ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Αποδοτικοί αλγόριθμοι για την Συσταδοποίηση Χρονοσειρών μέσω Προσέγγισης με Gaussian Processes

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μιχαήλ Α. Ξεφτέρης

Επιβλέπων: Δημήτριος Φωτάκης
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2021

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Αποδοτικοί αλγόριθμοι για την Συσταδοποίηση Χρονοσειρών μέσω Προσέγγισης με Gaussian Processes

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μιχαήλ Α. Ξεφτέρης

Επιβλέπων: Δημήτριος Φωτάκης
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 9η Ιουλίου 2021.

.....	.....	.....
Δημήτριος Φωτάκης	Αριστείδης Παγουρτζής	Γιώργος Στάμου
Αναπληρωτής Καθηγητής	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.
Ε.Μ.Π.		

Αθήνα, Ιούλιος 2021

.....

Μιχαήλ Α. Ξεφτέρης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Ευχαριστίες

Ολοκληρώνοντας τις σπουδές μου στο ΕΜΠ, θα ήθελα να ευχαριστήσω ιδιαίτερα τον επιβλέποντα καθηγητή μου κ. Φωτάκη για την εμπιστοσύνη που έδειξε στις δυνατότητές μου, τη συνεργασία μας, καθώς και την πολύτιμη βοήθειά του στη συνέχιση της ακαδημαϊκής μου πορείας. Επίσης, θα ήθελα να ευχαριστήσω τον Παναγιώτη Πατσιλινάκο για την άμεση και εποικοδομητική συνεργασία μας στην υλοποίηση της παρούσας διπλωματικής.

Τέλος, ένα μεγάλο ευχαριστώ στην αδελφή μου, τους γονείς μου και σε όλους τους φίλους μου για τη στήριξή τους σε όλη τη διάρκεια των σπουδών μου.

Περίληψη

Διάφορες τεχνικές εξόρυξης δεδομένων χρησιμοποιούνται στις μέρες μας για την ανάλυση δεδομένων. Ανάμεσα τους, η συσταδοποίηση είναι η πιο διαδεδομένη και χρησιμοποιείται σε περιπτώσεις που δεν υπάρχει κάποια προηγούμενη γνώση για τη δομή των συστάδων. Δεδομένα από πολλά πεδία όπως η οικονομία, η υγεία κ.α. αποθηκεύονται σε μορφή χρονοσειρών. Η συσταδοποίησή τους έχει ποικίλες εφαρμογές στο γονιδιώμα, στην ιατρική, στα οικονομικά. Το πρόβλημα είναι δύσκολο εξαιτίας του θορύβου και της μεγάλης διαστατικότητας που εκ φύσεως έχουν οι χρονοσειρές. Σε αυτή τη διπλωματική ασχολούμαστε με τη συσταδοποίηση χρονοσειρών με βάση το σχήμα τους. Το σημαντικότερο συστατικό των αλγορίθμων αυτής της κατηγορίας είναι η επιλογή του κατάλληλου μέτρου ομοιότητας. Το μέτρο αυτό θα πρέπει να συγκρίνει αποτελεσματικά τα σχήματα των χρονοσειρών. Η πιο διαδεδομένη τέτοια απόσταση είναι η Dynamic Time Warping (DTW), η οποία όμως έχει τετραγωνική πολυπλοκότητα που επηρεάζει σημαντικά την πολυπλοκότητα των αλγορίθμων συσταδοποίησης.

Οι περισσότερο υποσχόμενες λύσεις για τη μείωση της παραπάνω πολυπλοκότητας περιλαμβάνουν πρώτα την εφαρμογή μεθόδων για τη μείωση της διαστατικότητας των χρονοσειρών και έπειτα τη χρήση κλασικών αλγορίθμων συσταδοποίησης στα μειωμένης διαστατικότητας δεδομένα. Σε αυτή τη διπλωματική προτείνουμε μία νέα μέθοδο δύο σταδίων για τη συσταδοποίηση χρονοσειρών. Πρώτα μοντελοποιούμε τις χρονοσειρές με ορισμένα σημεία που ονομάζονται inducing points χρησιμοποιώντας Sparse Gaussian Process Regression [68], η οποία είναι μια προσεγγιστική μέθοδος για Gaussian Process Regression. Στη συνέχεια, οι χρονοσειρές περιγράφονται με τα λιγότερα σε αριθμό inducing points τα οποία οργανώνονται σε συστάδες με την εφαρμογή του αλγορίθμου k-means χρησιμοποιώντας ως μέτρο απόστασης μια τροποποιημένη εκδοχή της DTW. Τα πειράματα μας δείχνουν ότι η προσέγγισή μας δίνει μια γρήγορη και αποδοτική μέθοδο συσταδοποίησης.

Λέξεις κλειδιά

χρονοσειρά, συσταδοποίηση, Gaussian Process, DTW, k-means

Abstract

Various data mining techniques are currently being used to analyse data within different domains. Among all these approaches, clustering is the most-used technique in cases when category information is not available. The data in various systems such as finance, healthcare, and business, are stored as time series. Clustering such complex data can discover patterns which have valuable information. Time series clustering has been widely applied to genome data, medicine, finance, and in general, in any domain where pattern recognition is important. It is a challenging task because of the inherent noise and high dimensionality of time series data. This thesis is concerned with finding similar time series in shape. For shape-based clustering, the shape of time series is the key factor in identifying pattern similarity. The most important aspect of clustering algorithms is the similarity measure used to compare the time series shapes. Dynamic Time Warping (DTW) distance is particularly popular in that context. However, DTW has a quadratic time complexity that slows down the process of clustering.

The most promising solutions to the time complexity problem involve first performing dimensionality reduction on the time series data, and then clustering the reduced data with a conventional algorithm. In this thesis, we propose a two-stage framework for clustering time series data. First, we model the raw series by a set of inducing data points using Sparse Gaussian Process Regression (SGPR) [68], which is an approximation method for Gaussian Process Regression. The series as described by a lower number of data points are then grouped by applying k-means algorithm with a modified version of DTW as the distance measure. The experimental results indicate that the proposed approach leads to a fast, scalable and accurate clustering framework.

Keywords

time series, clustering, Gaussian Process, DTW, k-means

Contents

List of Figures

List of Tables

Κεφάλαιο 1

Εκτεταμένη Ελληνική Περίληψη

Αυτό το κεφάλαιο περιλαμβάνει μία περιληπτική παρουσιάση των περιεχομένων αυτής της διπλωματικής εργασίας στα ελληνικά.

1.1 Συσταδοποίηση Χρονοσειρών

Η χρονοσειρά είναι ένα σύνολο από παρατηρήσεις που περιγράφουν την εξέλιξη της συμπεριφοράς ενός μεγέθους στο χρόνο. Τα μεγέθη που περιγράφονται μπορούν να είναι οποιασδήποτε φύσης αρκεί να μπορούν να ποσοτικοποιηθούν. Οι αυξανόμενες δυνατότητες αποθήκευσης δεδομένων τις τελευταίες δεκαέτιες επιτρέπουν σε όλο και περισσότερες εφαρμογές τη διατήρηση δεδομένων για μεγάλο χρονικό διάστημα. Αυτό έχει ως αποτέλεσμα όλο και μεγαλύτερος όγκος δεδομένων να παράγονται στη μορφή χρονοσειρών σε ποικίλα πεδία. Συνεπώς οι χρονοσειρές συναντώνται σε πολλά πεδία όπως η οικονομία, οι κοινωνικές επιστήμες, η επιδημιολογία και οι φυσικές επιστήμες.

Μια χρονοσειρά X μήκους n είναι μια ακολουθία σημείων:

$$X = [(x_1, t_1), (x_2, t_2), ..., (x_n, t_n)] \quad || \quad (t_1 < t_2 < ... < t_i < ... < t_n) \quad (1.1)$$

οπου το x_i είναι η τιμή μιας παρατήρησης και το t_i αναπαριστά τη χρονική στιγμή που η παρατήρηση x_i μετρήθηκε. Αν έχουμε ισαπέχουσες χρονικές στιγμές οι τιμές του χρόνου t_i μπορούν να παραληφθούν.

Συσταδοποίηση ονομάζεται η διαδικασία εκείνη κατά την οποία ένα σύνολο από αντικείμενα, διαχωρίζονται σε ομάδες. Η καταχώρηση αντικειμένων στην ίδια ομάδα μεταφράζεται ως ομοιότητα των αντικειμένων αυτών και αντίστροφα (αντικείμενα που ανήκουν σε διαφορετικές ομάδες είναι λιγότερο όμοια).

Η συσταδοποίηση χρονοσειρών, λόγω της διαρκούς παραγωγής τέτοιων δεδομένων στις μέρες μας, είναι ιδιαίτερα σημαντική σε πολλά διαφορετικά πεδία. Η ανακάλυψη patterns των δεδομένων μας με τη συσταδοποίησή χρονοσειρών έχει ποικίλες εφαρμογές στην οικονομικές αγορές, στην ανάλυση ιατρικών δεδομένων, στη μελέτη της θερμοκρασίας κ.α. Μπορεί να χρησιμοποιηθεί, παραδείγματος χάριν, για την κατανοήση της συμπεριφοράς μια μερίδας χρηστών σε κάποια δραστηριότητα. Επιπλέον, η συσταδοποίηση χρονοσειρών είναι χρήσιμη για την αναγνώριση ιδιομορφιών και απότομων μεταβολών στις τιμές των χρονοσειρών, όπως για τον εντοπισμό κάποιας ασθένειας από τη σύγκριση δεδομένων από ιατρικές εξετάσεις. Επιπλέον, η συσταδοποίηση χρονοσειρών χρησιμοποιείται και ως ρουτίνα άλλων αλγορίθμων όπως η πρόβλεψη και το indexing. Στην διπλωματική αυτή θα ασχοληθούμε συγκεκριμένα με τη διαχωριστική συσταδοποίηση με βάση το σχήμα των χρονοσειρών (partitioning shape-based clustering). Στην διαχωριστική συσταδοποίηση, ο στόχος είναι να χωρίσουμε τα αντικείμενα σε k ομάδες, και κάθε ομάδα να περιέχει τουλάχιστον ένα αντικείμενο. Ο αριθμός k των clusters μας είναι γνωστός πριν την έναρξη της διαδικασίας. Ο πιο διαδεδομένος αλγόριθμος διαχωριστικής συσταδοποίησης είναι ο αλγόριθμος k-means.

Ο σχεδιασμός αλγορίθμων για τη συσταδοποίηση χρονοσειρών είναι ιδιαίτερα απαιτητικός εξαιτίας της φύσης των χρονοσειρών. Οι χρονοσειρές είναι πολύπλοκα δεδομένα μεγάλης διαστατικότητας, περιέχουν θόρυβο και απαιτούν σημαντικό χώρο για την αποθήκευσή τους. Αυτό έχει ως αποτέλεσμα οι αντίστοιχοι αλγόριθμοι για τη συσταδοποίησή τους να έχουν μεγάλη χρονική πολυπλοκότητα.

Για την κατηγορία συσταδοποίησης που μας ενδιαφέρει είναι απαραίτητη πρώτα απ΄ όλα μια απόσταση που να μετρά πόσο όμοιες είναι δύο χρονοσειρές με βάση το σχήμα τους. Μια τέτοια απόσταση θα πρέπει να μην επηρεάζεται από τις μεταθέσεις στο χρόνο, από τον διαφορετικό ρυθμό δειγματοληψίας και τα διαφορετικά μήκη που μπορεί να έχουν δύο χρονοσειρές. Γι΄ αυτούς τους λόγους, μια απλή απόσταση όπως η Ευκλείδια δεν είναι κατάλληλη για τη σύγκριση χρονοσειρών. Τα παραπάνω προβλήματα λύνονται με αποστάσεις που έχουν την δυνατότητα να αντιστοιχούν ένα με πολλά σημεία. Αυτες οι αποστάσεις ονομάζονται ελαστικές. Η ελαστική απόσταση που θεωρείται ευρέως η πιο αποτελεσματική για χρονοσειρές είναι η Dynamic Time Warping (DTW).

1.1.1 Dynamic Time Warping

Η Dynamic Time Warping (DTW) χρησιμοποιείται για τον υπολογισμό της ομοιότητας μεταξύ δύο χρονοσειρών που διαφέρουν σε μήκος ή σε ταχύτητα. Το βασικό της πλεονέκτημα είναι ότι επιτρέπει τη μη γραμμική στρέβλωση (επιμήκυνση ή συρρύκνωση) μιας αλληλουχίας τιμών ώστε να ταιριάξει με μία άλλη

αλληλουχία ακόμα και αν αυτές παρουσιάζουν χρονική υστέρηση. Επιπλέον, οι δύο χρονοσειρές που συγκρίνονται με την απόσταση DTW δεν χρειάζεται να έχουν το ίδιο μήκος. Βασικό της μειονέκτημα είναι η τετραγωνική χρονική πολυπλοκότητα ως προς το μήκος T της χρονοσειράς, $O(T^2)$.

Έστω δύο χρονοσειρές x, y μήκους n, m αντίστοιχα. Για τον υπολογισμό της απόστασης DTW δημιουργείται ένας πίνακας διαστάσεων $n \times m$. Κάθε στοιχείο $c_{i,j}$ (τοπικό κόστος) στη θέση (i, j) υπολογίζεται από τον τύπο:

$$c_{i,j} = (x_i - y_j)^2, \quad i \in [1:n], j \in [1:m] \tag{1.2}$$

Στη συνέχεια υπολογίζεται ένα warping path όπου ορίζεται ως:

$$W = w_1, w_2, ..., w_K, \quad max(n, m) \leq K \leq n + m - 1 \tag{1.3}$$

Τέλος, η απόσταση DTW υπολογίζεται από τον τύπο:

$$DTW(x, y) = \sqrt{\sum_{k=1}^{K} w_k} \tag{1.4}$$

1.1.2 DTW Barycenter Averaging

Ένα δεύτερο απαραίτητο συστατικό για τους περισσότερους αλγορίθμους διαχωριστικής συσταδοποίησης είναι ο υπολογισμός της χρονοσειράς-κέντρο ενός σετ χρονοσειρών (εκπρόσωπός τους). Πιο συγκεκριμένα θέλουμε να βρούμε μια χρονοσειρά η οποία ελαχιστοποιεί το άθροισμα των αποστάσεων με όλες τις χρονοσειρές ενός συνόλου. Ψάχνουμε να βρούμε τη λεγόμενη μέση χρονοσειρά του συνόλου. Το πρόβλημα αυτό για την απόσταση DTW λύνεται σε εκθετικό χρόνο και έτσι έχουν προταθεί διάφοροι ευρεστικοί αλγόριθμοι για τη προσέγγισή του όπως ο NonLinear Alignment and Averaging Filters (NLAAF) [30], ο Prioritized Shape Averaging (PSA) [47], ο Cross-Word Reference Template (CWRT) [4]. Οι παραπάνω μέθοδοι ανήκουν στην κατηγορία των λεγόμενων προοδευτικών μεθόδων. Εδώ θα περιγράψουμε μόνο την state-of-the-art μέθοδο για τον υπολογισμό της μέσης χρονοσειράς.

Ο γρηγορότερος και πιο αποτελεσματικός ευρεστικός αλγόριθμος που υπάρχει στη βιβλιογραφία ειναι ο DTW Barycenter Averaging (DBA) [52]. Ο DBA είναι ένας επαναληπτικός αλγόριθμος που στοχεύει στην ελαχιστοποίηση της απόστασης DTW της χρονοσειράς-κέντρου με όλες τις χρονοσειρές του σετ.

Ο αλγόριθμος ξεκινάει με μια προσωρινή χρονοσειρά-κέντρο (μπορεί να είναι μια τυχαία) η οποία αναβαθμίζεται σε κάθε επανάληψη. Κάθε επανάληψη περιλαμβάνει τα εξής:

- Υπολογισμός του DTW ανάμεσα στη χρονοσειρά-κέντρο και σε κάθε άλλη χρονοσειρά για να βρούμε τις συνδέσεις ανάμεσα στις συντεταγμένες της χρονοσειράς-κέντρου με όλες τις συντεταγμένες των χρονοσειρών του σετ.

- Η κάθε συντεταγμένη της χρονοσειράς-κέντρου γίνεται ίση με την τιμή του βαρύκεντρου των συνδεδεμένων με αυτή συντεταγμένων που υπολογίστηκαν στο προηγούμενο βήμα.

Ο DBA είναι, όπως αναφέρθηκε, ένας επαναληπτικός αλγόριθμος που συγκλίνει σε κάποιες επαναλήψεις. Κάθε επανάληψη έχει υπολογιστική πολυπλοκότητα $\Theta(N \cdot T^2)$ για το πρώτο βήμα και $\Theta(N \cdot T)$ για το δεύτερο βήμα, όπου N είναι ο αριθμός των χρονοσειρών και T το μήκος τους. Αν υποθέσουμε ότι ο αλγόριθμος τρέχει για I επαναλήψεις τότε η συνολική πολυπλοκότητα του αλγορίθμου είναι:

$$\Theta(I(N \cdot T^2 + N \cdot T)) = \Theta(I \cdot N \cdot T^2) \tag{1.5}$$

1.1.3 Αλγόριθμος k-means

Αφού περιγράψαμε τον αλγόριθμο DTW για τον υπολογισμό της απόστασης των χρονοσειρών και της μεθόδου DBA για τον υπολογισμό της μέσης χρονοσειράς ενός συνόλου, είμαστε έτοιμοι να περιγράψουμε τον αλγόριθμο k-means για χρονοσειρές. Ο k-means είναι ίσως ο συνηθέστερος αλγόριθμος διαχωριστικής συσταδοποίησης και γνωστός για την ταχύτητά του. Η εφαρμογή του για τη συσταδοποίηση χρονοσειρών με βάση το σχήμα τους λειτουργεί ως εξής:

1. Αρχικός καθορισμός των k συστάδων και των k εκπροσώπων των clusters.

2. Για κάθε χρονοσειρά βρίσκεται ο πλησιέστερος εκπρόσωπός της σύμφωνα με την απόσταση DTW.

3. Υπολογίζεται ο νέος εκπρόσωπος κάθε συστάδας με τον αλγόριθμο DBA.

4. Ο αλγόριθμος σταματάει είτε μετά από έναν προκαθορισμένο αριθμό επαναλήψεων ή αν το συνολικό άθροισμα των αποστάσεων των χρονοσειρών με τους εκπροσώπους τους γίνει μικρότερο άπο ένα προκαθορισμένο όριο. Διαφορετικά, επιστροφή στο βήμα 2.

Ο k-means έχει υπολογιστική πολυπλοκότητα ίση με $O(I \cdot N \cdot k \cdot T^2)$. Όπως βλέπουμε η χρονική πολυπλοκότητα της απόστασης DTW επηρεάζει σημαντικά την ταχύτητα του k-means.

1.2 Μοντελοποίηση Χρονοσειρών με Gaussian Processes

Ο σκοπός της διπλωματικής αυτής είναι να μειώσουμε την παραπάνω χρονική πολυπλοκότητα της συσταδοποίησης που οφείλεται στο DTW. Στη βιβλιογραφία υπάρχουν διάφορες μέθοδοι για να αντιμετωπίσουν αυτό το πρόβλημα. Η πιο σημαντική κατηγορία μεθόδων είναι οι μέθοδοι αναπαράστασης χρονοσειρών. Κάθε χρονοσειρά αναπαρίσταται σε έναν διαφορετικό χώρο μικρότερης διαστατικότητας και στη συνέχεια οι αναπαραστάσεις αυτές χρησιμοποιούνται για την συσταδοποίηση των χρονοσειρών.

Έχουν προταθεί πολλές διαφορετικές μέθοδοι αναπαράστασης χρονοσειρών που χρησιμοποιούνται για συσταδοποίηση. Σε αυτή την διπλωματική θα χρησιμοποιήσουμε τη μέθοδο Sparse Gaussian Process Regression, η οποία είναι μια προσεγγιστική μέθοδος για Gaussian Process Regression.

1.2.1 Gaussian Process Regression

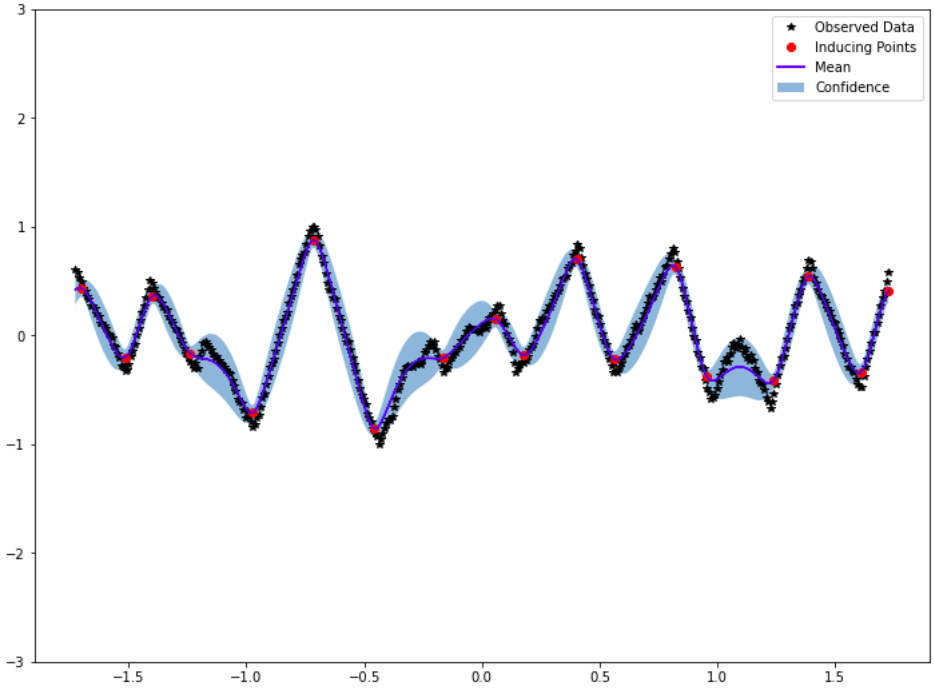
Μια Gaussian Process (GP) αναπαριστά ένα ενδεχομένως άπειρο σύνολο τυχαίων μεταβλητών διατεταγμένο στο χώρο ή στο χρόνο του οποίου κάθε πεπερασμένο υποσύνολο ακολουθεί από κοινού γκαουσιανή κατανομή. Αν θεωρήσουμε μια συνάρτηση ως ένα άπειρο σύνολο σημείων στο χώρο, τότε μια Gaussian Process λέμε ότι είναι μια κατανομή πάνω στο χώρο των συναρτήσεων. Μια GP περιγράφεται μοναδικά από μία συνάρτηση μέσης τιμής $m(x)$ και μία συνάρτηση συνδιακύμανσης (ή kernel) $k(x, x')$. Η επιλογή του kernel καθορίζει την ομαλότητα της συνάρτησης. Οι πιο γνωστές συναρτήσεις kernel είναι το RBF και το Matérn kernel.

