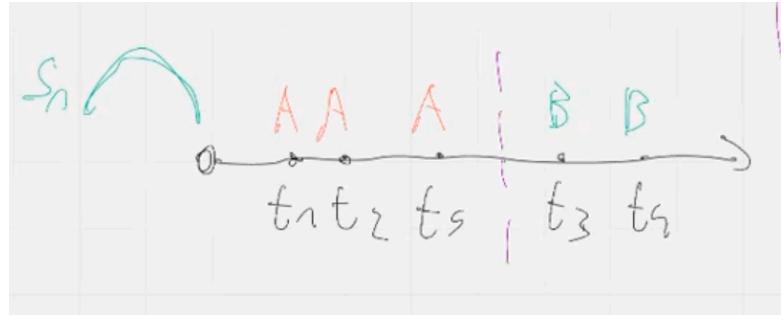
Split point distance from shapelet = 5.1



Si ripete la procedura con il successivo "candidate shapelet" e si seleziona alla fine quello (o quelli) con l'Information Gain maggiore.

Ricapitolando:

- uno shapelet è una parte di una TS utile per identificare una classe;
- quello che vogliamo fare è trasformare il dataset di TS in un dataset in cui, per ogni TS, abbiamo la sua distanza dai vari shapelet (distanza TS1 da S1, TS1 da S2, TS1 da S3; distanza TS2 da S1, TS2 da S2, TS3 da S3 ecc.);
- come si scelgono S1, S2, S3...? Per il punto 1, sono quelli hanno "discriminative power" più alto, cioè servono a identificare delle classi;
- quando parliamo di "distanza fra TS e Shapelet" ci riferiamo alla distanza minima fra TS e Shapelet fra tutte le possibili distanze. In altre parole, al *best alignment*;
- **Metodo forza bruta.** Si fissa un m (es.: m=4) e si muove questa "finestra" per tutta la, e tutte le TS, modificando anche m. In questo modo si ha un insieme di candidati. Si devono poi "confermare" gli shapelet. Ordiniamo per ciascun shapelet le distanze con le TS e cerchiamo di vedere se, ordinate in quel modo, è possibile dividere le distanze massimizzando l'*information gain* per ciascuna classe. Dividere in quel punto nell'immagine a lato massimizza la divisione fra le classi A e le classi B.



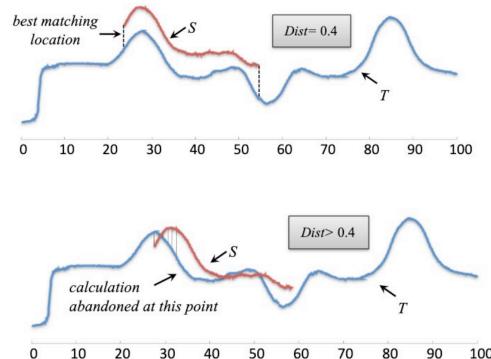
Questa procedura è particolarmente esosa: per ogni candidato va computata la distanza con *tutte* le TS, per ogni TS vanno calcolati più candidati. Ci sono due tecniche di velocizzazione.

Nel caso della **Distance Early Abandon** si riduce *the distance computation time between two TS* (= velocizziamo il calcolo) e nel caso della **Admissible Entropy Pruning** si riduce *the number of distance calculations* (= evitiamo di fare dei calcoli).

Distance Early Abandon

Si tiene in memoria la distanza migliore, se ne troviamo una più lunga la escludiamo subito

- We only need the minimum distance.
- Method
 - Keep the best-so-far distance
 - Abandon the calculation if the current distance is larger than best-so-far.

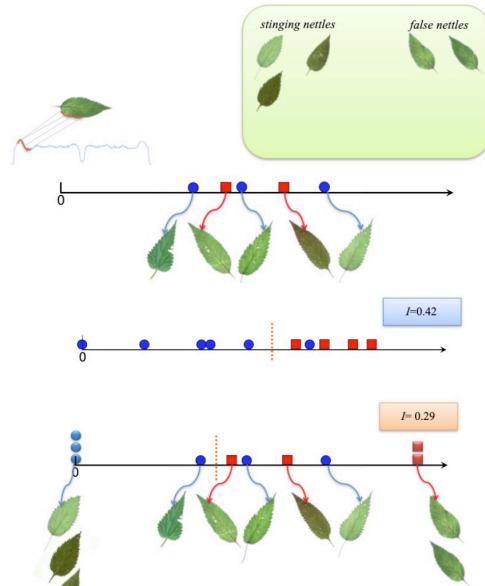


Admissible Entropy Pruning

De facto c'è bisogno solo degli shapelet che definiscono delle classi; cioè, ogni classe ha associato uno shapelet. Si simula la *best possibile division* con un piccolo set di distanze, se la si trova non si provano altri candidati, altrimenti sì.

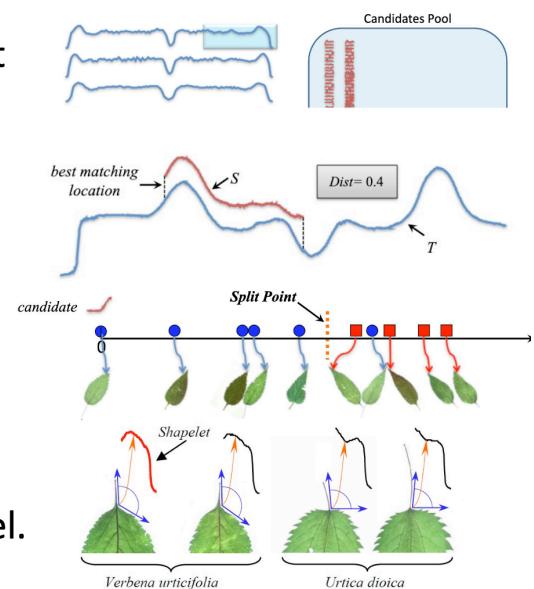
Infine, c'è un'altra alternativa: non estrarre uno shapelet ma creare uno shapelet partendo da uno creato random, provarlo più volte e pian piano adattarlo (un po' come un neural network).

- We only need the best shapelet for each class
- For a candidate shapelet
 - We do not need to calculate the distance for each training sample
 - After calculating some training samples, the upper bound of information gain < best candidate shapelet
 - Stop calculation
 - Try next candidate



Shapelet Summary

1. Extract all possible subsequences of a set given lengths (candidate shapelets)
2. For each candidate shapelet
 1. Calculate the distance with each time series keeping the minimum distance (best alignment)
 2. Evaluate the discriminatory effect of the shapelet through the Information Gain
3. Return the k best shapelets with the highest Information Gain.
4. Transform a dataset and train a ML model.



So long and thanks for all the fish.