Θεωρούμε ότι έχουμε ένα σετ δεδομένων (x_i, y_i) που αποτελείται από N παρατηρήσεις και υποθέτουμε ότι κάθε παρατήρηση y_i προέρχεται από μια συνάρτηση $f(x)$ με την προσθήκη κάποιου γκαουσιανού θορύβου, δηλαδή $y_i = f(x_i) + \epsilon_i$, με $\epsilon_i \sim N(0, \sigma^2)$. Μια GP μπορεί να χρησιμοποιηθεί ως prior κατανομή και να συνδυαστεί με τα δεδομένα μας για να μας δώσει την posterior κατανομή της συνάρτησης. Με την posterior κατανομή μπορούμε να προβλέψουμε την τιμή της συνάρτησης f σε νέα x_*. Το κυριότερο μειονέκτημα της Gaussian Process Regression είναι η κυβική της πολυπλοκότητα. Για την επίλυση του προβλήματος αυτού έχουν αναπτυχθεί γενικά πολλές διαφορετικές προσεγγιστικές μέθοδοι. Για το σκοπό μας στη διπλωματική αυτή θα επικεντρωθούμε σε μία τέτοια προσεγγιστική μέθοδο.

1.2.2 Sparse Gaussian Process Regression

Η Sparse Gaussian Process Regression (SGPR) είναι μια προσεγγιστική μέθοδος για Gaussian Process Regression που προτάθηκε από τον Τίτσια το 2009 [68]. Η SGPR χρησιμοποιεί ένα σετ έξτρα σημείων που ονομάζονται inducing points τα οποία συνοψίζουν το σύνολο των δεδομένων που έχουμε. Αυτά τα inducing points δεν ανήκουν στα δεδομένα, είναι νέα, λιγότερα σε αριθμό, σημεία των οποίων οι τιμές και οι θέσεις πρέπει να υπολογιστούν. Ο στόχος είναι να προσεγγιστεί η posterior κατανομή του πλήρους GP μοντέλου με μια variational κατανομή. Αυτό γίνεται ελαχιστοποιώντας την KL divergence ανάμεσα στις δύο αυτές κατανομές.

Οπότε, η τιμή και η τοποθεσία (time location) των inducing points μαθαίνεται από τη βελτιστοποίηση μιας αντικειμενικής συνάρτησης. Με την posterior κατανομή της μεθόδου, που υπολογίζεται αναλυτικά σε κλειστή μορφή, μπορεί κάποιος να κάνει προβλέψεις για την τιμή της συνάρητης f σε νέες θέσεις του πεδίου ορισμού. Η κατανομή αυτή εξαρτάται μόνο από τα inducing points και όχι από το σύνολο των δεδομένων όπως η posterior του πλήρους GP μοντέλου. Στο σχήμα 1.1 απεικονίζεται η posterior κατανομή του προσεγγιστικού μοντέλου SGPR όταν εφαρμόζεται σε μια χρονοσειρά.



Σχήμα 1.1: Μοντελοποίηση χρονοσειρών με SGPR.

Η υπολογιστική πολυπλοκότητα της προσεγγιστικής αυτής μεθόδου είναι $O(nm^2 + m^3)$, όπου n ο συνολικός αριθμός των δεδομένων μας, m ο αριθμός των inducing points και $m < n$. Ο αριθμός m των inducing points επιλέγεται από τον χρήστη εκ των προτέρων.

Τέλος, ένα θεωρητικό αποτέλεσμα που έδειξε ο Burt [16] και λειτουργεί ως ένδειξη για τη χρήση λίγων inducing points είναι ότι στη περίπτωση που έχουμε δεδομένα που ακολουθούν κανονική κατανομή στο χώρο και το RBF kernel, τότε ασυμπτωματικά αρκούν *logn* inducing points για το μηδενισμό της KL divergence.

1.3 Προτεινόμενη μέθοδος

Σε αυτή τη διπλωματική προτείνουμε μία νέα μέθοδο δύο σταδίων για τη συσταδοποίηση χρονοσειρών με βάση το σχήμα τους. Πρώτα, μοντελοποιούμε τις χρονοσειρές με σύνολα από inducing points εφαρμόζοντας Sparse Gaussian Process Regression σε κάθε χρονοσειρά. Στη συνέχεια, αυτά τα σύνολα από inducing points οργανώνονται σε συστάδες με την εφαρμογή του αλγορίθμου k-means. Για την εφαρμογή του αλγορίθμου k-means χρειαζόμαστε ένα μέτρο απόστασης και μια μέθοδο για τον υπολογισμό του κέντρο ενός συνόλου από σετ inducing points.

Το σύνολο των inducing points μιας χρονοσειράς διατηρεί το σχήμα της και γι΄ αυτό αντιμετωπίζουμε κάθε σετ από inducing points ως μια χρονοσειρά με λιγότερα στοιχεία. Για τη σύγκρισή τους χρησιμοποιούμε ως βάση την απόσταση DTW. Η δυσκολία έγκειται στο γεγονός ότι οι τοποθεσίες (time locations) των διαφορετικών σετ από inducing points είναι διαφορετικές. Γι΄ αυτό το λόγο χρησιμοποιούμε μια δισδιάστατη εκδοχή της DTW με μία παράμετρο α. Το μόνο που αλλάζει σε σχέση με τον κλασικό αλγόριθμο του DTW είναι ότι η συνάρτηση τοπικού κόστους γίνεται:

$$c_{i,j} = (x_i - y_j)^2 + \alpha \cdot (t_{x_i} - y_{y_j})^2 \tag{1.6}$$

Για $\alpha = 0$ έχουμε το κανονικό μονοδιάστατο DTW, ενώ για $\alpha = 1$ έχουμε το δισδιάστατο DTW. Η αύξηση της τιμής της παραμέτρου α έχει ως αποτέλεσμα να αποτρέπεται η αντιστοίχιση πολλών σημείων μιας χρονοσειράς σε ένα και μόνο σημείο της άλλης. Έτσι, από μια μεγάλη τιμή της παραμέτρου α και μετά, η αντιστοίχιση που ορίζει η απόστασή μας παραμένει ίδια.

Για τον υπολογισμό των κέντρων (μέσων) κατά τον αλγόριθμο k-means χρησιμοποιείται ο αλγόριθμος DBA με τη νέα απόσταση χωρίς κάποια άλλη αλλαγή. Συνεπώς, η υπολογιστική πολυπλοκότητα της μεθόδου μας είναι:

$$O(N \cdot T \cdot m^2 + k \cdot I \cdot N \cdot m^2) \tag{1.7}$$

όπου είναι N ο αριθμός των χρονοσειρών, T το μήκος των χρονοσειρών, m ο αριθμός των inducing points, k ο αριθμός των συστάδων και I ο αριθμός των επαλήψεων του k-means.

Στο πλήρες κείμενο προτείνεται και μια δεύτερη πιο πολύπλοκη απόσταση βασισμένη και αυτή στον αλγόριθμο DTW που κάνει χρήση της δυνατότητας που μας δίνει η Sparse Gaussian Process Regression για την πρόβλεψη τιμών της συνάρτησης f. Η απόσταση αυτή παραλείπεται σε αυτή την περίληψη για λόγους συντομίας. Η δεύτερη μέθοδος, σε αντίθεση με την απλούστερη που περιγράψαμε παραπάνω, δεν ελέγχεται πειραματικά λόγω ενός θέματος υλοποίησης.

Η αποτελεσματικότητα και η ταχύτητα της μεθόδου μας με την απλή απόσταση επαληθεύτηκε πειραματικά χρησιμοποιώντας 15 βάσεις χρονοσειρών από το αρχείο UCR [20]. Συγκρίναμε την μέθοδό μας με $m = x \cdot logT$ inducing points, όπου $x = 1$, 2, 3, 4 και 5, και ορισμένες τιμές της παραμέτρου α, με την εφαρμογή του αλγορίθμου k-means με DTW και DBA σε ολόκληρες τις χρονοσειρές. Ο k-means με DTW και DBA θεωρείται στη βιβλιογραφία state-of-the-art μέθοδος όσον αφορά την ποιότητα της συσταδοποίησης. Η μέθοδός μας έχει καλύτερα αποτελέσματα στις περισσότερες βάσεις χρονοσειρών και είναι αρκετά ταχύτερη αφού έχει υπολογιστική πολυπλοκότητα:

$$O(N \cdot T \cdot log^2 T + k \cdot I \cdot N \cdot log^2 T) \tag{1.8}$$

Συμπερασματικά, η μέθοδος που προτείνουμε μείωνει το θόρυβο και τη διαστατικότητα των χρονοσειρών χάρη στη δύναμη των Gaussian Processes. Με αυτό τον τρόπο, επιτυγχάνει σημαντική μείωση στην χρονική πολυπλοκότητα της συσταδοποίησης χρονοσειρών χωρίς να περιορίζει την αποτελεσματικότητά της.

Chapter 2

Introduction

2.1 Background

A time series is a sequence of observations measured successively in time. It is essentially classified as dynamic data because its feature values change as a function of time. The most obvious example for a time series is probably the Dow Jones or the development of certain stock prices. Nowadays, with the increasing power of data storage during the last decades, many real-world applications store and keep data for a long time. As a consequence, time series data is generated in many different fields. It can be found in economics (unemployment rates), social sciences (population), finance, epidemiology (mortality rates), and the physical sciences (pollution levels) [88].

Time series analysis includes a variety of techniques, such as classification, clustering, and forecasting. In this thesis, we consider time series clustering. Clustering is considered the most important unsupervised learning problem. It is a data mining technique where similar objects are placed into related or homogeneous groups without advanced knowledge of the groups' definitions. Time series clustering problems arise when we observe a sample of time series and we want to group them into different categories or clusters. Clustering such complex objects is particularly advantageous because it leads to discovery of interesting patterns in time series datasets. As these patterns can be either frequent or rare, several research challenges have arisen such as: developing methods to recognize dynamic changes in time series, anomaly detection and character recognition. Time series clustering can also be used as a subroutine in other data mining algorithms, such as rule discovery, and indexing. Finding the clusters of time series has applications in many different fields [6]:

- **Financial Markets**: In financial markets, the values of the stocks represent time series which vary with time. The clustering of such time

series can provide insights into the trends in the underlying data.

- **Medical Data**: Different kinds of medical data such as EEG readings are in the form of time series. The clustering of such time series can provide an understanding of the common shapes in the data. These common shapes can be related to different kinds of diseases.

- **Earth Science Applications**: Numerous applications in earth science, such as temperature or pressure, correspond to series, which can be mined in order to determine the frequent trends in the data.

- **Spatio-temporal Data**: Trajectory data can be considered a form of multivariate time series data. The trends in these series can be used in order to determine the important trajectory clusters in the data.

There are three different ways to cluster time series, namely shape-based, feature-based or model-based [74]. In the shape-based method, the shapes of two time series are matched as well as possible. In the feature-based approach, static features from each time series are calculated and the clustered. In model-based methods, a raw time series is transformed into model parameters and then a clustering method is applied to the extracted model parameters. This thesis is concerned with shape-based clustering.

From a different point of view, clustering algorithms can be classified as partitioning, hierarchical, density-based, grid-based or model-based algorithms. Here, we only consider partitioning clustering algorithms because they are relatively scalable and and easy to implement. In partitioning clustering, the goal is to make k groups from n unlabelled objects in the way that each group contains at least one object. The most popular partitioning clustering algorithm is the well-known k-means. This thesis focuses on partitioning shape-based clustering.

2.2 Motivation

Time series clustering is a challenging issue for data miners. First of all, time series databases are often very large databases. They require a large amount of memory that dramatically slows down the process of clustering. Another major challenge is that time series data is a type of temporal data which is naturally high dimensional [7]. Handling such data to perform clustering is difficult and leads to complicated clustering methods. Finally, to make the clusters, similar time series should be found. An appropriate similarity measure between time series is a key issue for any clustering process.

However, such a process is complicated, because time series are inherently noisy and include outliers and shifts, at the other hand the length of time series varies and the distance among them needs to be calculated. These issues have made the choice of the distance measure a major challenge for researchers.

A suitable similarity measure must provide a concrete way of evaluating the distance between any two points. For shape-based clustering, the shape of time series is the key factor in identifying pattern similarity. Some transformations, such as offset shifting, translation in time and time scaling will not change the shape of a time series. We expect to find a dissimilarity measure that is invariant to these transformations. Simply using Euclidean distance is not a good choice as it can be easily affected by shifting. The distance that outperforms every other in this context is the common Dynamic Time Warping (DTW) [81]. DTW is a robust measure for time series and takes into consideration the alignment along a time axis. Intuitively, the sequences are warped in a non-linear fashion to match each other. Unfortunately, DTW has a quadratic complexity that significantly reduces clustering speed.

Partitioning methods are relatively scalable and fast. The most typical and widely used partitioning method is k-means, and many other algorithms incorporate its basic ideas. This method consists of two steps: determine the "closest" center for each object, and then update these centers. The algorithm iteratively runs the above steps until convergence. First, it requires an appropriate similarity measure between two time series. The common distance measure for shape-based clustering is Dynamic Time Warping. To find a method that calculates a center (average) of a set of time series under DTW is a difficult problem, which can be seen as a multiple alignment problem, and several heuristics have been proposed in the literature. The state-of-the-art method for center calculation is DTW Barycenter Averaging (DBA) [52]. DBA is a global averaging method which refines an initially average time series, in order to minimize its squared distance (DTW) to the set of time series. K-means algorithm with DTW as a distance measure and DBA as an averaging method has a time complexity of $O(I \cdot N \cdot k \cdot T^2)$, where I is the number of iterations, N is the number of time series, k is the number of clusters and T is the length of the series. The problem here is the quadratic complexity of DTW ($O(T^2)$)

Considering all these difficulties in the clustering of time series, dimensionality reduction is the common solution to increase the performance and speed of the clustering process. Dimensionality reduction is a preprocessing step considered to be a fundamental and important process in time series data mining. It represents the raw time series in another space by transforming them to a lower dimensional space or by feature extraction. The learned

representations are then clustered instead of the whole time series data. The goal here is to represent the data without slowing down the execution time and without a significant data loss.

There is a lot of work in the literature considering representation methods for time series. For example, Adaptive Piecewise Constant Approximation (APCA) [38] transforms each time series by a set of constant value segments of varying lengths such that their individual reconstruction errors are minimal. Another approach is the splines. Iorio et al. [34] proposed to model the time series by using P-spline smoothers and then to cluster the functional objects as summarized by the optimal spline coefficients using k-means algorithm and Dynamic Time Warping.

2.3 Thesis contribution

In this thesis, we propose a novel two-stage framework for clustering time series based on their shapes. First, we use Sparse Gaussian Process Regression (SGPR) as a representation method to model the series by a (much) lower number of data points. We then apply k-means algorithm on these learned points with a modified version of Dynamic Time Warping (DTW) as the distance measure.

We suppose each time series is a noisy realization of a functional form. To find the function that better fits the time series we perform Gaussian Process Regression. Gaussian Process (GP) is a collection of random variables indexed by time, such that every finite collection of those random variables has a multivariate normal distribution. It is completely characterized by a mean and a kernel function. GPs are powerful tools and are widely used for regression tasks. The main drawback of Gaussian Process Regression (GPR) is its cubic complexity.

They have been proposed many methods to overcome this computational issue. Here, we focus on a method, proposed by Titsias [68], called Sparse Gaussian Process Regression (SGPR). SGPR learns m extra data points (inducing points) that summarize the time series data. These m inducing points are fewer than the series data points and are used to perform the regression task. The method has a time complexity of $O(T \cdot m^2)$.

We model each of the N time series using Sparse Gaussian Process Regression and we get N sets of m inducing points. We then perform clustering with k-means using these representations. We propose two different distance measures, based on DTW, and their corresponding averaging methods, based on DBA, to handle the sets of inducing points. The framework has a total time complexity of $O(N \cdot T \cdot m^2 + k \cdot I \cdot N \cdot m^2)$.

Our framework can handle noise and outliers and because of its adaptive nature can reduce the dimensionality of the time series significantly. It improves the time and space complexity of shape-based clustering without losing quality. Due to implementation reasons we only tested our framework with one of two proposed distances. The experimental study shows that our framework using $m = x \cdot logT$ (where $x = 1, 2, 3, 4, 5$) inducing points outperforms in most datasets the classical k-means algorithm with DTW on the whole time series data, which is a state-of-the-art method.

2.4 Thesis structure

The remaining of the thesis is structured as follows. In Chapter 3 we review similarity measures for time series, and emphasis is given to the description of Dynamic Time Warping. In Chapter 4 we present existing methods for averaging a set of time series data. Within Chapter 5, we describe some representation methods for time series and present two well-known clustering algorithms. In the next chapter (Chapter 6), we introduce Gaussian Processes and Sparse Gaussian Process Regression. In Chapter 7 we review some previous work and propose our framework for time series clustering. Finally, in Chapter 8 we present the experimental results of our framework and in Chapter 9 we make our concluding remarks.

Chapter 3

Distance measures

Time series data is a collection of observations obtained through repeated measurements over time. Time series data is everywhere, since time is a constituent of everything that is observable. As our world gets increasingly instrumented, sensors and systems are constantly emitting a relentless stream of time series data.

In the following sections we give a formal definition of time series data (Section 3.1) and we introduce shaped-based measures which compare the overall shape of the time-series based on its actual values. Shaped-based measures can be divided into two subgroups: lock-step measures (Section 3.2) and elastic measures (Section 3.3).

3.1 Time series data

A time series is an ordered sequence of observations at successive time points. It is a common type of dynamic data that naturally arises in many different scenarios such as statistics, signal processing, pattern recognition, astronomy, finance, and largely in any domain of applied science and engineering which involves temporal measurements. Time series pose some challenging issues due to their large size and high dimensionality [7]. In this context, dimensionality of a series is related to time, and it can be understood as the length of the series. Additionally, a single time series may consist several values that change on the same time scale (multivariate time series). Adopting the definition of time series stated by Esling and Agon [23], we have the following:

Definition 3.1.1 (Time series)**.** A time series X of length n is a sequence of pairs

$$X = [(x_1, t_1), (x_2, t_2), ..., (x_n, t_n)] \quad || \quad (t_1 < t_2 < ... < t_i < ... < t_n) \quad (3.1)$$

where each x_i is a data point in a d-dimensional space and each t_i represents the point in time when x_i was measured. If the relevant time-series share the same sampling rates, the time stamps can be omitted and the time series can be regarded as an ordered sequence of d-dimensional data points.

Time series data are analyzed using a variety of statistical techniques, such as classification, clustering, and anomaly detection. This thesis focuses on time series clustering. Clustering is a well-known unsupervised machine learning method for dividing observations into groups (called clusters) such that observations within the same cluster tend to be more similar than those in different clusters [91].

To determine whether time series data are similar, one must first decide on a measure to quantify this similarity. The choice of distance is fundamental and it is particularly important in the presence of dynamic data, such as time series. Many different distance measures have been proposed in the literature. Here we focus on shape-based measures.

3.2 Lock-step measures

In this section, we introduce the definition of Minkowski distance, a lock-step measure. As with all lock-step measures, this distance measure require both time series to be of equal length (n=m) and compare time point i of time series x with the same time point i of time series y. Note that we examine the lock-step distance measures from a time series perspective, but that these measures can also be used for non-time series clustering assignments. The only requirement is that all observations are numerical vectors of equal length.

3.2.1 Minkowski distance

The Minkowski distance is defined by

$$d_{min}(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \tag{3.2}$$

This is the L_p-norm of the difference between two equal length vectors. It is the generalization of the commonly used Euclidean distance ($p = 2$), Manhattan distance ($p = 1$) and Chebyshev distance ($p = \infty$). The formulas for those distance measures can be found in Equations (3.3) and (3.4). For

clustering, usually only the Euclidean distance and Manhattan distance are considered.

$$\textbf{Euclidean distance}: d_{euc}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.3}$$

$$\textbf{Manhattan distance}: d_{man}(x, y) = \sum_{i=1}^{n}|x_i - y_i| \tag{3.4}$$

Minkowski distance is a metric distance function, since it obeys to the three fundamentals metric properties: non-negativity, symmetry and triangle inequality. It is very intuitive, free of parameters, and takes linear time, meaning that the time complexity for all p is $O(n)$. On the other hand, it has some limitations, such as: high sensitivity to small distortions in the time axis [40], noise and outliers [56] because fixed pairs of data are compared.

3.3 Elastic measures

For most applications, simple distance measures such as L_p-norms are sufficient and they also provide a low time complexity [73]. Nevertheless there are cases when the overall shape of two time series is similar, but one of them is accelerated or decelerated. Let's take the example of recorded speech. The same word spoken by two different speakers will produce time series similar in shape, but deformed by the speakers pace and intonation. In order to find the similarity and thus to achieve a better alignment, we have to warp the time axis. This is illustrated by Figure 3.1. The upper part of the image shows a simple alignment of two similar time series using Euclidean distance, which will produce a rather high dissimilarity value due to its sensitivity to irregularities in the time axis. This issue is addressed by elastic distance measures, such as Dynamic Time Warping (DTW).

Elastic distance measures are designed to work with time-series data. They create a non-linear mapping to align the series and allow comparison of one-to-many points. This makes it possible for them to warp in time and be more robust when it comes to, for example, handling outliers.

In this section, we introduce the most commonly used elastic distance measure, Dynamic Time Warping (DTW). The experimentation in [23], [92], [73] has shown that, on average, DTW is the best available distance measure for time series.

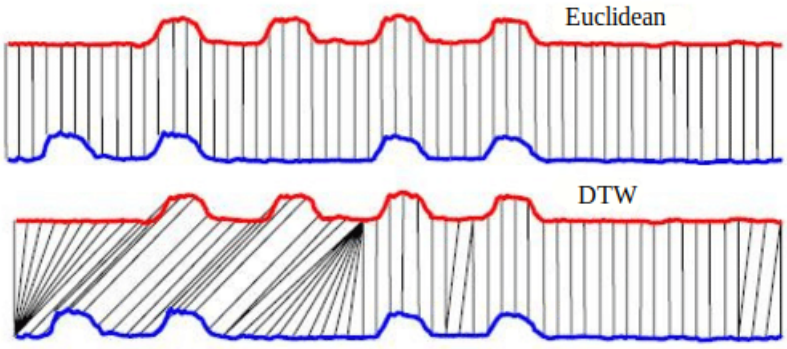


Figure 3.1: Euclidean vs DTW. Source: Employing Subsequence Matching in Audio Data Processing (Fig. 6 in [71]).

3.3.1 Dynamic Time Warping (DTW)

The Dynamic Time Warping (DTW) is one of well-known elastic distance measures between two given (time-dependent) sequences that finds an optimal alignment between them under certain restrictions. Intuitively, the sequences are warped in a non-linear fashion to match each other. While DTW originally has been used to compare different speech patterns in automatic speech recognition [57], it is currently used in many areas, such as data mining and time series clustering [48], computer vision and computer animation [1], protein sequence alignment and chemical engineering [70] and music and signal processing [2].

In this subsection, we first introduce the main ideas of classical DTW (Section 3.3.1.1) and then we summarize several modifications concerning local (Section 3.3.1.2) and global parameters (Section 3.3.1.3).

3.3.1.1 Classical DTW

Given two time series $X = (x_1, x_2, ..., x_n), n \in \mathbb{N}$ and $Y = (y_1, y_2, ..., y_m), m \in \mathbb{N}$ represented by the sequences of values DTW yields optimal solution in $O(n \cdot m)$. The only restriction placed on the data sequences is that they should be sampled at equidistant points in time (this problem can be resolved by resampling).

In the following, we fix a feature space denoted by Φ. Then $x_i, y_j \in \Phi$ for $i \in [1 : n]$ and $j \in [1 : m]$. To compare two different features $X, Y \in \Phi$, one needs a local distance measure which is defined to be a function:

$$d : \Phi \times \Phi \to \mathbb{R} \geq 0 \tag{3.5}$$

Intuitively d has a small value (low cost) when sequences are similar

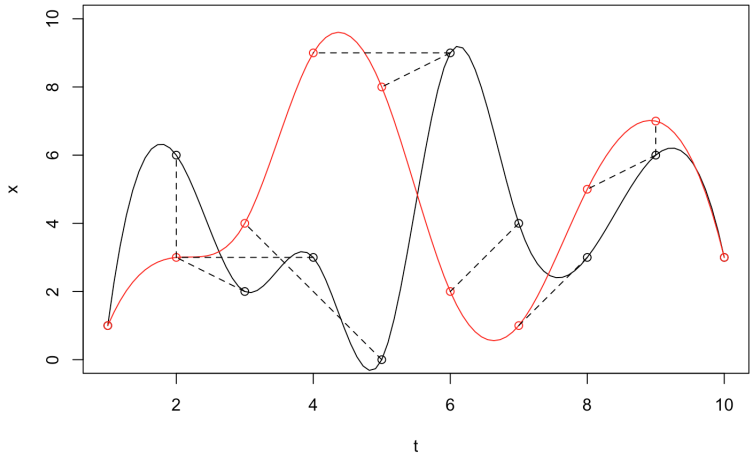


Figure 3.2: Raw time series, dotted lines show the desirable alignment. Source: Dynamic time warping for predictive modeling using sensor data (Fig. 1 in [21]).

and large value (high cost) when they are different. Since the Dynamic Programming algorithm lies in the core of DTW it is common to call this function, the cost function. The task of optimal alignment of the sequences then becomes the task of arranging all sequence points by minimizing the overall cost.

Evaluating the cost measure for each pair of elements of the sequences X and Y, one obtains the distance matrix $C \in \mathbb{R}^{n \times m}$. Algorithm starts by computing this matrix, which is also called local cost matrix. The local cost matrix for the alignment of two sequences X and Y is:

$$C \in \mathbb{R}^{n \times m} : \quad c_{i,j} = (x_i - y_j)^2, \quad i \in [1:n], j \in [1:m] \tag{3.6}$$

After computing the local cost matrix, the goal is to find an alignment between X and Y having minimal overall cost. Intuitively, such an optimal alignment runs along a "valley" of low cost within the cost matrix C, Figure 3.3. This alignment path (or warping path) defines the correspondence of an element $x_i \in X$ to $y_j \in Y$ following three conditions. The next definition formalizes the notion of a warping path.

Definition 3.3.1 (Warping path)**.** An (n, m)-warping path (or simply referred to as warping path) is a sequence $p = (p_1, ..., p_L)$ with $p_l = (n_l, m_l) \in [1:n] \times [1:m]$ for $l \in [1:L]$ satisfying the following three conditions:

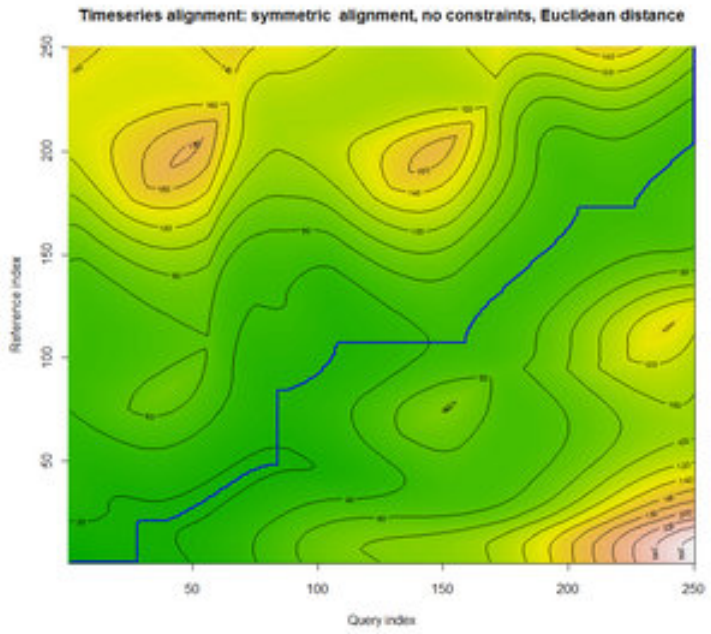


Figure 3.3: Time series alignment, cost matrix heatmap. Source: Dynamic Time Warping Algorithm Review (Fig. 2 in [61]).

1. **Boundary condition:** $p_1 = (1, 1)$ and $p_L = (n, m)$.
2. **Monotonicity condition:** $n_1 \leq n_2 \leq ... \leq n_L$ and $m_1 \leq m_2 \leq ... \leq m_L$.
3. **Step size condition:** $p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1)\}$ for $l \in [1 : L-1]$.

Note that the step size condition (3) implies the monotonicity condition (2), which nevertheless has been quoted explicitly for the sake of clarity. A warping path $p = (p_1, ..., p_L)$ defines an alignment between two sequences $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_m)$ by assigning the element x_{n_l} of X to the element y_{m_l} of Y. The boundary condition (1) states that the starting and ending points of the warping path must be the first and the last points of the aligned sequences. In other words, the alignment refers to the entire sequences X and Y. The monotonicity constraint preserves the time-ordering of points. Finally, the step size condition limits the warping path from shifts in time while aligning sequences.

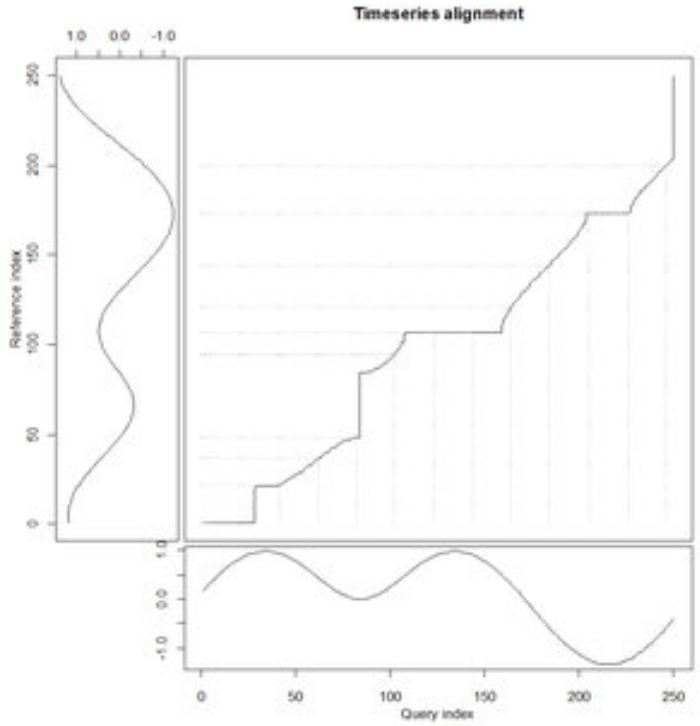


Figure 3.4: The optimal warping path aligning time series from the Figure 3.3. Source: Dynamic Time Warping Algorithm Review (Fig. 3 in [61]).

The cost associated with a warping path will be:

$$c_p(X,Y) = \sqrt{\sum_{l=1}^{L} c(x_{n_l}, y_{m_l})} \tag{3.7}$$

The warping path which has a minimal (optimal) cost called the optimal warping path. We will refer to this path as P*.

In order to find such an optimal path, we need to test every possible warping path between X and Y which could be computationally challenging. To overcome this challenge, DTW employs a Dynamic Programming-based algorithm with complexity $O(n \cdot m)$.

The DTW distance $DTW(X,Y)$ between X and Y is then defined as the total cost of P*:

$$DTW(X,Y) = c_{p*}(X,Y) = min\{c_p(X,Y), p \in P^{n \times m}\} \tag{3.8}$$

The $P^{n \times m}$ is the set of all possible warping paths and builds the accumu-

lated cost matrix or global cost matrix D which is defined as follows:

1. First row: $D(1,j) = \sum_{l=1}^{j} c(x_1, y_l), j \in [1,m]$.
2. Second row: $D(i,1) = \sum_{l=1}^{i} c(x_l, y_1), i \in [1,n]$.
3. All other elements: $D(i,j) = min\{D(i-1,j-1), D(i,j-1), D(i-1,j-1)\}$
   $+ c(x_i, y_j), i \in [1,n], j \in [1,m]$.

Algorithm 1 builds the accumulated cost matrix, where X and Y are the input time series and C is the local cost matrix representing all the pairwise distances between X and Y.

Algorithm 1 AccumulatedCostMatrix(X, Y, C)

```
1: n ← |X|
2: m ← |Y|
3: dtw[] ← new[n × m]
4: dtw(0,0) ← 0
5: for i = 1; i ≤ n; i++ do
6:     dtw(i,1) ← dtw(i−1,1) + c(i,1)
7: for j = 1; j ≤ m; j++ do
8:     dtw(1,j) ← dtw(1,j−11) + c(1,j)
9: for i = 1; i ≤ n; i++ do
10:    for j = 1; j ≤ m; j++ do
11:        dtw(i,j) ← c(i,j)+min{dtw(i−1,j); dtw(i,j−1); dtw(i−1,j−1)}
12: return dtw
```

Once we built the accumulated cost matrix, the warping path could be found by simple backtracking from the point $p_{end} = (m,n)$ to the $p_{start} = (1,1)$ following the greedy strategy described by Algorithm 2.

We continue with some remarks about Dynamic Time Warping. First, it is easy to see that the DTW distance is symmetric when the local cost measure c is symmetric. Second, the DTW distance is in general not positive definite even if this holds for c. For example, one obtains $DTW(X,Y) = 0$ for the sequences $X = (x_1, x_2)$ and $Y = (x_1, x_1, x_2, x_2, x_2)$ in case $c(x_1, x_1) = c(x_2, x_2) = 0$. Furthermore, the DTW distance generally does not satisfy the triangle inequality even when c is a metric. This fact is illustrated by the following example.

Consider the three sequences $x = [0,1,1,2]$, $y = [0,1,2]$ and $z = [0,2,2]$. Using the distance $c(x,y) = |x-y|$ as a metric, we have that $DTW(x,z) = 2$,

Algorithm 2 OptimalWarpingPath(dtw)

```
1: path[] ← new  array
2: i = rows(dtw)
3: j = columns(dtw)
4: while (i > 1)&(j > 1) do
5:     if i = 1 then
6:         j ← j − 1
7:     else if j = 1 then
8:         i ← i − 1
9:     else
10:        if dtw(i − 1, j) = min{dtw(i − 1, j); dtw(i, j − 1);
11:        dtw(i − 1, j − 1)} then
12:            i ← i − 1
13:        else if dtw(i, j − 1) = min{dtw(i − 1, j); dtw(i, j − 1);
14:        dtw(i − 1, j − 1)} then
15:            j ← j − 1
16:        else
17:            i ← i − 1;   j ← j − 1
18:        path.add((i, j))
19: return path
```

$DTW(x, y) = 0$ and $DTW(y, z) = 1$. Therefore the triangle inequality does not hold for DTW, since it is not true that: $DTW(x, z) \leq DTW(x, y) + DTW(y, z)$.

The time complexity of the DTW algorithm is $O(n \cdot m)$ [81]. Assuming that $n \geq m$, the time complexity is $O(n^2)$. The 50 years old quadratic time bound was broken in 2018, an implementation due to Gold and Sharir that enables computing DTW in $O(n^2/loglog(n))$ time and space [27]. The natural implementation of DTW also has $O(n \cdot m)$ space complexity. This bound was recently broken using a divide-and-conquer algorithm by Tralie and Dempsey, yielding a linear space complexity of $O(n + m)$. This algorithm has an added advantage of being amenable to parallel computation [69].

Various modifications have been proposed in order to speed up DTW computations as well as to better control the possible routes of the warping paths. In the next sections, we will discuss some of these variations.

3.3.1.2 Slope Constraint

The step size condition of a warping path represents a kind of local continuity condition, which ensures that each element of X is assigned to an element of Y and vice versa. However, one drawback of this condition is that sometimes it tends to create an unrealistic correspondence between time series by assigning a single element of one sequence to many consecutive elements of the other sequence. This leads to long vertical and horizontal segments in the warping path.

In order to avoid such phenomena, one can modify the step size condition to locally prevent horizontal and vertical displacements. For that, some sideways displacements are artificially added to make the path as diagonal as possible [46]. A first solution, presented in Figure 3.5, consists in considering three local displacements. This process can be extended to a larger number of possible displacements around the diagonal.

Let as call K_p the number of elementary steps of the longest local displacement. The recursion of the resulting accumulated cost matrix D for $K_p = 2$ is given by:

$$D_{i,j} = min \begin{cases} D_{i-1,j-2} + d_{i,j-1} + d_{i,j} \\ D_{i-1,j-1} + d_{i,j} \\ D_{i-2,j-1} + d_{i-1,j} + d_{i,j} \end{cases} \tag{3.9}$$

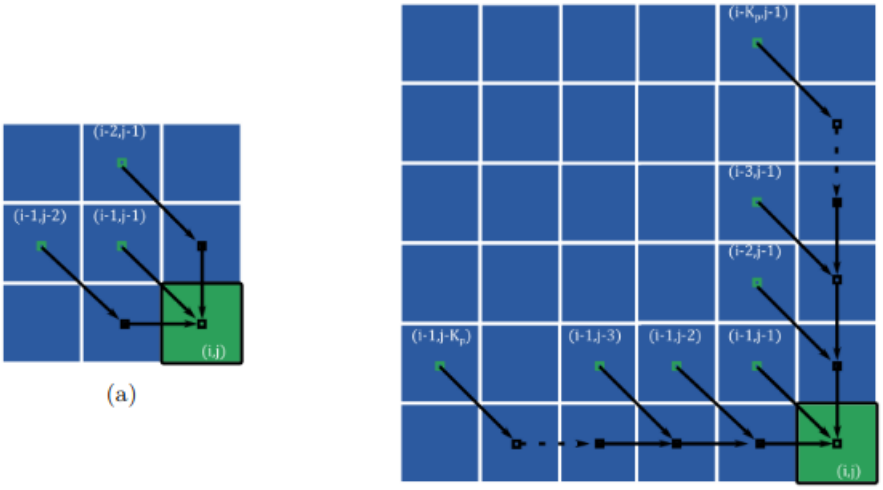


Figure 3.5: The figure illustrates the allowed values used to compute the new $D_{i,j}$. (a) With a constraint slope $K_p = 2$, (b) the general case of local constrained admissible steps with any K_p. Source: Time-series averaging using constrained dynamic time warping with tolerance (Fig. 6 in [45]).

For the general case, it is given by:

$$D_{i,j} = min \begin{cases} D_{i-1,j-K_p} + \sum_{k=1}^{K_p} d_{i,j-(K_p-k)} \\ D_{i-1,j-(K_p-1)} + \sum_{k=1}^{K_p-1} d_{i,j-(K_p-1-k)} \\ . \\ . \\ . \\ D_{i-1,j-2} + d_{i,j-1} + d_{i,j} \\ D_{i-1,j-1} + d_{i,j} \\ D_{i-2,j-1} + d_{i-1,j} + d_{i,j} \\ . \\ . \\ . \\ D_{i-(K_p-1),j-1} + \sum_{k=1}^{K_p-1} d_{i-(K_p-1-k),j} \\ D_{i-K_p,j-1} + \sum_{k=1}^{K_p} d_{i-(K_p-k),j} \end{cases} \tag{3.10}$$

3.3.1.3 Global path constraints

Another common DTW variant is to impose global constraint conditions on the admissible warping paths. Such constraints improve the computational cost and optimize the DTW sensitivity similarly to the step function constraints. More precisely, let $R \subseteq [1:n] \times [1:m]$ be a subset referred to as global constraint region. Then a warping path with respect to R is a warping path that entirely runs within the region R. The subset of matrix that the warping path is allowed to visit is called a warping window or a band.

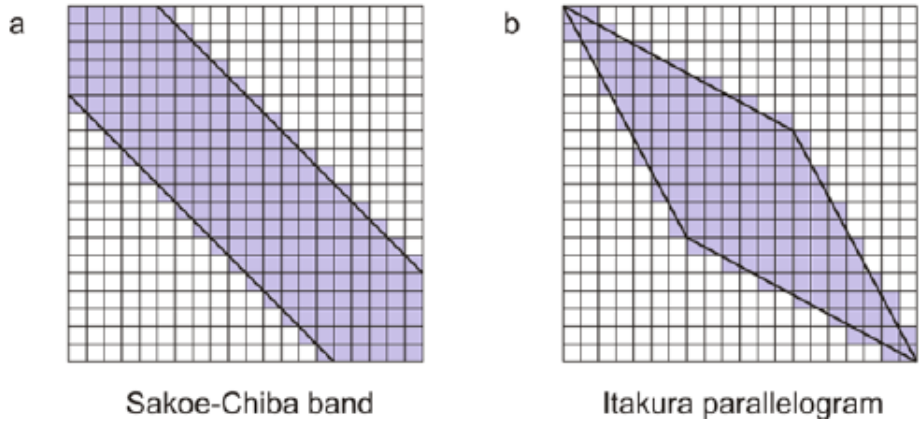


Figure 3.6: Examples of global constraints: (a) Sakoe-Chiba band, (b) Itakura parallelogram. Source: Similarity Measures and Dimensionality Reduction Techniques for Time Series Data Mining (Fig. 8 in [18]).

Two well-known global constraints are the Sakoe-Chiba band [57] and the Itakura parallelogram [35], Figure 3.6. The Sakoe-Chiba region is defined through a window size parameter which determines the largest temporal shift allowed from the diagonal in the direction of the longest time series. On the other hand, the Itakura constraint region is a parallelogram. Contrary to the Sakoe-Chiba band, whose width is constant, the Itakura parallelogram has a varying width, allowing for larger time shifts in the middle than at the first and last time points. It is defined through a max slope parameter which determines the slope of the steeper side.

Chapter 4

Time series averaging

As mentioned in the previous chapter, the traditional Euclidean distance metric is not, in general, an accurate similarity measure for time series. Dynamic Time Warping (DTW) has been proposed and is widely recognized as a relevant measure in various time series applications. Given this similarity measure, many distance-based algorithms can be used in the time series setting. However, many of them, like the well-known K-means algorithm, also require an averaging method and highly depend on its quality.

Unfortunately, estimating the centroid of a set of time series under time warp is not a trivial problem. In this chapter, we present initially the consensus sequence problem (Section 4.1) and then we review the multiple temporal alignment problem (Section 4.2) presenting some progressive and iterative centroid estimation approaches under time warp.

4.1 Consensus sequence

As we focus on DTW, we will only detail the consensus sequence problem from that side. In the context of sequences, the term consensus is used with two meanings: (i) the medoid sequence (Section 4.1.1) and (ii) the average sequence of the set of time series (Section 4.1.2).

4.1.1 Medoid sequence

The goal is to find a sequence in the center of a set of time series. The commonly accepted definition of a center is the object minimizing the sum of (squared) distances to time series within the same set. This sum is also known as Within Group Sum of Squares (WGSS) or inertia. When the center must be found in the dataset, it is called the medoid sequence. In the context

of time series clustering, algorithms like K-medoids are using the medoid of a set of sequences which in some cases can handle noises and outliers better than K-means.

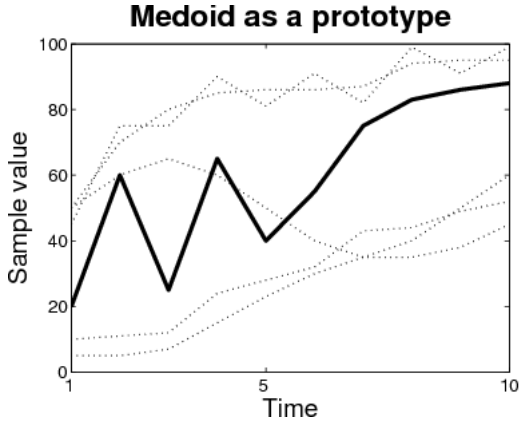


Figure 4.1: Medoid as a prototype. Source: Time-series Clustering by Approximate Prototypes (Fig. 1 in [32]).

4.1.2 Average sequence

When the search space of the center is not restricted, we use the term average sequence. We introduce the definition of an average sequence when the corresponding similarity measure is Dynamic Time Warping.

Definition 4.1.1 (Average sequence)**.** Let $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ be a set of N time series $\mathbf{x}_i = (x_{i1}, ...x_{iT}), i \in \{1, ..., N\}$ and $\mathbb{S}$ the space of all time series sequences. $\forall \mathbf{s} \in \mathbb{S}$, the average sequence $\mathbf{c}$ should satisfy the following:

$$\sum_{i=1}^{N} DTW(\mathbf{x}_i, \mathbf{c}) \leq \sum_{i=1}^{N} DTW(\mathbf{x}_i, \mathbf{s}) \tag{4.1}$$

Since there is no information on the length of the average sequence $\mathbf{c}$, the search cannot be limited to sequences of a given length, so all possible length values for averages have to be considered. In the following, we will review some methods used to determine the multiple temporal alignment of a set of time series.

4.2 Multiple temporal alignments

Temporal warping alignment of time series is a tricky problem [3] and has been an active research topic in many scientific disciplines such as speech

recognition [17] and bio-informatics [3]. It can be shown that a multiple alignment provides an average sequence and conversely [51]. In order to estimate the centroid of two time series under DTW, one standard way is to embed the time series into a new Euclidean space defined by their temporal warping alignment. Then the centroid can be estimated as the average of the linked elements. The problem becomes more complex when the number of time series is more than two, as one needs to determine a multiple alignment that links simultaneously all the time series on their commonly shared elements.

In this subsection, we first give an exact solution to the problem (Section 4.2.1) and then we review some progressive (Section 4.2.2) and iterative centroid estimation approaches (Section 4.2.3).

4.2.1 Exact solution

An exact solution to the multiple alignment problem can be given by extending DTW for aligning N sequences [52]. For example, instead of computing DTW by comparing three values in a square, one have to compare seven values in a cube for aligning three sequences. This can be generalized by computing DTW in a N-dimensional hypercube. Thus, C can be found by averaging column by column the multiple alignment. However this method presents two major difficulties that prevent its use. First, the multiple alignment process has a time complexity of $\Theta(T^N)$, with T being the time series length. Second, the global length of the multiple alignment can be on the order of T^N, requiring unrealistic amounts of memory [72].

Unfortunately, many years of well-motivated research have not provided any exact scalable algorithm, neither for the consensus sequence problem, nor for the multiple alignment problem. In the following, we review several methods for estimating the centroid of aligning more than two time series.

4.2.2 Progressive approaches

The progressive approaches estimate the global centroid by combining pairwise time series centroids through different strategies. Here, we present some of the most important ones.

4.2.2.1 NonLinear Alignment and Averaging Filters (NLAAF)

NonLinear Alignment and Averaging Filters (NLAAF) [30] is a time series averaging strategy that uses a simple pairwise method where each coordinate of the average sequence is calculated as the center of the mapping produced by DTW. Initially, we randomly select $(N/2)$ pairs of time series and we

then apply the method to each pair. That way, $(N/2)$ averaged sequences are created. Then those $(N/2)$ sequences, in turn, are pairwise averaged into $(N/4)$ sequences, and so on, until one sequence is left. As illustrated in Figure 4.2, the averaging method (between two sequences) is applied $(N-1)$ times.

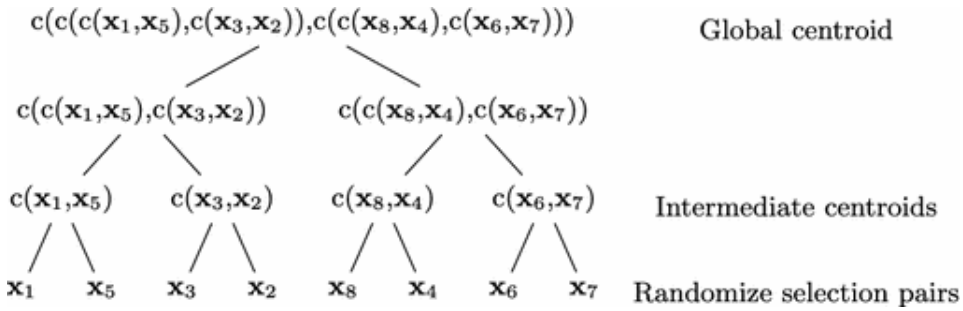


Figure 4.2: Centroid estimation by random pairwise centroid combination. Source: A Comparison of Progressive and Iterative Centroid Estimation Approaches Under Time Warp (Fig. 1 in [64]).

The main drawback of the strategy is the growth of its resulting mean [52]. Each use of the averaging method can almost double the length of the average sequence. As a result, the NLAAF procedure could produce a global average sequence up to $N \times T$ in length. Consequently, NLAAF is generally used with a process that reduces the length of the average, which unfortunately leads to loss of information and poor results. Furthermore, the average depends on the selection of time series as different choices lead to different results.

NLAAF produces an average of two time series computing DTW between these two series, which has a time complexity of $\Theta(T^2)$. Then, to compute the temporary average series, it requires $\Theta(T)$ operations. To reduce the length of the average, Uniform Scaling is commonly used which has a time complexity of $\Theta(T + 2T + 3T + ... + T^2) = \Theta(T^3)$, and thus the complexity of averaging two series is $\Theta(T + T^2 + T^3) = \Theta(T^3)$. Consequently, NLAAF has time complexity [52]:

$$\Theta((N-1) \cdot (T + T^2 + T^3)) = \Theta(N \cdot T^3) \tag{4.2}$$

4.2.2.2 Prioritized Shape Averaging (PSA)

Prioritized Shape Averaging was introduced in [47] to avoid the bias induced by the random selection of pairs during the NLAAF procedure. PSA is a framework that uses the Ascendant hierarchical scheme. The pairwise time

series averaging is guided by the dendrogram obtained through hierarchical clustering, meaning that the most similar time series are averaged first.

Although this strategy removes the bias, the length of the average sequence remains a problem. In case one alignment between two sequences leads to two connected components, the overall resulting mean will have only two coordinates. To solve this problem the authors proposed to replicate each coordinate of the average sequence as many times as there were associations in the corresponding connected component. Unfortunately, this repetition causes the same problem observed with NLAAF.

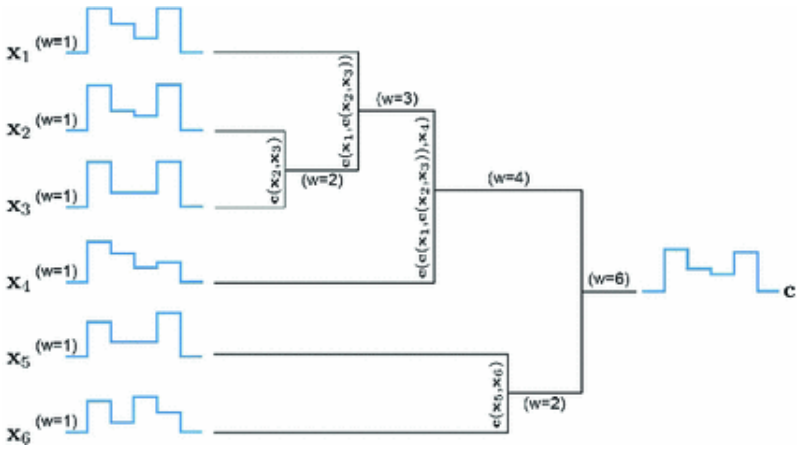


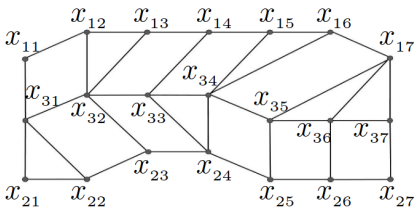
Figure 4.3: Example of six time series sequence averaging using PSA.
Source: A Comparison of Progressive and Iterative Centroid Estimation Approaches Under Time Warp (Fig. 2 in [64]).

PSA computes an average of two time series following the same procedure as NLAAF. Additionally, as PSA is using a hierarchical strategy to order time series, it has at least to compute a dissimilarity matrix, which requires $\Theta(N^2 \cdot T^2)$ operations. The overall PSA averaging of a set of N series then requires [52]:

$$\Theta((N-1) \cdot (T + T^2 + T^3) + N^2 \cdot T^2) = \Theta(N \cdot T^3 + N^2 \cdot T^2) \tag{4.3}$$

4.2.2.3 Cross-Word Reference Template (CWRT)

Abdulla et al. [4] proposed another way to estimate the centroid, where DTW between each time series and a reference one, usually the time series medoid, is first performed. The global estimated centroid is then computed by averaging the time-aligned time series across each point. This approach is called Cross-Word Reference Template (CWRT).



	1	2	...	7
$\mathbf{x}_1$	x_{11}	x_{12}, x_{13}	...	x_{17}
$\mathbf{x}_2$	x_{21}, x_{22}	x_{23}	...	x_{27}
$\mathbf{x}_3$	x_{31}	x_{32}	...	x_{37}
$\mathbf{c}$	$\text{avg}(x_{11}, x_{21}, x_{22}, x_{31})$	$\text{avg}(x_{12}, x_{13}, x_{23}, x_{32})$	...	$\text{avg}(x_{17}, x_{27}, x_{37})$

Figure 4.4: Centroid estimation based on a reference time series. Source: A Comparison of Progressive and Iterative Centroid Estimation Approaches Under Time Warp (Fig. 3 in [64]).

The resulting centroid has the same length as the reference time series and the advantage that does not depend on the order in which time series are processed. Nevertheless, the method still remains a heuristic approach.

4.2.3 Iterative approaches

Iterative approaches work similarly to the progressive ones, but they are able to reduce the error propagation by repeatedly refining the centroid and realigning it to the initial time series, until its stabilization. Here we introduce the most well-known iterative heuristic, DTW Barycenter Averaging (DBA) [52].

4.2.3.1 DTW Barycenter Averaging (DBA)

DBA is a global averaging method which refines an initially (potentially arbitrary) average time series, in order to minimize its squared distance (DTW) to the set of time series.

Let us provide a description of the mechanism of DTW Barycenter Averaging. The goal is to minimize the sum of squared DTW distances from the average time series to the set of time series (WGSS). This sum is created by single distances between each coordinate of time series and coordinates of time series associated with it. Therefore, the contribution of one coordinate of the average series to the total WGSS is actually a sum of euclidean distances between this coordinate and coordinates of time series associated with it during the computation of DTW. We should note that a single coordinate of one of the time series may contribute to the new position of several coordinates of the average. As a result, any coordinate of the average series is updated with contributions from one or more coordinates of each series. To minimize this partial sum for each coordinate of the average, we take the barycenter of this set of coordinates. DBA computes each coordinate of the average series

as the barycenter of its associated coordinates of the set of series. Thus, each coordinate will minimize its part of the total WGSS in order to minimize the total WGSS. The updated average is formed once all barycenters are computed.

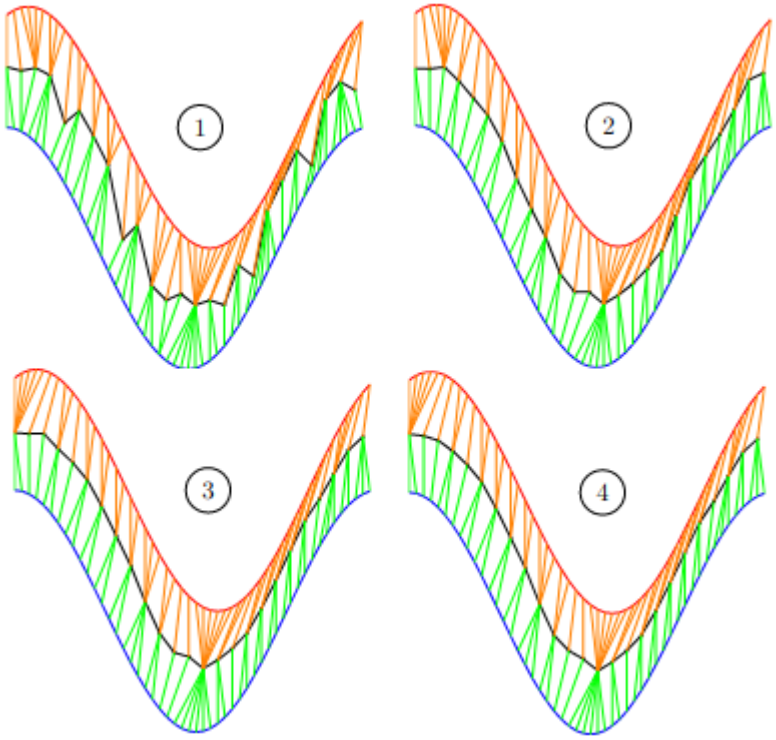


Figure 4.5: DBA iteratively adjusting the average of two time series.
Source: A global averaging method for dynamic time warping, with applications to clustering (Fig. 2 in [52]).

The barycenter is defined as:

$$barycenter\{X_1, ..., X_k\} = \frac{X_1 + ... + X_k}{k} \tag{4.4}$$

Technically, for each refinement iteration, DBA runs in two steps:

- Computing DTW between each individual series and the temporary average, in order to find associations between coordinates of the average series and coordinates of the set of time series.
- Updating each coordinate of the average series as the barycenter of coordinates associated to it during the previous step.

After each iteration the associations created by DTW may change. As it is impossible to predict how these associations will change, the two steps are repeated until convergence. Thus, DBA is an iterative strategy. Figure 4.5 shows four iterations of DBA on an example with two series. Algorithm 3 presents the complete DBA computation.

The DBA algorithm starts with an initial average series, the length and the values of which are selected by the user. Experiments have shown that a length of around T performs well in practice. Regarding the values of the initial coordinates, a random sequence or an element of the set of series are usually used. It is important to note also that DBA has a guarantee of convergence [52].

DTW Barycenter Averaging, as explained previously, consists of two steps at each iteration. First, it determines the set of associations between coordinates and thus it has to compute DTW once per series to average, that is N times. Therefore the complexity of Step 1 is $\Theta(N \cdot T^2)$. Second, it has to update the average series, which requires $\Theta(N \cdot T)$ operations. If we let I be the number of iterations of DBA, then the overall time complexity of the algorithm is:

$$\Theta(I(N \cdot T^2 + N \cdot T)) = \Theta(I \cdot N \cdot T^2) \tag{4.5}$$

As $I \ll T$, the time complexity of DBA is smaller than PSA and NLAAF ones.

DBA is currently a reference method to average a set of sequences consistently with Dynamic Time Warping. Schultz and Jain [60] showed that DBA can be seen as a majorize-minimize algorithm that converges to necessary conditions of optimality after finitely many iterations. Empirical results have showed that for increasing sample sizes the proposed stochastic subgradient (SSG) algorithm is more stable and finds better solutions in shorter time than the classical DBA algorithm on average.

Algorithm 3 DBA

$C = (C_1, ..., C_{T'})$ ▷ The initial average series
$S_1 = (s_{1_1}, ..., s_{1_T})$ ▷ The 1^{st} series to average
.
.
.
$S_n = (s_{n_1}, ..., s_{n_T})$ ▷ The n^{th} series to average
Let T be the length of time series
Let $assocTab$ be a table of size T' containing in each cell a set of coordinates associated to each coordinate of C
Let $m[T, T]$ be a temporary DTW (cost,path) matrix

$assocTab \leftarrow [0, ..., 0]$
$i = rows(dtw)$
$j = columns(dtw)$
for seq in S **do**
    $m \leftarrow DTW(C, seq)$
    $i \leftarrow T'$
    $j \leftarrow T$
    while $(i \geq 1)\&(j \geq 1)$ **do**
        $assocTab[i] \leftarrow assocTab[i] \cup seq_j$
        $(i, j) \leftarrow second(m[i, j])$
for i=1 to T **do**
    $C'_i = barycenter(assocTab[i])$
return C'

Chapter 5

Time series clustering

Clustering [79] is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups (clusters). A special type of clustering that handles dynamic data is time series clustering.

Time series clustering is an interesting data mining concept, which is motivated by several research challenges including similarity search of bio-informatics sequences [65], as well as the challenge of developing methods to recognize dynamic change in time series [31]. It is also an important task in the field of finance and marketing research [8]. For example, in a marketing database, different daily patterns of sales of a specific product in a store can be discovered. Furthermore, clustering time series has received significant attention not only as a powerful stand-alone exploratory method, but also as a preprocessing step or subroutine for other tasks such as prediction and recommendation [74].

Reviewing existing works in the literature, it is implied that there are essentially three different ways to cluster time series, namely shape-based, feature-based and model-based [74].

In the shape-based method, the shapes of two time series are matched as well as possible, by a non-linear stretching and contracting of the time axes. This method has also been labelled as a raw-data-based method since it typically works directly with the raw data. Shape-based algorithms usually employ conventional clustering algorithms, which are compatible with static data while their distance measure has been modified with an appropriate one for time series.

In the feature-based approach, static features from each time series are calculated. This greatly reduces the computational complexity of the clustering process. Although the feature extraction process is often generic, extracted features need not be similar for all applications. Every feature should be

chosen based on its suitability for the domain in question.

In model-based methods, a raw time series is transformed into model parameters (we assume an underlying parametric model for each time series) and then a suitable model distance and a clustering method is chosen and applied to the extracted model parameters. However, it is shown that usually model-based approaches have scalability problems, and their performance reduce when the clusters are close to each other.

In this thesis, we only consider shape-based clustering. Our goal is to make k groups from n unlabelled objects in the way that each group contains at least one object (partitioning clustering). The number k is considered known beforehand.

A shared component of many different approaches for time series clustering is dimension reduction. Time series dimension reduction is known as time series representation as well. A representation method represents the raw time series in another space by transforming them to a lower dimensional space. A conventional clustering algorithm can be applied then to cluster these representations instead of the whole time series. The reduced representation can also be stored in memory instead of the whole series saving a large amount of memory.

In this chapter, we initially describe some representation methods for time series (Section 5.1) and then present two basic algorithms for partitioning shape-based clustering (Section 5.2).

5.1 Representation methods

Data representation is one of the main challenging issues for time series clustering. Because, all raw time series data cannot fit in the main memory [42] that increases the need for dimensionality reduction. In addition, the time series data are multidimensional, which is a difficulty for many clustering algorithms to handle, and it slows down the calculation of the similarity measurement. Consequently, it is very important to represent the data without slowing down the algorithm's execution time and without a significant data loss. In fact, it is a trade-off between speed and quality and all efforts must be made to obtain a proper balance point between quality and execution time. Some requirements can be listed for any time series representation methods [49]:

- Significantly reduce the data size.
- Maintain the local and global shape characteristics of the time series.

- Acceptable computational cost.
- Reasonable level of reconstruction from the reduced representation.
- Insensitivity to noise or implicit noise handling.

In taxonomy of representations [9], there are generally four representation types: non-data adaptive, data adaptive, model-based and data dictated representation approaches as are depicted in Figure 5.1.

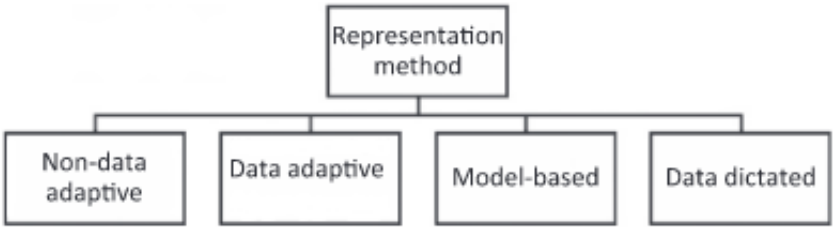


Figure 5.1: Hierarchy of different time series representation approaches. Source: Time-series clustering - A decade review (Fig. 4 in [9]).

5.1.1 Non-data adaptive

Non data-adaptive techniques use the same set of parameters for dimensionality reduction regardless of the underlying data. The first technique suggested for dimensionality reduction of time series was the Discrete Fourier Transform (DFT) [10].

The basic idea of spectral decomposition is that any signal, no matter how complex, can be represented by the superposition of a finite number of sine (or/and cosine) waves, where each wave represented by a single complex number known as a Fourier coefficient. A time series represented in this way is said to be in the frequency domain. Agrawal et al. [10] observed that only the first few waves appear to be dominant and therefore the rest can be omitted without any great impact on the reconstruction error. Thus the final time series representation after DFT are the coefficients of the first k waves. A very important property of DFT for data mining applications is Parseval's Theorem. It states that the total energy of a signal in the time domain is preserved in its projection into frequency space [40]. Disregarding the error introduced by the truncation at k, this means that the euclidean distance will hold the same for the original signal as its transformation. The reduction of dimensionality is implied by the usage of k complex coefficients. Furthermore, the Parseval's Theorem provides that the distance between

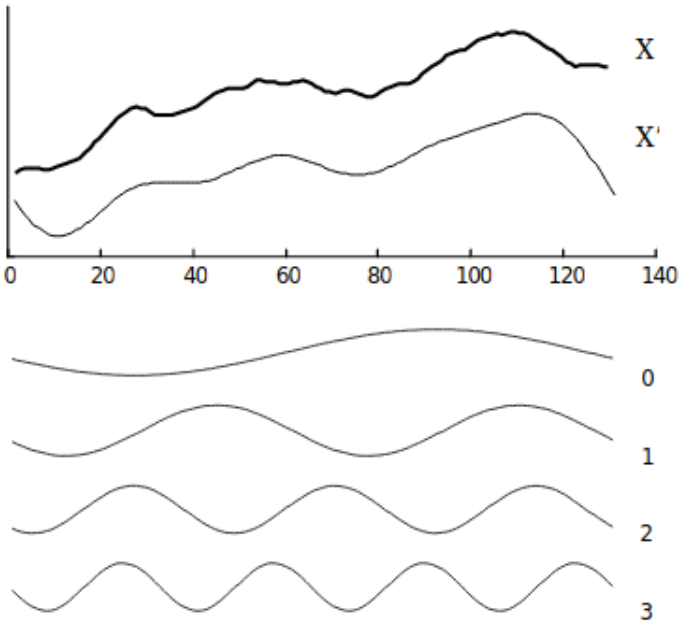


Figure 5.2: The first 4 Fourier bases can be combined in a linear combination to produce X', an approximation of the sequence X. Note each basis wave requires two numbers to represent it (phase and magnitude). Source: Dimensionality Reduction for Fast Similarity Search in Large TimeSeries Databases (Fig. 4 in [39]).

series is preserved and DFT can be calculated efficiently with $O(nlogn)$. A concern is that the coefficient truncation of positive terms at k causes the distance in the frequency space to be less than the truth distance, resulting in false positives in applications such as similarity search.

A related approach is the Discrete Wavelet Transform (DWT) [80]. While DFT uses sinusoidal waves to represent the general shape of a time sequence, DWT processes the series at different scales and resolution. DWT uses localised wavelets of final energy to represent the data. A key advantage it has over Fourier transforms is temporal resolution: it captures both frequency and location information (location in time). A mother wavelet defines the overall shape and further analysing wavelets derived through shift and scaling add the necessary details to the representation. Thus the characteristics of the transform can be controlled by the choice of the mother wavelet as all further wavelets derive from it. One drawback is that classical DWT is only defined for sequences with length of powers of two, which can be overcome by zero-padding, smooth-padding or periodic extension. Although there are

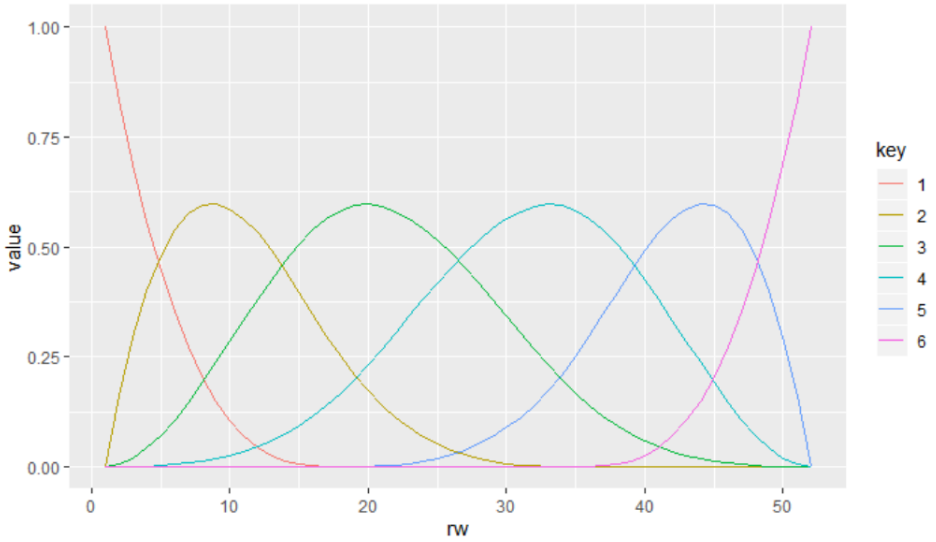


Figure 5.3: B-splines Basis with 6 knots and 3rd-Degreee Polynomials. Source: Using B-Splines and K-means to Cluster Time Series (Fig. 4 in [44]).

contradictory claims on the performance of DWT [40], DWT's superiority lies in its time complexity of $O(n)$ and the multilevel resolution.

Another approach is B-splines [44]. B-Splines are a way to approximate non-linear functions by using a piece-wise combination of polynomials. They have two components, a basis and coefficients. The basis determines the hyperparameters: how many local models to use (called knots) and what degree of polynomial to use in each model. The coefficients (weights) are then multiplied by this basis to approximate the original time series. Note that these knots are equally spaced [34]. By summarizing the raw data using the estimated spline coefficients, one obtains an efficient reduction of the dimensionality of the partitioning task and can use conventional clustering algorithms such as k-means (see Section 5.2.1) to cluster the data.

A completely different approach, especially targeting the domain of time series, is the Piecewise Aggregate Approximation (PAA) [39]. The idea is to segment a time series of length T into N consecutive sequences of same length. Then the mean is calculated for each of those sequences resulting in a new representation of N mean value points. PAA also supports comparison of series with different lengths and supports the Euclidean distance measure.

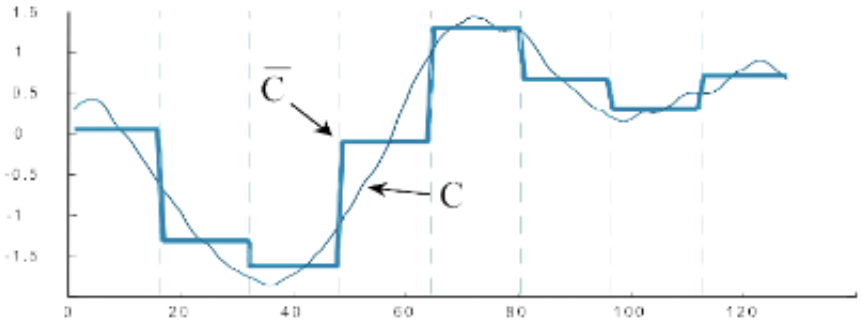


Figure 5.4: PAA method is illustrated. C represents the original time series and $\bar{C}$ is the PAA approximation using the averages of eight subsequences of equal length. Source: Time Series Clustering in the Field of Agronomy (Fig. 2.4 in [11]).

5.1.2 Data adaptive

Data adaptive representation methods are performed on all time series in datasets and try to minimize the global reconstruction error using arbitrary length (non-equal) segments.

As DFT and DWT, Singular Value Decomposition (SVD) is another transformation based approach [24]. The important difference is that while DFT and DWT apply local transformations, SVD acts globally. SVD examines the entire data and rotates the axes to maximise variance along the first few dimensions. The resulting representation consists of the first few dimensions. Although SVD is an optimal transformation in the sense of minimal reconstruction error, it requires the computation of eigenvalues for large data matrices making it computationally very expensive.

Chakrabarti et al. [38] proposed an improved and data adaptive version of PAA, called Adapive Piecewise Constant Approximation (APCA). While PAA stores the means of consecutive fixed length segments, APCA allows the segments to be of different length, thus more adapting to the data. This means that a region of low activity can be represented by one long segment and regions with high activity are depicted by several short segments. The final representation stores two numbers per segment: its mean and the segment length. In terms of dimensionality reduction, PAA with N segments corresponds to APCA with $N/2$ segments. This now gives rise to the question of how to determine the best possible segmentation for APCA. Chakrabarti et al. proposed a method achieving an almost optimal representation for APCA in $O(nlogn)$ time.

5.1.3 Model-based

Model-based approaches assume that a given time series was produced by an underlying model. Dimensionality reduction is obtained by representing the time series by the model's parameters, used to produce the series. As a consequence time series similarity is measured based on the model parameters. There are several approaches using parametric temporal models such as the ARMA [78] and ARIMA models [77]. More sophisticated approaches include Markov Chains or Hidden Markov Models (HMM) [50]. Given this variety of representations, it is a complex task to choose the best approach for a given context, as each approach has its special properties which might be inconvenient in one case but a virtue in another.

5.1.4 Data dictated

In the non-data adaptive, data adaptive, and model-based approaches user can define the compression-ratio based on the application in hand. In contrast, data dictated methods automatically determine the dimension reduction rate. The most common example of data dictated method is clipped data [12] (discretising time series data to above or below the median).

5.2 Partitioning shape-based clustering

Partitioning clustering is a type of clustering where all observations in the data are partitioned into k different clusters. This type of clustering can also be regarded as a combinatorial optimization problem, which aim at minimizing intracluster distance while maximizing intercluster distance [59]. Finding a global optimum would require trying all possible clusterings, which is infeasible even for small data sets. Therefore, several heuristics for finding local optima are developed. Two of them, k-means and k-medoids, are commonly used partitioning algorithms that build clusters around the means and medoids of observations, respectively.

These heuristics work with a distance measure. As we have seen in Chapter 3, the best on average similarity measure for shape-based clustering is Dynamic Time Warping (DTW). In this section, we present k-means (Section 5.2.1) and k-medoids (Section 5.2.2) algorithms with Dynamic Time Warping as a distance measure.

5.2.1 k-means

Even though k-means was first proposed in 1955, it is still one of the most commonly used clustering methods. The general k-means algorithm can be found in Algorithm 4.

Algorithm 4 k-means clustering algorithm (Lloyd).

1: Decide on a value for the number of clusters k.
2: Initialize k cluster centers c_1, c_2,..., c_k.
3: **while** stopping condition not met **do**
4: **for** $i = 1$ to N **do**
5: - find the nearest centroid (c_1, c_2,..., c_k) of datapoint x_i.
6: - assign the datapoint x_i to that cluster.
7: **for** $j = 1$ to k **do**
8: new centroid $c_j \leftarrow$ average of all datapoints assigned to that cluster.
9: **return** cluster assignment and centroids c_1, c_2,..., c_k.

We describe k-means algorithm for time series clustering. The first step is to determine the optimal number of clusters k. Unfortunately, there is no definitive answer to this question. The optimal number of clusters is somehow subjective and depends on the method used for measuring similarities and the parameters used for partitioning. Here, we assume that the number of clusters k is known.

The next step is to initialize k cluster centers. A common method is the random initialization method. It randomly selects k time series from the set of time series and takes these as the initial centers. To avoid finding local optima when applying k-means, some implementations of the algorithm consider multiple random initializations.

The third step in the k-means algorithm is the while-loop that runs until a stopping criteria is met. Two commonly used stopping criteria are convergence and maximum number of iterations. Convergence can take multiple forms, for example when no time series are assigned a different cluster center. Within the while-loop in step three, two for-loops can be found.

The first for-loop (steps 4-6) assigns to each time series x_i the cluster whose center has minimum distance to x_i. Here, the distance is Dynamic Time Warping (DTW). After the time series are assigned a cluster in the first for-loop, the second for-loop (steps 7-8) determines the new cluster centers. The centers are obtained by taking the DTW Barycenter Averaging (DBA), which is the best method for averaging time series, of all sequences in a given cluster. Once the stop condition for the while-loop is met, the algorithm returns the cluster assignment for each time series and the cluster centroids.

We now consider the time complexity of k-means algorithms for time series clustering. If N is the number of time series, T is the maximum time series length and k is the number of clusters then the first for-loop requires $O(N \cdot k \cdot T^2)$. If I' is the maximum number of iterations for DBA, then the second for-loop has a time complexity of $O(I' \cdot N \cdot k \cdot T^2)$. Both for-loops are repeated in the while-loop for a numbers of iterations I''. If $I = I' \cdot I''$, the whole algorithm has a time complexity of $O(I \cdot N \cdot k \cdot T^2)$, where $I, k \ll N$. It must be noted that Algorithm 4 does not guarantee finding the optimal k-means solution, due to the possibility of local minima to be found.

5.2.2 k-medoids

k-medoids is a clustering method related to k-means in the sense that its objective is to partition the time series data into k sets. The main difference between k-means and k-medoids is that in k-medoids observations are taken as centers. It is an NP-hard optimization problem and thus heuristics are used to obtain k-medoids partitions.

The most popular heuristic for k-medoids is the Partitioning Around Medoids (PAM) algorithm [37]. The pseudo-code for this algorithm can be found in Algorithm 5. Note that the initialization is similar to that of Algorithm 4. The while-loop, however, is different. Here swaps are considered in an iterative manner and if a swap decreases the total sum of distances between all time series and their closest medoid, the swap is made. Here only the distance measure is necessary, which is again Dynamic Time Warping (DTW).

Algorithm 5 Partitioning Around Medoids (PAM) algorithm.

```
1: Decide on a value for the number of clusters k.
2: Initialize k observations as the initial medoids.
3: while no change in the centroid assignment do
4:     for each medoid c do
5:         for each non-medoid observation o do
6:             if swapping c and o improves the solution then
7:                 Swap c and o.
8: return the k medoids and the cluster assignment.
```

We now examine the time complexity of the PAM algorithm. Observe that in every iteration, for all k medoids, $(N - k)$ swaps are considered. Calculating the swapping cost for any of these swaps has time complexity $O(N - k)$. This gives every iteration in the PAM algorithm a time complexity

of $O(k \cdot (N-k)^2)$ and thus the whole algorithm has a time complexity of $O(I \cdot k \cdot (N-k)^2)$. Note that this time complexity does not include the time complexity of determining the distance (DTW) matrix. The main advantage k-medoids has over k-means is that it is more robust in the presence of noise and outliers. Also, being able to calculate the distance matrix in advance is an advantage. A disadvantage of k-medoids compared to k-means is that it comes with a higher time complexity.

Chapter 6

Time series modelling with Gaussian Processes

Gaussian Process is a powerful machine learning technique [55]. Its simplest and most useful application is fitting a function to the data. This is called regression and can be successfully used in time series modelling without any adjustment. A regression function $f(t)$ of a time series is just a function that depends on time. For that reason, in this chapter we present modelling (regression) with Gaussian Processes for any arbitrary dataset.

For a given set of data points, there are potentially infinitely many functions that fit the data. Gaussian Processes offer an elegant solution by assigning a probability to each of these functions. The mean of this probability distribution then represents the most probable characterization of the data. Additionally, being a Bayesian method allows us to incorporate the confidence of the prediction into the regression result.

In this chapter we initially give a detailed description of Gaussian Processes (Section 6.1) and then present the basics of regression using Gaussian Processes (Section 6.2). In the last section, we describe a well-known approximation method for Gaussian Process Regression (Section 6.3).

6.1 Gaussian Process

To better understand Gaussian Processes, we first need to understand the mathematical foundation that they are built on. As the name suggests, the Gaussian distribution is the basic building block of Gaussian Processes. In particular, we are interested in the multivariate case, where each random variable is distributed normally and their joint distribution is also Gaussian.

In the following, we first explore some properties of multivariate Gaussian

distributions (Section 6.1.1), we give a definition of Gaussian Processes (Section 6.1.2) and then overview their components (Section 6.1.3).

6.1.1 Multivariate Gaussian distribution

The multivariate normal distribution or multivariate Gaussian distribution is a generalization of the one-dimensional (univariate) normal distribution to higher dimensions. One definition is the following [84]:

Definition 6.1.1 (Multivariate Gaussian distribution)**.** A random vector is said to be k-variate normally distributed if every linear combination of its k components has a univariate normal distribution.

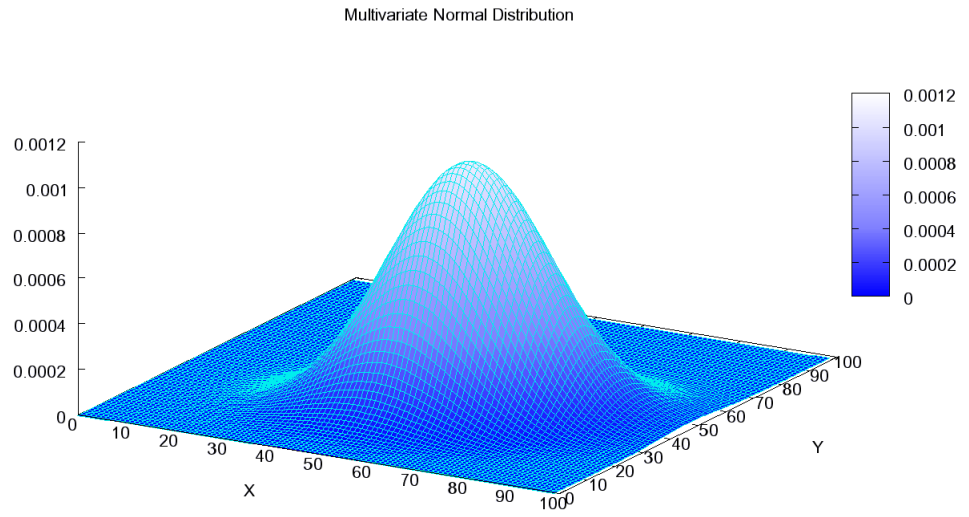


Figure 6.1: Multivariate Normal Distribution. Source: Wikipedia - Multivariate normal distribution (Fig. 2 in [84]).

The multivariate Gaussian distribution is defined by a mean vector $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$. The mean vector $\boldsymbol{\mu}$ describes the expected value of the distribution and each of its components describes the mean of the corresponding dimension. The covariance matrix $\boldsymbol{\Sigma}$ models the variance along each dimension and determines how the random variables are correlated. $\boldsymbol{\Sigma}$ is always symmetric and positive semi-definite. The diagonal of the matrix consists of the variance σ_i^2 of the i-th random variable. The off-diagonal

elements σ_{ij} describe the correlation between the i-th and j-th random variable.

To denote that $\boldsymbol{X}$ follows a normal distribution, we write:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{6.1}$$

The covariance matrix $\boldsymbol{\Sigma}$ describes the shape of the distribution and is defined as follows:

$$\boldsymbol{\Sigma} = \mathrm{Cov}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathbb{E}\left[(\boldsymbol{x}_i - \boldsymbol{\mu}_i)(\boldsymbol{x}_j - \boldsymbol{\mu}_j)^T\right] \tag{6.2}$$

Gaussian distributions are so useful in statistical modelling because they have the nice algebraic property of being closed under conditioning and marginalization. Being closed under conditioning and marginalization means that the resulting distributions from these operations are also Gaussian, which makes many problems tractable. Through marginalization we can extract partial information from multivariate probability distributions. Given a normal probability distribution over random variables $\boldsymbol{X}$ and $\boldsymbol{y}$, we can easily compute the marginalized probability distributions of $\boldsymbol{X}$ or $\boldsymbol{y}$. On the other hand, conditioning is used to determine the probability of one variable depending on another variable. This operation is the cornerstone of Gaussian Processes since it allows Bayesian inference.

6.1.2 Definition

Now that we have explored some of the basic properties of multivariate Gaussian distributions, we are ready to define Gaussian Processes. We can view Gaussian Process as a generalization of multivariate Gaussian distribution to infinitely many variables. A formal definition of Gaussian Processes is [82]:

Definition 6.1.2 (Gaussian Process)**.** A Gaussian Process is a stochastic Process (a collection of random variables indexed by time or space), such that every finite collection of those random variables has a multivariate normal distribution, i.e. every finite linear combination of them is normally distributed.

The distribution of a Gaussian Process is the joint distribution of all those (infinitely many) random variables, and as such, it is a distribution over functions with a continuous domain, e.g. time or space. Just like a Guassian

distribution, a Gaussian Process is completely defined by a mean function $\boldsymbol{m}(\boldsymbol{x})$ giving the mean at any point of the input space and a covariance matrix $\boldsymbol{\Sigma}$ which is defined by its covariance function $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x'})$ that sets the covariance between points. The covariance function is also called kernel function. To denote that a function $\boldsymbol{f}$ follows a Gaussian Process, we write:

$$\boldsymbol{f}(\boldsymbol{x}) \sim \mathcal{GP}(\boldsymbol{m}(\boldsymbol{x}), \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x'})) \tag{6.3}$$

6.1.3 Kernels

The mean $\boldsymbol{m}(\boldsymbol{x})$ can take any value. If we have domain knowledge about the expected values of the function $\boldsymbol{f}$ at every location, we can encode this knowledge in $\boldsymbol{m}$. But in most cases, we have no idea. So it is usual to define the mean to be a zero function ($\boldsymbol{m}(\boldsymbol{x}) = \boldsymbol{0}$). This is possible because you can always normalize your data so they have zero mean. Furthermore, using a fixed (deterministic) mean function $\boldsymbol{m}(\boldsymbol{x})$ is trivial [55]: Simply apply fixed mean function the usual zero mean GP to the difference between the observations and the fixed mean function.

The crucial ingredient in Gaussian Processes is the way we set up the covariance matrix. The covariance matrix should be positive definite and encodes our assumptions about the function which we wish to learn. The matrix is generated by evaluating the kernel k, which is also called covariance function, pairwise on all the points. The kernel receives two points $\boldsymbol{x}, \boldsymbol{x'} \in \mathbb{R}^n$ as an input and returns a similarity measure between those points in the form of a scalar.

$$k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \quad \boldsymbol{\Sigma} = Cov(\boldsymbol{x}, \boldsymbol{x'}) = k(\boldsymbol{x}, \boldsymbol{x'}) \tag{6.4}$$

A basic similarity assumption is that data points with inputs $\boldsymbol{x}$ which are close are likely to be similar. In the language of Gaussian Processes, similar means to have high positive covariance.

Kernel functions are widely used in machine learning, because they conceptually embed the input points into a higher dimensional space in which they then can measure the similarity efficiently. This approach is called the "kernel trick" [83].

Kernels are separated into stationary and non-stationary kernels. Stationary kernels are functions invariant to translations, and the covariance of two points is only dependent on their relative position. Thus, they are functions of $\boldsymbol{x} - \boldsymbol{x'}$. If further the covariance function is a function only of $||\boldsymbol{x} - \boldsymbol{x'}||$ then it is called isotropic. Non-stationary kernels do not have this constraint and depend on an absolute location. In the following, we explore two of the most commonly used stationary kernels.

6.1.3.1 RBF kernel

The Radial Basis Function kernel, Gaussian kernel or Squared Exponential Kernel has the form:

$$k_{\mathrm{SE}}(\boldsymbol{x}, \boldsymbol{x'}) = \sigma_f^2 \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x'}||^2}{2\ell^2}\right) \tag{6.5}$$

The RBF kernel has become the de-facto default kernel for Gaussian Processes [85]. It is universal and you can integrate it against most functions that you need to. It also has only two parameters:

- The lengthscale l determines the length of the 'wiggles' in the function $\boldsymbol{f}$.
- The output variance σ_f^2 determines the average distance of the function $\boldsymbol{f}$ away from its mean. Every kernel has this parameter out in front; it's just a scale factor.

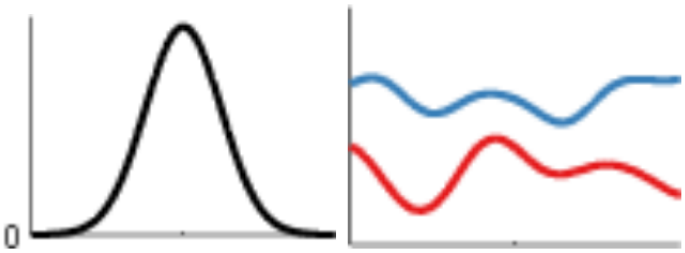


Figure 6.2: RBF kernel. Source: The Kernel Cookbook: Advice on Covariance functions (Fig. 1 in [22]).

6.1.3.2 Matérn kernel

The class of Matérn kernels is a generalization of the RBF [55]. It has an additional parameter ν which controls the smoothness of the resulting function. The smaller ν, the less smooth the approximated function is. As $\nu \to \infty$, the kernel becomes equivalent to the RBF kernel. Important values are $\nu = 1.5$ (once differentiable functions) and $\nu = 2.5$ (twice differentiable functions).

The kernel is given by:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{\Gamma(\nu)2^{\nu-1}}\left(\frac{\sqrt{2\nu}}{l}d(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}}{l}d(\boldsymbol{x}_i, \boldsymbol{x}_j)\right) \tag{6.6}$$

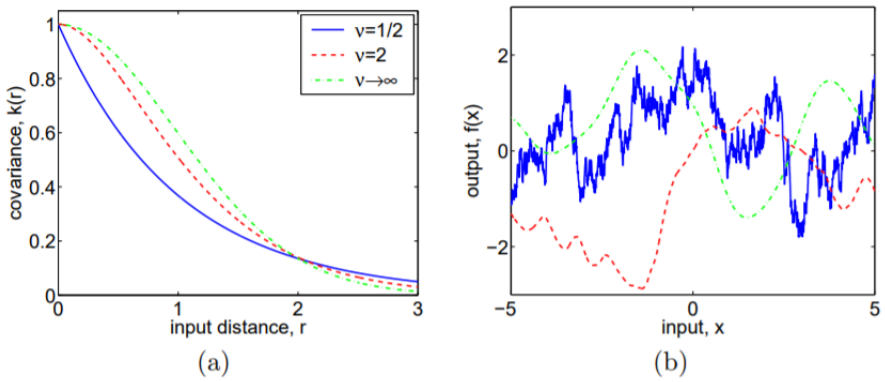


Figure 6.3: Matérn kernel for different values of ν. Source: Gaussian Processes for Machine Learning, Chapter 4: Covariance Functions (Fig. 4.1 in [55]).

where $d_{euc}(\cdot,\cdot)$ is the Euclidean distance, $K_\nu(\cdot)$ is a modified Bessel function and $\Gamma(\cdot)$ is the gamma function.

6.2 Gaussian Process Regression

In this section, we review Gaussian Process Regression. Regression is a set of statistical processes for estimating the relationships between a dependent variable and one or more independent variables, and is widely used for prediction and forecasting [87]. More specifically, it is a supervised learning problem in which we wish to learn a mapping from inputs to continuously valued outputs, given a training set of input-output pairs.

Parametric models for regression assume that the training data has been generated by an underlying function f defined in terms of some parameters $\boldsymbol{w}$. The functional mapping f along with a particular parameter set $\boldsymbol{w}$ defines the model. The goal is to find the set of parameters that provide the "best" explanation of the data. Some examples of parametric regression are linear regression (Figure 6.4) and polynomial regression [76]. The main problem with parametric regression is that complex models usually have poor generalization performance and suffer from overfitting. Overfitting can be avoided by using a simpler model. However, if the model is too simple, its predictive performance in the training data will be poor [14].

The Bayesian framework is an alternative approach for regression that counters the problem of overfitting [89]. It works by specifying a prior distribution, $p(\boldsymbol{w})$, on the parameters $\boldsymbol{w}$ of a functional form, and relocating probabilities based on evidence (i.e. observed data) using Bayes' Rule.

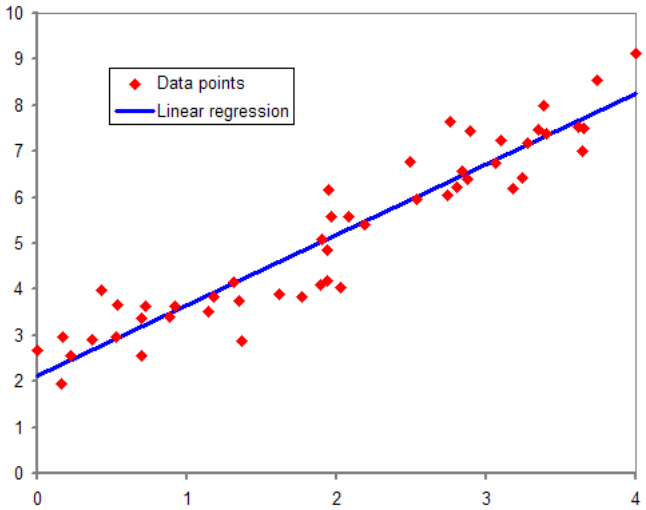


Figure 6.4: Linear Regression. Source: Regression analysis (Fig. 1 in [87]).

The updated distribution $p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{X})$, called the posterior distribution, thus incorporates information from both the prior distribution and the dataset.

Gaussian Process Regression (GPR) is a Bayesian nonparametric model (i.e. not limited by a functional form) [55], so rather than calculating the probability distribution of parameters of a specific function, GPR calculates the probability distribution over all admissible functions that fit the data. However, similar to the above, we specify a prior (on the function space), calculate the posterior using the training data, and compute the predictive posterior distribution on our points of interest.

In the next subsections, we first review the Gaussian Process model (Section 6.2.1) and then discuss how we can learn the hyperparameters of the model (Section 6.2.2).

6.2.1 Gaussian Process model

Suppose we have n training inputs $\boldsymbol{X} = [\boldsymbol{x}_1 ... \boldsymbol{x}_n]$ which reside in an input space $\mathcal{X}$, which may be continuous or discrete. The input $\boldsymbol{x}_i$ is associated with a target y_i. Combining the targets into a vector we get $\boldsymbol{y} = [y_1 ... y_n]^T$. We want to find a regression function from $\boldsymbol{X}$ to $\boldsymbol{y}$. This is a typical regression task.

We restrict our attention here to functions that have a domain $\mathcal{X}$ and range $\mathbb{R}$, meaning that $f : \mathcal{X} \to \mathbb{R}$. We assume that each observation y_i is

dependent on a latent variable f_i as follows:

$$y_i = f_i + \epsilon_i \tag{6.7}$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is i.i.d noise. This noise assumption together with the model directly gives rise to the likelihood, the probability density of the observations given the parameters.

Collecting n latent variables into a vector we have $\boldsymbol{f} = [f_1 ... f_n]^T$. In the Gaussian Process model for regression, we place a zero-mean multivariate Gaussian prior distribution over $\boldsymbol{f}$:

$$\boldsymbol{f}|\boldsymbol{X}, \theta \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}) \tag{6.8}$$

where $\boldsymbol{K}$ is a $n \times n$ covariance matrix dependent on $\boldsymbol{X}$ and some hyperparameters $\boldsymbol{\theta}$. The (i, j) element of $\boldsymbol{K}$ is equal to $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, where $k(.,.)$ is a kernel function which is parameterized by $\boldsymbol{\theta}$.

Given some observations and a covariance function, we wish to make a prediction using the Gaussian Process model. We consider a test point $\boldsymbol{x}_*$ and its associated latent variable f_*. The joint distribution of $\boldsymbol{f}$ and f_* is also a zero-mean multivariate Gaussian, and is found by augmenting (6.8) with the new latent variable f_*:

$$\begin{bmatrix} \boldsymbol{f} \\ f_* \end{bmatrix} \Bigg| \boldsymbol{X}, \boldsymbol{\theta} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K} & \boldsymbol{k} \\ \boldsymbol{k}^T & k \end{bmatrix}\right) \tag{6.9}$$

where $\boldsymbol{k} = [k(\boldsymbol{x}_1, \boldsymbol{x}_*) ... k(\boldsymbol{x}_n, \boldsymbol{x}_*)]^T$ is the $n \times 1$ vector formed from the covariance between the training inputs and x_*. The scalar $k = k(\boldsymbol{x}_*, \boldsymbol{x}_*)$.

We can now express the joint distribution over the observed targets $\boldsymbol{y}$ and test target y_* given the Gaussian noise assumption:

$$\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \Bigg| \boldsymbol{X}, \boldsymbol{\theta} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K} + \sigma^2 \boldsymbol{I} & \boldsymbol{k} \\ \boldsymbol{k}^T & k + \sigma^2 \end{bmatrix}\right) \tag{6.10}$$

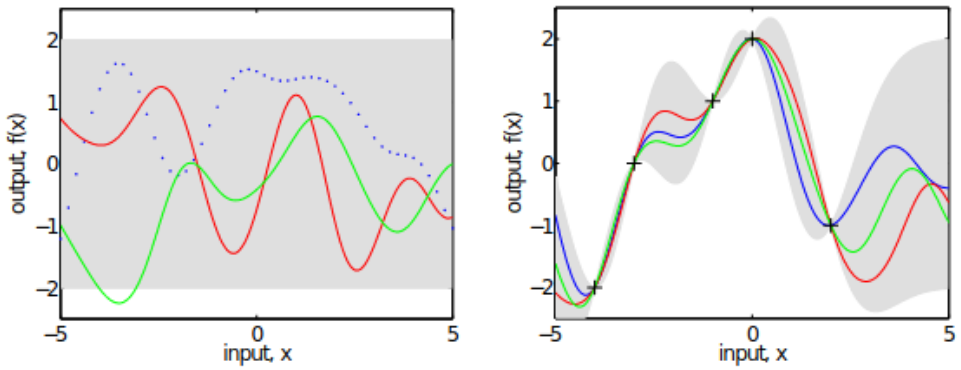
Because the joint distribution is Gaussian, we can condition on $\boldsymbol{y}$ to find the posterior distribution [55]:

$$y_*|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\theta}, \sigma^2 \sim \mathcal{N}\left(m(\boldsymbol{x}_*), var(\boldsymbol{x}_*)\right) \tag{6.11}$$

where the predictive mean and variance are:

$$m(\boldsymbol{x}_*) = \boldsymbol{k}^T (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y} \tag{6.12}$$

$$var(\boldsymbol{x}_*) = k + \sigma^2 - \boldsymbol{k}^T (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k} \tag{6.13}$$



(a), prior (b), posterior

Figure 6.5: Panel (a) shows three functions drawn at random from a GP prior. Panel (b) shows three random functions drawn from the posterior, i.e. the prior conditioned on the five noise free observations indicated. In both plots the shaded area represents the point wise mean plus and minus two times the standard deviation for each input value, for the prior and posterior respectively.
Source: Gaussian Processes for Machine Learning, Chapter 2: Regression (Fig. 2.2 in [55]).

More generally, we can compute the multivariate Gaussian predictive distribution for a set of m test points $\boldsymbol{X}_* = [\boldsymbol{x}_{1*}...\boldsymbol{x}_{m*}]$ as follows:

$$m(\boldsymbol{X}_*) = \boldsymbol{K}_*^T(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y} \tag{6.14}$$

$$cov(\boldsymbol{X}_*) = \boldsymbol{K}_{**} + \sigma^2\boldsymbol{I} - \boldsymbol{K}_*^T(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{K}_* \tag{6.15}$$

where $\boldsymbol{K}_*$ is a $n \times m$ matrix of covariances between the training inputs and test points, and $\boldsymbol{K}_{**}$ is a $m \times m$ matrix of covariances between the test points.

In Gaussian Process Regression we find a posterior density over the latent variables and then integrate over that posterior density to make predictions. We can perform the integral analytically because all the distributions are Gaussian. To find the marginal likelihood of the model, we have to calculate the integral over the product of the likelihood function and the prior density. The marginal likelihood is given by:

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}, \sigma^2) = \int p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{X}, \boldsymbol{\theta}, \sigma^2)p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{\theta})d\boldsymbol{f} \tag{6.16}$$

$$= \int \mathcal{N}(\boldsymbol{f}, \sigma^2\boldsymbol{I})\mathcal{N}(\boldsymbol{0}, \boldsymbol{K}) \tag{6.17}$$

$$= \frac{1}{(2\pi)^{n/2}|\boldsymbol{K} + \sigma^2\boldsymbol{I}|^{1/2}} exp(-\frac{1}{2}\boldsymbol{y}^T(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}) \tag{6.18}$$

For numerical reasons the log marginal likelihood is useful:

$$log\, p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}, \sigma^2) = -\frac{n}{2}log2\pi - \frac{1}{2}log|\boldsymbol{K} + \sigma^2\boldsymbol{I}| - \frac{1}{2}\boldsymbol{y}^T(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y} \tag{6.19}$$

which is the log-evidence of this Gaussian Process model.

6.2.2 Learning the Hyperparameters

The GPR model has model parameters, the hyperparameters $\boldsymbol{\theta}$ of the covariance function and the observation variance σ^2. We introduced them as part of the modelling process, but we don't know what values we should set those model parameters to. We can see how the values of the parameters affect the prediction of the model for a SE kernel in Figure 6.6. For the model to be useful, we need a principled method to find optimal values for the parameters $\boldsymbol{\theta}$, σ^2. Optimal means that with those values, the model can best explain the training data.

In many cases we may have a prior belief about the form of the data. To incorporate this information into the learning of the values for the parameters, we can first construct a prior distribution on the hyperparameters $\boldsymbol{\theta}$ and the noise variance σ^2. We then find the posterior density over $\boldsymbol{\theta}$, σ^2 as follows:

$$p(\boldsymbol{\theta}, \sigma^2|\boldsymbol{y}, \boldsymbol{X}) \propto p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}, \sigma^2)\, p(\boldsymbol{\theta}, \sigma^2) \tag{6.20}$$

which is the likelihood function times the prior density. We now can find the parameters maximizing the posterior density. This gives us the so called maximum a posteriori, or MAP values, which are used to make predictions [26].

For simplicity, the prior density is often ignored and the maximization of the posterior probability of the parameters reduces to maximization of the likelihood. The maximization problem is non-convex in general and it is carried out through gradient-based optimization techniques such as the Adam optimization algorithm [41], or L-BFGS [75]. Numerically, it is easier to perform this maximization in the *log* domain with eq. (6.19).

As with all non-convex optimization problems, local optima can be a problem, although perhaps not too serious a one in practice, since the number of hyperparameters is usually small relative to the number of data points [43]. The asymptotic time complexity of Gaussian Process Regression is dominated by the inversion of $\boldsymbol{K}$, the covariance matrix, which takes $O(n^3)$ time.

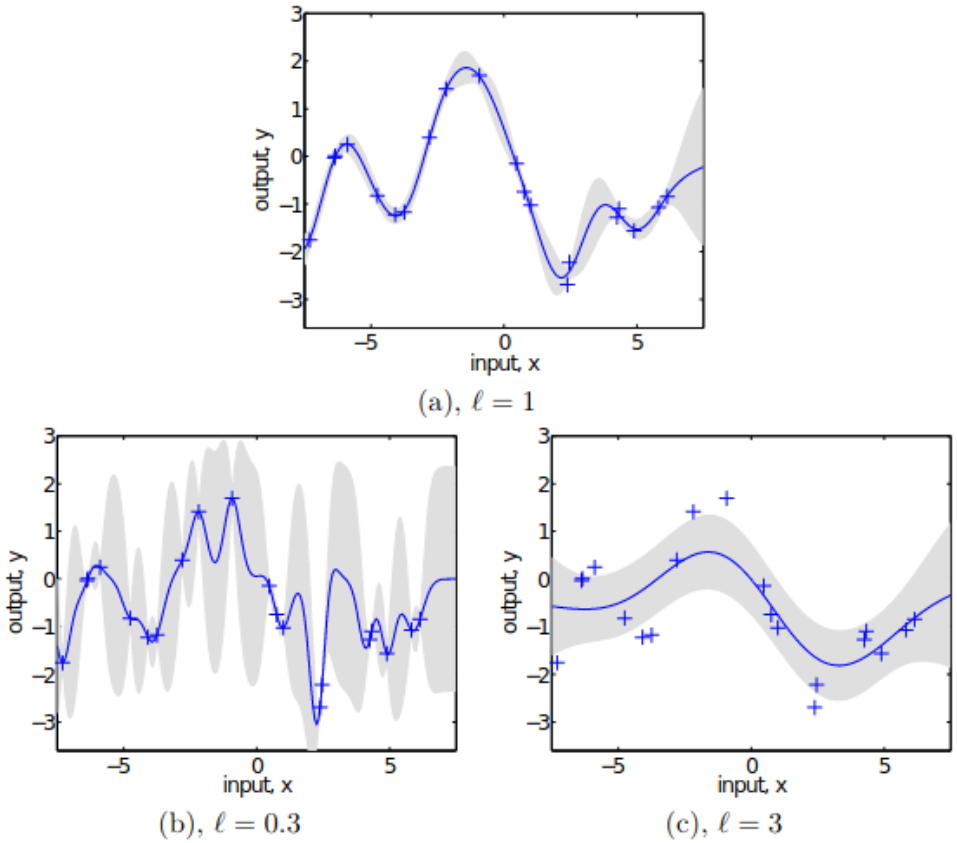


Figure 6.6: (a) Data is generated from a GP with hyperparameters (l, σ_f, σ) =(1,1,0.1), as shown by the + symbols. Using Gaussian process prediction with these hyperparameters we obtain a 95% confidence region for the underlying function f (shown in grey). Panels (b) and (c) again show the 95% confidence region, but this time for hyperparameter values (0.3,1.08,0.00005) and (3.0,1.16,0.89) respectively.
Source: Gaussian Processes for Machine Learning, Chapter 2: Regression. (Fig. 2.5 in [55])

6.3 Sparse Gaussian Process Regression

Although Gaussian Processes have many desirable properties from a modelling point of view, they can handle regression problems with at most a few thousand training cases on today's desktop machines, due to their cubic complexity. To overcome the computational limitations of GPs, numerous authors have suggested a wealth of sparse approximation methods. Common to most of these approximation schemes is that only a subset of the latent

variables are treated exactly, and the remaining variables are given some approximate, but computationally cheaper treatment [54].

Maybe the most popular family of sparse approximation schemes are inducing point methods. These methods can be seen as learning about a small number ($m < n$) quantities that are highly informative of what the posterior Gaussian Process is doing more globally. Inducing point methods have managed to scale up GP training and predictions to large datasets (Smola & Bartlett [62]; Quinonero-Candela & Rasmussen [53]; Snelson & Ghahramani [63]; Hensman et al. [33]; (Wilson & Nickisch [90]). In the following, we present the elegant (VFE) Variational Free Energy framework (or Sparse Gaussian Process Regression, SGPR) proposed by Titsias [68] in 2009.

6.3.1 Variational Free Energy

We wish to define a variational approximation to the posterior GP mean and covariance function retaining the exact GP prior. For this reason, we introduce a set of m extra variables ($m < n$), the inducing variables (or pseudo-data points): let the vector $\boldsymbol{u}$ contain values of the latent function f at locations (or pseudo-inputs) $\boldsymbol{Z} = [\boldsymbol{z}_1 ... \boldsymbol{z}_m]$ which live in the same space as $\boldsymbol{X}$.

In the VFE framework, one augments the joint distribution $p(\boldsymbol{y}, \boldsymbol{f})$ with the inducing variables $\boldsymbol{u}$ so that the joint becomes:

$$p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}) = p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}, \boldsymbol{u}) = p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u}) \tag{6.21}$$

We assume that $\boldsymbol{u}$ are function values drawn from the same GP prior as the training function values $\boldsymbol{f}$. The joint distribution of the latent function values $\boldsymbol{f}$ and the inducing variables $\boldsymbol{u}$ according to the prior then becomes:

$$p(\boldsymbol{f}, \boldsymbol{u}) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{f} \\ \boldsymbol{u}\end{bmatrix} \middle| \boldsymbol{0}, \begin{bmatrix}\boldsymbol{K} & \boldsymbol{K}_u \\ \boldsymbol{K}_u^T & \boldsymbol{K}_{uu}\end{bmatrix}\right) \tag{6.22}$$

where $\boldsymbol{K}_u = \boldsymbol{K}_{fu}$ and $\boldsymbol{K}_u^T = \boldsymbol{K}_{uf}$. Applying the multivariate Gaussian conditional rule to eq. (6.22) we derive:

$$p(\boldsymbol{f}|\boldsymbol{u}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{K}_u\boldsymbol{K}_{uu}^{-1}\boldsymbol{u}, \boldsymbol{K} - \boldsymbol{K}_u\boldsymbol{K}_{uu}^{-1}\boldsymbol{K}_u^T) \tag{6.23}$$

Applying the multivariate Gaussian marginalization rule to eq. (6.22) we get:

$$p(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{0}, \boldsymbol{K}_{uu}) \tag{6.24}$$

We want to approximate the exact posterior distribution $p(\boldsymbol{f}, \boldsymbol{u}|\boldsymbol{y})$ by using the set of m auxiliary inducing variables $\boldsymbol{u}$ evaluated at the pseudo-inputs $\boldsymbol{Z}$, which are independent from the training inputs. To approximate $p(\boldsymbol{f}, \boldsymbol{u}|\boldsymbol{y})$, we introduce the variational distribution $q(\boldsymbol{f}, \boldsymbol{u})$. We wish that $\boldsymbol{u}$ is a sufficient statistic for the parameter $\boldsymbol{f}$ in the sense that $\boldsymbol{c}$ and $\boldsymbol{f}$ are independent given $\boldsymbol{u}$, i.e. it holds $p(\boldsymbol{c}|\boldsymbol{u}, \boldsymbol{f}) = p(\boldsymbol{c}|\boldsymbol{u})$ for any $\boldsymbol{c}$. Then $\boldsymbol{u}$ summarizes the training data, and $q(\boldsymbol{c}) = p(\boldsymbol{c}|\boldsymbol{y})$ and $p(\boldsymbol{f}|\boldsymbol{u}) = p(\boldsymbol{f}|\boldsymbol{u}, \boldsymbol{y})$ are true. However, in practice the assumption of $\boldsymbol{u}$ being a sufficient statistic is unlikely to hold and we should expect $q(\boldsymbol{c})$ to be only an approximation to the exact predictive distribution $p(\boldsymbol{c}|\boldsymbol{y})$. Notice also that the quality of the approximation will crucially depend on the locations $\boldsymbol{Z}$ of the inducing variables, which are variational parameters.

For an optimal setting of the inducing variables, the exact posterior $p(\boldsymbol{f}, \boldsymbol{u}|\boldsymbol{y})$ factorises as $p(\boldsymbol{f}, \boldsymbol{u}|\boldsymbol{f}) = p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u}|\boldsymbol{y})$. This tells us that the variational distribution $q(\boldsymbol{f}, \boldsymbol{u})$, which is a joint distribution, must satisfy the same factorization as well:

$$q(\boldsymbol{f}, \boldsymbol{u}) = p(\boldsymbol{f}|\boldsymbol{u})q(\boldsymbol{u}) \tag{6.25}$$

where $p(\boldsymbol{f}|\boldsymbol{u})$ is given by eq. (6.23), and we define the marginal variational distribution $q(\boldsymbol{u})$ to be a multivariate Gaussian distribution:

$$q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{\mu}, \boldsymbol{A}) \tag{6.26}$$

To find an approximation to the exact posterior $p(\boldsymbol{f}, \boldsymbol{u}|\boldsymbol{y})$ by $q(\boldsymbol{f}, \boldsymbol{u})$, we want to minimize the KL divergence:

$$KL[q(\boldsymbol{f}, \boldsymbol{u})||p(\boldsymbol{f}, \boldsymbol{u}|\boldsymbol{y})] = \int q(\boldsymbol{f}, \boldsymbol{u}) \, log \frac{q(\boldsymbol{f}, \boldsymbol{u})}{p(\boldsymbol{f}, \boldsymbol{u}|\boldsymbol{y})} d\boldsymbol{f} d\boldsymbol{u} \tag{6.27}$$

Minimizing the KL divergence in eq. (6.27) is equivalently expressed as the maximization of the following variational lower bound on the true log marginal likelihood [68] (evidence lowerbound (ELBO) or variational free energy (VFE)):

$$log \, p(\boldsymbol{y}) = log \int p(\boldsymbol{y}|\boldsymbol{f}) p(\boldsymbol{f}|\boldsymbol{u}) p(\boldsymbol{u}) d\boldsymbol{f} d\boldsymbol{u} \tag{6.28}$$

$$\geq F_V(\boldsymbol{Z}, \boldsymbol{\theta}) = \int p(\boldsymbol{f}|\boldsymbol{u}) q(\boldsymbol{u}) log \frac{p(\boldsymbol{y}|\boldsymbol{f}) p(\boldsymbol{f}|\boldsymbol{u}) p(\boldsymbol{u})}{p(\boldsymbol{f}|\boldsymbol{u}) q(\boldsymbol{u})} d\boldsymbol{f} d\boldsymbol{u} \tag{6.29}$$

$$= \int q(\boldsymbol{u}) \Big\{ \int p(\boldsymbol{f}|\boldsymbol{u}) log \, p(\boldsymbol{y}|\boldsymbol{f}) d\boldsymbol{f} + log \frac{p(\boldsymbol{u})}{q(\boldsymbol{u})} \Big\} d\boldsymbol{u} \tag{6.30}$$

We can firstly maximize the bound by analytically solving for the optimal choice of the variational distribution q. The bound after this maximization is:

$$F_V(\boldsymbol{Z}, \boldsymbol{\theta}) = log[\mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, \sigma^2\boldsymbol{I} + \boldsymbol{Q})] - \frac{1}{2\sigma^2}Tr(\tilde{\boldsymbol{K}}) \tag{6.31}$$

where $\boldsymbol{Q} = \boldsymbol{K}_u\boldsymbol{K}_{uu}^{-1}\boldsymbol{K}_u^T$ and $\tilde{\boldsymbol{K}} = Cov(\boldsymbol{f}|\boldsymbol{u}) = \boldsymbol{K} - \boldsymbol{K}_u\boldsymbol{K}_{uu}^{-1}\boldsymbol{K}_u^T$. Details of this derivation of this bound are given in a technical report of Titsias [67]. The quantity in eq. (6.31) is computed in $O(nm^2 + m^3)$ time and is a lower bound of the true log marginal likelihood for any value of the inducing inputs $\boldsymbol{Z}$. Further maximization of the bound can be achieved by optimizing over $\boldsymbol{Z}$ using gradient-based methods. To compute the optimal q, we differentiate eq. (6.28) with respect to $q(\boldsymbol{u})$ without imposing any constraints. This gives us the mean and covariance matrix of $q(\boldsymbol{u})$:

$$\boldsymbol{\mu} = \sigma^{-2}\boldsymbol{K}_{uu}\,\boldsymbol{\Sigma}\,\boldsymbol{K}_u^T\,\boldsymbol{y} \tag{6.32}$$

$$\boldsymbol{A} = \boldsymbol{K}_{uu}\,\boldsymbol{\Sigma}\,\boldsymbol{K}_{uu}\,\boldsymbol{y} \tag{6.33}$$

where $\boldsymbol{\Sigma} = (\boldsymbol{K}_{uu} + \sigma^{-2}\boldsymbol{K}_u^T\boldsymbol{K}_u)^{-1}$. This now fully specifies our variational GP.

We are ready now to make predictions in unseen input points $\boldsymbol{X}_*$ about their function values $\boldsymbol{f}_*$. For that reason we use $p(\boldsymbol{f}_*|\boldsymbol{y})$. Following the derivation in [93] we have:

$$p(\boldsymbol{f}_*|\boldsymbol{y}) = \int p(\boldsymbol{f}_*|\boldsymbol{u})q(\boldsymbol{u}) \tag{6.34}$$

We know $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{\mu}, \boldsymbol{A})$ and we can derive the formula for $p(\boldsymbol{f}_*|\boldsymbol{u})$ by applying the multivariate Gaussian conditional rule on the prior $p(\boldsymbol{f}_*, \boldsymbol{u})$. This results in the formula for the predictive distribution:

$$p(\boldsymbol{f}_*|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{f}_*|\,\boldsymbol{B}\boldsymbol{\mu},\, \boldsymbol{B}\boldsymbol{A}\boldsymbol{B}^T + \boldsymbol{K}_{**} - \boldsymbol{B}\boldsymbol{K}_{*u}^T) \tag{6.35}$$

where $\boldsymbol{B} = \boldsymbol{K}_{*u}\boldsymbol{K}_{uu}^{-1}$.

This formula shows that we only need to invert the matrix $\boldsymbol{K}_{uu}$. It also confirms that to make predictions, the VFE model does not need training data anymore - the formula uses the inducing points. Therefore, training and prediction take $O(nm^2 + m^3)$ and $O(m^2)$ time respectively.

VFE is guaranteed to recover the true posterior when $\boldsymbol{Z} = \boldsymbol{X}$. The bound F_V monotonically improves with more resources [89]. However, local optima in the objective function may cause suboptimal solutions to be found with different initializations. In practice, an optimizer for the F_V allocates more

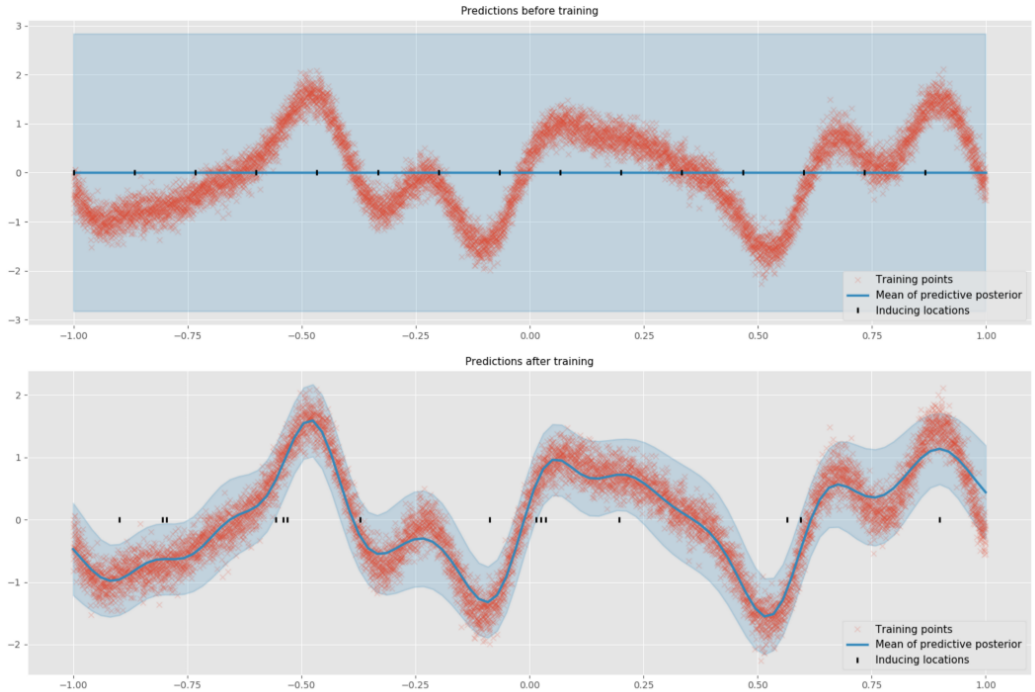


Figure 6.7: Regression with VFE framework. Source: Sparse and Variational Gaussian Process (SVGP) — What To Do When Data is Large (Fig. 4 in [93]).

inducing locations in regions where the training data changes more quickly [93].

The lower bound F_V to the marginal likelihood provides some way to assess the quality of the approximation. While knowledge of the KL divergence between the posterior and approximation would be ideal, the lower bound can guide when to add more capacity to the approximation: a halt in the increase of the bound with increasing m can indicate that the bound has become tight. A practical rule of thumb is to use as many inducing variables as your budget, such as time, memory, permits.

Burt et al. [16] showed recenlty that with high probability the KL divergence can be made arbitrarily small by growing m more slowly than n. A particular case is that for regression with normally distributed inputs in d-dimensions with the Squared Exponential kernel, $m = O(log^d n)$ suffices.

On the other hand, there are challenging datasets that remain far out of VFE framework's reach [15]. The problem arises from the fact that the method in practice is functionally local in the sense that each pseudo-data point sculpts out the approximate posterior in a small region of the input space around it. Consequently, when the range of the inputs is large compared

to the range of the dependencies in the posterior, many pseudo-data points are required to maintain the accuracy of the approximation. This means that the number of pseudo-data points must grow with the number of data points if restoration accuracy is to be maintained. In other words, m must be scaled with n and so VFE scheme has not reduced the scaling of the computational complexity in these datasets.

Chapter 7

Proposed Framework

For most shape-based clustering algorithms, the quality of the output depends highly on the distance measure used. In the last decade, a consensus has emerged that the DTW distance measure is the best measure in most time series domains, almost always outperforming the Euclidean distance and other rivals. Because DTW is intrinsically slow due to its quadratic time complexity, there are two ideas that are commonly used to mitigate the problem of using such a distance measure.

The first one is to make DTW faster. They have been proposed many methods to speed up Dynamic Time Warping. There are methods that use constraints to reduce the time complexity of DTW, such as the Sakoe-Chiba band and the Itakura parallelogram (Section 3.3.1.3). Other approaches address the complexity issue by using lower bound based pruning that provides approximately linear time complexity [19]. There are also methods that approximate Dynamic Time Warping to speed it up, such as fastDTW that performs nearly in linear time [58]. Unfortunately, most of these methods cannot be used for clustering due to the absence of an averaging method. The methods that can be used for clustering can't handle the noise and outliers that naturally time series have.

The second idea is to develop representation techniques that can reduce the dimensionality (and thus the required memory) of time series, while still preserving their fundamental characteristics. These methods reduce the complexity of clustering algorithms with DTW by performing DTW on the reduced representation of the data. There is a plethora of time series representation methods, each of them proposed for the purpose of supporting similarity search and clustering (Section 5.1).

Based on the second class of methods, we propose a novel two-stage framework called Sparse Gaussian Processes for Clustering (SGPC) for clustering time series data. SGPC uses Sparse Gaussian Process Regression (SGPR) as

a representation method and then k-means algorithm with a modified version of DTW for clustering. We first review some related work (Section 7.1) and then present our framework (Section 7.2).

7.1 Related work

The field of clustering is vast, and even the subfield of clustering time series has an enormous literature. Much of the works on time series clustering are concerned with clustering based on time series representation methods, which are directly related to our goals. Here, we are only interested in partitioning clustering based on time series shapes. In the following, we present such previous work.

Gullo et al. [29] proposed an approach for time series clustering using DSA as the representation method, DTW as the distance measure and k-means as the clustering algorithm. By using the DSA model (Derivative time series Segment Approximation) [28], which is a data adaptive representation method, a time series of length n is transformed in linear time ($O(n)$) into a new, smaller sequence by the following steps:

- **Derivation:** computation of the first derivatives of the original series to capture its significant trends. The model considers for each point the slope of the line from the left neighbour to the right neighbour.
- **Segmentation:** identification of segments consisting of tight derivative points. The segmentation of a time series of length n consists in identifying a set of break-points to partition it into p ($p \ll n$) contiguous, variable-length subsequences of points (segments) having similar features. The critical aspect in segmentation is to determine the segment break-points. DSA uses the sliding windows approach: a segment grows until the first point such that the absolute difference between it and the mean of the previous points is above a certain threshold, and the process repeats starting from the next point not yet considered.
- **Segment Approximation:** representation of each segment by involving synthetic information. All individual segments of a derivative time series are finally modeled with a synthetic information capturing their respective main features. More precisely, each segments mapped to a pair formed by the timestamp of the last point in the segment, and an angle that explains the average slope of the segment.

The authors then proposed to cluster the new smaller sequences by using k-means with DTW. This approach uses a fast representation method that

runs in linear time and reduces significantly the time complexity of time series clustering. On the downside, it has only been applied to mass spectrometry data and is also sensitive to noise.

Unfortunately, the approach mentioned above does not facilitate the removal of the noise from time series data. In order to handle noisy time series, many authors have suggested to exploit a flexible definition of the series functional form and to adopt some dimensionality reduction strategy. These methods are based on the functional data analysis framework introduced by Ramsay and Silverman [13]. The main idea is to describe the data by using optimal linear combinations of basis functions (e.g. by using B-spline bases) and to perform the clustering on the extracted signals or on the estimated basis coefficients.

An example is the proposal of Abraham, Cornillon, Matzner-Løber and Molinari [5]. They suggested a two-stage clustering method: fitting the functional data by B-splines and partitioning the estimated model coefficients using k-means algorithm. In a similar direction, Iorio et al. [34] proposed a parsimonious model-based framework for clustering longitudinal data. They modeled the raw series by a linear combination of P-spline smoothers. The series as described by the lower dimensional vectors of P-spline coefficients are then grouped by applying k-means algorithm with an appropriate distance measure, such as Dynamic Time Warping. The authors defined the splines over a domain spanned by equidistant knots across the series. This approach can be seen as in a half-way between the non-data adaptive and model based representation methods according to the classification made in Chapter 5. The main disadvantage of this category of approaches is that such approaches highly depend on the locations of the knots, which are selected by the user. The proposal is to place a generous number of equally spaced knots (but still smaller than the available observations), which has a negative impact on the time complexity of clustering.

Our framework differs from the previous ones due to the choice of the representation technique. Sparse Gaussian Process Regression (SGPR) can be seen as a model based and data adaptive representation method. SGPR can handle noisy data with outliers (model-based) and because of its adaptive nature can reduce the dimensionality of the time series significantly. The resulting representations also lead to memory reduction. In the following section, we present Sparse Gaussian Processes for Clustering (SGPC).

7.2 SGPC Framework

SGPC framework consists of two stages. First, we model the raw series by a set of inducing data points using Sparse Gaussian Process Regression (Section 7.2.1). The series as described by a (much) lower number of data points are then grouped by applying k-means algorithm. We propose two different distance measures for SGPC (Section 7.2.2) that are based on Dynamic Time Warping (DTW). In the last subsection, we study the time complexity of our framework (Section 7.2.3).

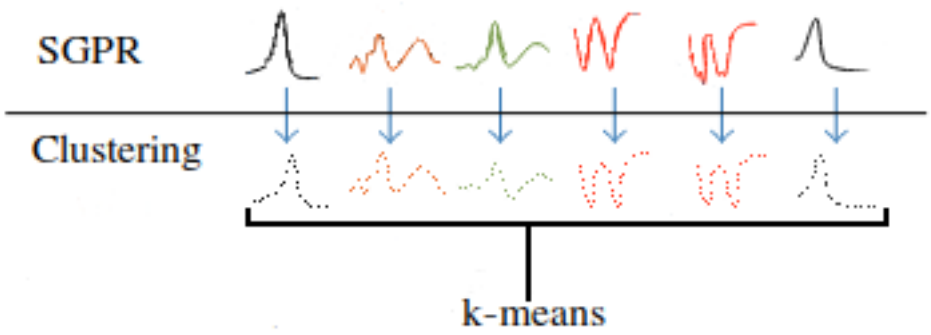


Figure 7.1: SGPC uses Sparse Gaussian Process Regression (SGPR) for dimensionality reduction and then k-means algorithm with a modified version of DTW for clustering.

7.2.1 Modelling stage

Suppose we have a dataset of N univariate time series. Each time series $\boldsymbol{X}_i$ has length T with inputs $(t_{i,1}, t_{i,2}, ..., t_{i,T})$ and corresponding outputs $(y_{i,1}, y_{i,2}, ..., y_{i,T})$. Here we suppose that the time series have equal length, but our framework can also handle time series of different lengths without any modification.

In order to denoise our data, we model each time series $\boldsymbol{X}_i$ as a latent function $\boldsymbol{f}_i$ corrupted by some random noise. We assume that each observation $y_{i,t}$ of $\boldsymbol{X}_i$ is dependent on a latent variable $f_{i,t}$ as follows:

$$y_{i,t} = f_{i,t} + \epsilon_{i,t} \tag{7.1}$$

where $\epsilon_{i,t} \sim \mathcal{N}(0, \sigma^2)$ is i.i.d noise.

We model each time series by using Sparse Gaussian Process Regression (also called Variational Free Energy). Each time series $\boldsymbol{X}_i$ is summarized

from m ($m \ll T$) inducing variables $\boldsymbol{u}_i$ and their corresponding locations $\boldsymbol{z}_i$. In fact, we get a multivariate gaussian distribution over $\boldsymbol{u}_i$, but for clustering we are only interested in their mean values. Covariance matrices are not important for clustering. They also require $O(m^2)$ in memory and every distance between them has cubic time complexity $O(m^3)$.

Therefore, we transform each time series $\boldsymbol{X}_i$ to the vector $\boldsymbol{X}'_i$ that retains the shape of the series (see Figure 7.2):

$$\boldsymbol{X}'_i = (\boldsymbol{u}_i, \boldsymbol{z}_i) = \big((u_{i,1}, z_{i,1}), (u_{i,2}, z_{i,2}), ..., (u_{i,m}, z_{i,m})\big) \tag{7.2}$$

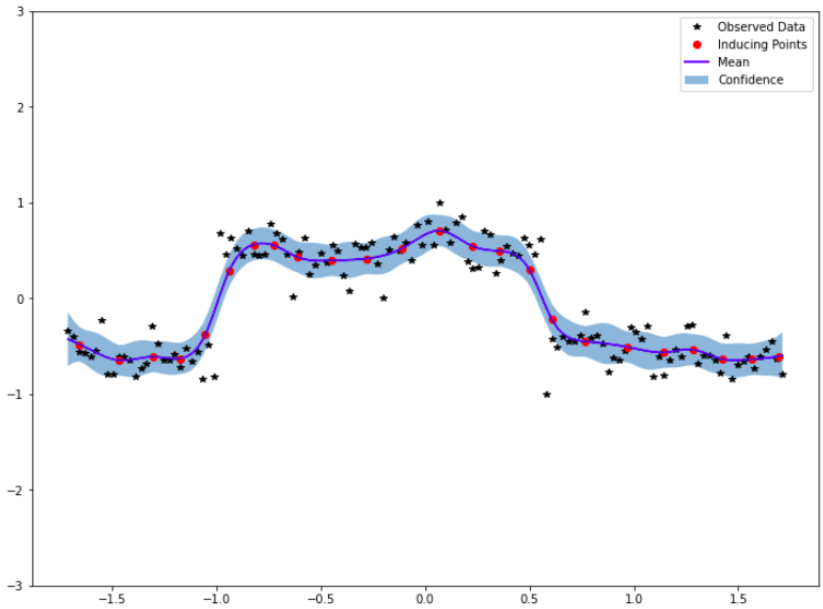


Figure 7.2: SGPR uses a set of inducing points for dimensionality reduction (red points).

The number of inducing points m is selected by the user. We should also choose a mean and a kernel function for SGPR. We usually place a constant-mean GP prior and the kernel function encodes our assumptions about the nature of the data. The only restriction here is that the kernel should be stationary. Then the learned representations are used for time series clustering.

7.2.2 Clustering stage

We wish to cluster our time series based on their shapes, so k-means algorithm with DTW is a convenient and effective method. The problem here

is that the inducing variables of two different transformed series $\boldsymbol{x}$, $\boldsymbol{y}$ have different corresponding locations $\boldsymbol{z}_x$, $\boldsymbol{z}_y$. In the following, we propose two different distance measures and their averaging methods for our framework.

7.2.2.1 Simple approach

To solve the problem of different timestamps between inducing points of different time series we use Dynamic Time Warping in two dimensions. We also add an extra parameter α changing the local cost function as follows:

$$c(x_i, y_j) = (x_i - y_j)^2 + \alpha \cdot (t_{x_i} - y_{y_j})^2 \tag{7.3}$$

The rest DTW algorithm remains unchanged. We should note that when $\alpha \to 0$ we get the 1-D version of DTW. When $\alpha \to 1$ we get the normal 2-D version of DTW. When the value of α grows, the timestamps term of the distance becomes dominant and our distance becomes a lock-step measure. We remind that the optimizer allocates more inducing locations to regions where the training data changes more quickly and allocates less inducing locations to regions where the training data is smoother. Therefore, we can say that the timestamps term captures differences between time series shape's complexity.

We can now use k-means algorithm with this modified version of DTW for clustering the inducing variables and their locations. This distance has time complexity $O(m^2)$.

As an averaging method we use DTW Barycenter Averaging without any modification. Therefore, the average method has time complexity $O(I \cdot N \cdot m^2)$, and the total complexity of the k-means algorithm is $O(k \cdot I \cdot N \cdot m^2)$, where k is the number of clusters and I is the number of iterations used.

7.2.2.2 Second approach

In the previous approach we have not take into account the power of Gaussian Process Regression. We can actually predict the function value at every test point x_* with the following formula:

$$p(\boldsymbol{f}_*|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{f}_*|\, \boldsymbol{B}\boldsymbol{\mu},\, \boldsymbol{B}\boldsymbol{A}\boldsymbol{B}^T + \boldsymbol{K}_{**} - \boldsymbol{B}\boldsymbol{K}_{*u}^T) \tag{7.4}$$

where $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{\mu}, \boldsymbol{A})$, $\boldsymbol{B} = \boldsymbol{K}_{*u}\boldsymbol{K}_{uu}^{-1}$. We want to predict only the mean value of the function's random variable. This prediction costs $O(m)$ which is too expensive.

The idea here is to approximate the predictive mean $f(x_*)$ taking into account only the first inducing point u_L to the left of x_*, and the first inducing

point u_R to the right of x_*. We approximate $f(x_*)$ at test point $x_* \in (u_L, u_R)$ between two inducing points u_L and u_R with:

$$f(x_*) \approx \begin{bmatrix} k(x_*, u_L) & k(x_*, u_R) \end{bmatrix} \cdot \begin{bmatrix} k(u_L, u_L) & k(u_L, u_R) \\ k(u_R, u_L) & k(u_R, u_R) \end{bmatrix}^{-1} \cdot \begin{bmatrix} u_L \\ u_R \end{bmatrix} \tag{7.5}$$

For ease of explanation, we assume that we have a Gaussian kernel (RBF) and a zero-mean GP prior. The distance works for every simple stationary kernel and a constant-mean GP prior. The formula 7.5 with the RBF kernel becomes:

$$f(x_*) \approx A \exp\left(-\frac{(x_* - x_{u_L})^2}{2\ell^2}\right) + B \exp\left(-\frac{(x_* - x_{u_R})^2}{2\ell^2}\right) \tag{7.6}$$

where $A, B \in \mathbb{R}$. The formula 7.6 gives us an approximation to $f(x_*)$ in constant time $O(1)$.

We wish now to use this formula to improve our distance (the simple approach). To do that we have to use (predict) more points to better capture the shape of each time series.

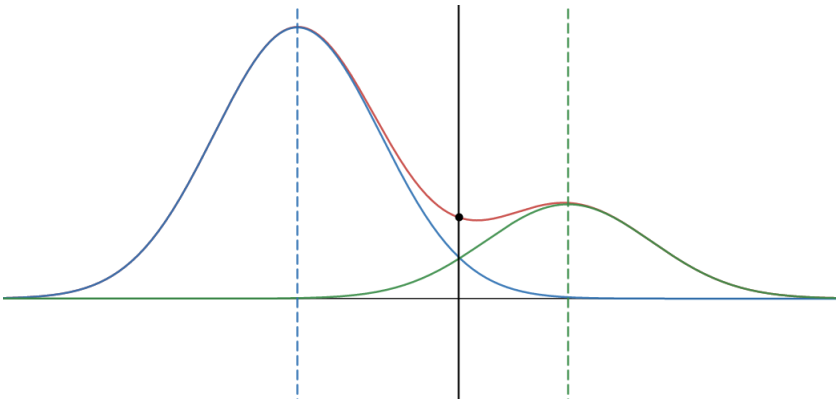


Figure 7.3: The case when $A \cdot B > 0$. The blue and green curves are the Gaussians, and the red one is the predictive curve. The black line shows us the intersection point.

The formula 7.6 consists of two Gaussian components with centers x_{u_L} and x_{u_R}, respectively. We want to find the intersection point of these two Gaussians and predict the function value at that input. We can find that point using a simple modified version of binary search at (x_{u_L}, x_{u_R}). Running binary search, until a given precision is reached, costs us practically $O(1)$.

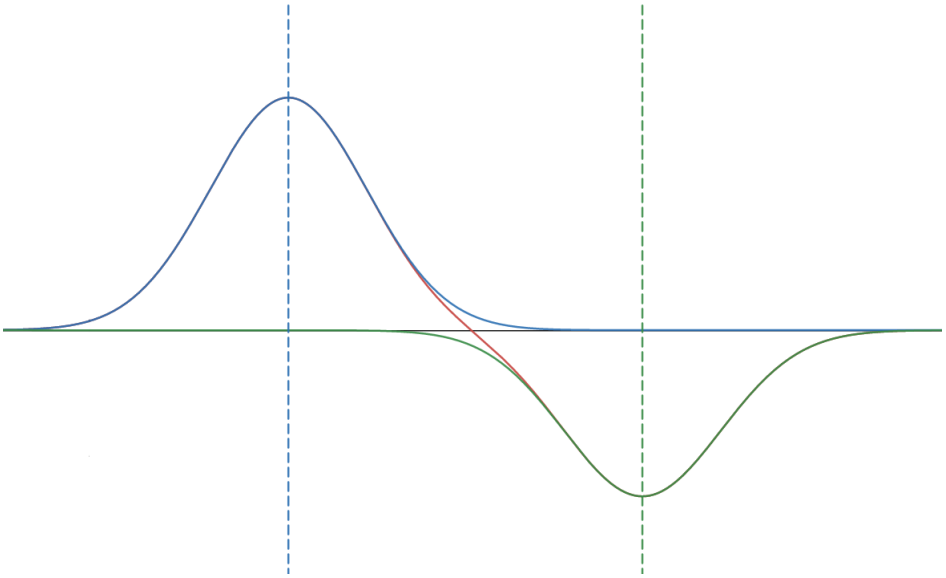


Figure 7.4: The case when $A \cdot B < 0$. The blue and green curves are the Gaussians, and the red one is the predictive curve. The two Gaussians don't intersect.

The case when $A \cdot B > 0$ is illustrated in Figure 7.3. As we can see, the extra predicted point better captures the shape of the curve between the centers of the two Gaussians.

When $A \cdot B < 0$, the two Gaussian components don't intersect as we can see in Figure 7.4. The solution is to reflect the negative component over x-axis and then to find the intersection point. This is illustrated in Figure 7.5.

The new distance is also based on Dynamic Time Warping with cost function:

$$c(x_i, y_j) = d_*(x_i, y_j)^2 + \alpha \cdot d_*(t_{x_i}, t_{y_j})^2 \tag{7.7}$$

The distance d_* uses the predictions as presented above to better compare the local shapes of two inducing points. Let $pred(x_l, x_r)$ be the predicted value at the intersection point between inducing points x_l and x_r. The distance d_* is:

$$\begin{aligned} d_*(x_i, y_j) = |x_i - y_j| + |pred(x_{i-1}, x_i) - pred(y_{j-1}, y_j)| \\ + |pred(x_i, x_{i+1}) - pred(y_j, y_{j+1})| \end{aligned}$$

The distance d_* between timestamps can be computed accordingly with the inputs of the predicted values. It is obvious that the extra predicted points don't affect the complexity of the distance which is again $O(m^2)$.

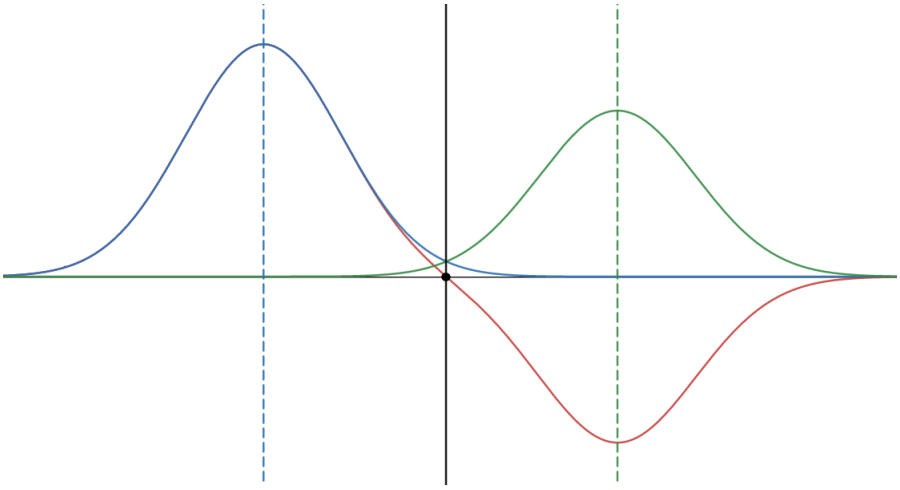


Figure 7.5: The case when $A \cdot B < 0$. The blue and green curves are the Gaussians, and the red one is the predictive curve. The reflection of the green Guassian gives us the intersection point.

As an averaging method we can use DTW Barycenter Averaging with a small modification. For every coordinate of the temporary average (at every DBA's iteration) we now compute 3 values, the new central value (the "former" inducing point), the left and the right value (the "former" predicted points). The new central value is computed by taking the barycenter of all central values of the association, created by the alignment under the new distance. The new left value is computed by taking the barycenter of all left values of the association. The right value is computed accordingly.

After the computation of the temporary average, we should ensure that the right value of a central point x_{i-1_R}, which is $pred(x_{i-1_R})$, is equal with the left value of the next central point x_{i_L}, which is $pred(x_{i_L})$. To solve this continuity problem we just take the average of the two values:

$$pred'(x_{i-1_R}) = pred'(x_{i_L}) = \frac{pred(x_{i-1_R}) + pred(x_{i_L})}{2} \tag{7.8}$$

The same procedure is followed for the timestamps. The complexity of this modified version of DBA is $O(I \cdot N \cdot m^2)$. Therefore, the total complexity of the k-means algorithm is $O(k \cdot I \cdot N \cdot m^2)$.

7.2.3 Complexity

Both distances have the same time complexity, and thus we do a single computational analysis.

In the first stage of our framework, Sparse Gaussian Process Regression is used for each time series. SGPR runs in $O(T \cdot m^2)$, and thus the first stage has a time complexity of $O(N \cdot T \cdot m^2)$.

In the second stage, k-means algorithm is applied which has a time complexity of $O(k \cdot I \cdot N \cdot m^2)$. Consequently, Sparse Gaussian Processes for Clustering framework has a total complexity of:

$$O(N \cdot T \cdot m^2 + k \cdot I \cdot N \cdot m^2) \tag{7.9}$$

It is obvious that the time complexity of our framework depends directly on the number of inducing points used. It has been confirmed over many experiments that we can approximate time series using a much smaller number of inducing points than the series length. A particular case, which acts as a strong indication of using a small number of inducing points, is that for normally distributed inputs with the Squared Exponential kernel, $m = O(logT)$ suffices (see Chapter 6). Therefore, we use $m = x \cdot logT$ inducing points for our framework, where $x \ll T$, and the total complexity becomes:

$$O(N \cdot T \cdot log^2 T + k \cdot I \cdot N \cdot log^2 T) \tag{7.10}$$

Consequently, SGPC has a time complexity an order lower than the complexity of k-means algorithm with DTW.

Chapter 8

Experimental study

In this chapter, we compare the proposed SGPC framework to the k-means algorithm with DTW applied on the whole time series data. The k-means algorithm with DTW and DBA is a state of the art method for time series shape-based clustering [36]. Unfortunately, we study experimentally only the first distance proposed (simple approach) due to an implementation issue. The existing implementation of DBA uses a majorize-minimize algorithm that converges to necessary conditions of optimality, which is better in practice than the classical DBA's implementation. The modification of this majorize-minimize algorithm for the second proposed distance is, if possible, far from trivial. The simple distance gives us very good clustering results as we show in this chapter. We believe that an implementation of the second distance with the modification of the classical DBA's algorithm will not significantly improve our results (if they improve them at all), and thus we omit such a study.

A wide range of datasets is used to evaluate the efficiency of the proposed approach, in the context of clustering. We first describe the experimental settings for the evaluation of both methods (Section 8.1), and then present the results obtained (Section 8.2). Finally, we discuss some difficulties encountered in the experimental study (Section 8.3).

8.1 Experimental settings

In this section, we provide the details about our experimental settings to make all the experiments reproducible.

Dataset	NB. CLUSTER	NB. TS.	TS. LENGTH
Beef	5	60	470
ECG200	2	200	96
Arrowhead	3	211	251
Meat	3	120	448
Plane	7	210	144
ProximalPhalanxTW	6	605	80
Fish	7	350	463
DiatomSizeReduction	4	322	345
DistalPhalanxTW	6	539	80
Trace	4	200	275
Lightning7	7	143	319
ECGFiveDays	2	884	136
SonyAIBORobotSurface	2	621	70
OSULeaf	6	206	242
Coffee	2	56	286

Table 8.1: Datasets description.

8.1.1 Data description

While it is easy to measure the performance of supervised learning algorithms, such as algorithms for classification problems, it is often hard to measure the performance of unsupervised learning algorithms, such as clustering algorithms. The reason for this, is that it is subjective what makes a clustering 'good'. The performance of a clustering depends on the goal and criteria of the clustering and may therefore differ per application. Our approach is to quantify clustering quality by using classification datasets.

For this, we use the largest public collection of class-labeled time series datasets, namely, the UCR time series collection [20]. These datasets are annotated and every sequence can belong to only one class, which, in the context of clustering, should be interpreted as the cluster where the sequence belongs. We work with 15 of these datasets, both synthetic and real, which span several different domains. Each dataset is univariate and contains from 56 to 884 sequences. The sequences in each dataset have equal length, but from one dataset to another the sequence length varies from 70 to 470. Table 8.1 indicates for each dataset: the number of clusters it includes (NB. CLUSTER), the number of instances (NB. TS.) and the time series length (TS. LENGTH).

8.1.2 Data preprocessing

It is important to decide on which preprocessing should be applied to the data. We apply scaling, since we want to cluster based on similar shapes

in the time series. Two common ways of scaling are normalization and standardization. In normalization between -1 and 1, the data is scaled into a range of $[-1, 1]$ by using the following formula:

$$X' = 2 \cdot \frac{X - X_{min}}{Xmax - Xmin} - 1 \tag{8.1}$$

In standardization, the data is scaled to have mean 0 and a standard deviation of 1. This is done by using the following formula:

$$X' = \frac{X - \mu}{\sigma} \tag{8.2}$$

where μ represents the mean and σ the standard deviation of all values in vector X. We scale time series values to the interval $[-1, 1]$ and also standardize the time-axis with mean zero and standard deviation one.

8.1.3 Platform & Implementation

We run our experiments on an Intel Core i5-9300H processor running at 2,4 GHz with 8 GB of RAM. Both approaches are implemented in Python. We use GPyTorch library [25] for implementing Sparse Gaussian Process Regression (SGPR) and Tslearn package [66] for k-means algorithm with DTW as a similarity measure and DBA as an averaging method.

8.1.4 Metrics

We compare the two approaches on both accuracy and runtime. For accuracy, we use an external index to measure the similarity of formed clusters to the externally supplied class labels (ground truth). In particular, we use Adjusted Rand Index (ARI) [86] to evaluate clustering accuracy over the fused training and test sets of each dataset. Here, we don't report standard deviations since their values are similar for both methods.

- **Rand Index (RI):** computes a similarity measure between two clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings. RI is defined as $RI = \frac{TP+TN}{TP+TN+FP+FN}$, where TP is the number of time series pairs that belong to the same class and are assigned to the same cluster, TN is the number of time series pairs that belong to different classes and are assigned to different clusters, FP is the number of time series pairs that belong to different classes but are assigned to the same cluster, and FN is the number of time series pairs that belong to the same class but are assigned to different clusters.

- **Adjusted Rand Index (ARI):** RI does not take a constant value (such as zero) for two random clustering. ARI is a corrected-for-chance version of the RI which works better than RI and many other indices. The Adjusted Rand index is thus ensured to have a value close to 0.0 for random labeling independently of the number of clusters and samples and exactly 1.0 when the clusterings are identical.

As both methods evaluated are nondeterministic, we report the average Adjusted Rand Index (ARI) over 20 runs; in every run we use a different random initialization.

For runtime, we compute CPU time utilization and report time ratios for our comparison.

8.1.5 Parameter & Initialization settings

We use $m = x \cdot logT$ inducing points for our SGPC framework, where T is the time series length and $x = 1, 2, 3, 4, 5$. We choose a constant-mean GP prior and a Matérn kernel function with parameter $\nu = 1.5$. We use a modified version of DTW with a parameter α (simple approach). We conduct our experiments with parameters $\alpha = [0, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10]$.

We use the Adam algorithm to optimize the parameters of SGPR with learning rate $lr = 0.1$ for the GP parameters (kernel + noise) and $lr = 0.1/x$ for the inducing point locations. For the initialization of inducing points locations, m equally spaced points in time are used.

All initializations used in k-means and DBA algorithms are random. The maximum number of iterations of the k-means algorithm for a single run is 50 and DBA's 100.

8.2 Experimental results

In this section, we present our experimental results. We compare the two approaches mentioned on both clustering quality (Section 8.2.1) and runtime (Section 8.2.2).

8.2.1 Clustering quality

We compare the Adjusted Rand Index (ARI) of the two methods for 15 UCR datasets. We ran experiments for the different values of the number of inducing points $m = x \cdot logT$, where $x = [1, 2, 3, 4, 5]$, and the parameter $\alpha = [0, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10]$.

Metric Adjusted Rand Index (ARI)	k-means	SGPR+k-means (m=3logT)		
Dataset		**a=1**	**worst**	**best**
Beef	0.084	**0.1**	0.083	**0.11**
ECG200	0.13	0.12	0.11	**0.2**
Arrowhead	0.11	**0.22**	0.064	**0.22**
Meat	0.47	0.44	0.44	**0.47**
Plane	0.72	0.7	0.68	**0.73**
ProximalPhalanxTW	0.41	0.35	0.33	**0.41**
Fish	0.25	0.24	0.14	0.24
DiatomSizeReduction	0.28	**0.83**	**0.59**	**0.83**
DistalPhalanxTW	0.44	**0.44**	0.34	**0.44**
Trace	0.72	**0.78**	0.65	**0.84**
Lightning7	0.22	**0.3**	0.21	**0.3**
ECGFiveDays	0.11	0.1	0.094	**0.14**
SonyAIBORobotSurface	0.26	**0.28**	**0.28**	**0.6**
OSULeaf	0.12	**0.12**	0.1	**0.15**
Coffee	0.24	0.12	0.079	0.2

Table 8.2: Adjusted Rand Index scores for $m = 3 \cdot logT$.

Here we report the experimental results for $m = 3 \cdot logT$ inducing points. Table 8.2 indicates the best and the worst ARI scores among the scores for the different values of α, and the ARI score for the constant value $\alpha = 1$. As we can see the results are very good and in some datasets, such as Diatom Size Reduction, even impressive. The bad results (worst ARI) for the datasets Arrowhead and Coffee are given by low values of α. The distance for low values of α can match many points of a time series to one point of another series. When the total points are few, this may cause unwanted results.

The results for $m = x \cdot logT$, where $x = 2$, 4 and 5 are similar with a few exceptions. These low scores (exceptions) are given by low or high values of the parameter α. The reasons that high values of the parameter can cause problems are the following. Our distance becomes a lock-step measure and also the optimizer, due to the non-convexity of the objective function, can fail to allocate the inducing points correctly. The clustering results for $x = 1$ are similar but quite unstable varying α, which is expected. To obtain both good clustering quality and low running time, we propose to use $x = 2$, 3, or 4 for the number $m = x \cdot logT$ of inducing points.

The datasets that overall are difficult for our method are Coffee, Fish and Meat. The proposed framework achieved lower ARI scores (for all values of α) on these datasets for most cases (number of inducing points). The full

SGPR+k-means (m=xlogT)	x=1		x=2		x=3		x=4		x=5	
	a=1	best	a=1	best	a=1	best	a=1	best	a=1	best
Winning performances	7	11	9	12	8	13	11	12	8	12

Table 8.3: Winning performances for different numbers of inducing points.

results can be found in Appendix A.

We report in Table 8.3 only the number of datasets that our method has better ARI score than the k-means algorithm (winning performances) when taking the best ARI score for the different numbers of α and the ARI score for $\alpha = 1$. As we can see our framework gives good clustering results.

8.2.2 Runtime

We report time ratios between the two approaches. We remind that the k-means algorithm with DTW and DBA on the whole time-series has time complexity:

$$O(k \cdot I \cdot N \cdot T^2) \tag{8.3}$$

and our framework has a total complexity of:

$$O(N \cdot T \cdot log^2 T + k \cdot I \cdot N \cdot log^2 T) \tag{8.4}$$

We can see that our framework is asymptotically faster. We let $\boldsymbol{T_k}$ be the CPU utilization time of the k-means algorithm on the whole time series, $\boldsymbol{T_a}$ be the CPU utilization time of the modelling stage (SGPR) and $\boldsymbol{T_b}$ be the CPU utilization time of the clustering stage of our framework. The first stage of our framework is a preprocessing step, and thus we report two ratios. We report the ratio $\boldsymbol{T_k}/\boldsymbol{T_b}$ comparing the clustering stages and also the ratio $\boldsymbol{T_k}/(\boldsymbol{T_a} + \boldsymbol{T_b})$ comparing the total runtime of the methods.

Table 8.4 shows that our framework is significantly faster in most cases even though the datasets are relatively small. The 4 datasets that the classical method is approximately 2 times faster than our framework have the smallest lengths among the datasets (70, 80, 80, 96).

8.3 Discussion

Here we discuss some of the problems encountered when conducting our experiments. In the following, we report the issues we faced and the proposed solutions:

Dataset	NB. TS.	TS. LENGTH	Ratio T_k/T_b	Ratio $T_k/(T_a+T_b)$
Beef	60	470	**524.9**	**38.4**
ECG200	200	96	**7**	0.48
Arrowhead	211	251	**161**	**14.1**
Meat	120	448	**148.7**	**9.7**
Plane	210	144	**26**	**1.9**
ProximalPhalanxTW	605	80	**5.1**	0.6
Fish	350	463	**229.7**	**32.3**
DiatomSizeReduction	322	345	**219.1**	**22.4**
DistalPhalanxTW	539	80	**4.7**	0.65
Trace	200	275	**148.6**	**11.9**
Lightning7	143	319	**108.7**	**15.2**
ECGFiveDays	884	136	**70.7**	**5.2**
SonyAIBORobotSurface	621	70	**5.7**	0.57
OSULeaf	206	242	**272.5**	**38.3**
Coffee	56	286	**163.4**	**11**

Table 8.4: Experimental CPU time ratios.

- **Initialization of inducing points locations:** due to the non-convexity of the SGPR'S objective function for parameter learning, we get different inducing point locations for different initializations (local optima). The solution proposed is to initialize the inducing point locations of each time series in the same way. We can either initialize the inducing points with equally spaced points in time, or initialize them randomly with the same seed.

- **Adam's learning parameter:** SGPR's objective function for the inducing points locations is sensitive to the choice of Adam's learning rate. When the learning rate is too large, we might get some inducing points outside the range of the training data on the time-axis. When it is too small, the learning becomes local in the sense that each inducing point sculpts out the approximate posterior in a small region of the input space around it. The learning rate should be selected carefully.

- **Normalizing/Standardizing the data:** GPyTorch requires all data (timestamps included) to be normalized/standardized since it expects all computed distances to be on the order of 1. That's the reason we standardize our timestamps.

Chapter 9

Conclusion

In this thesis, we provided a novel two-stage framework for shape-based time series clustering using Sparse Gaussian Process Regression. The proposed framework leads to fast, scalable and accurate clustering. Experiments conducted on 15 time series datasets show that our framework is state-of-the-art compared to a widely used existing method. We believe that using Gaussian Processes for time series clustering is a promising research direction for the future. The next step would be to design appropriate distance measures for more expressive kernel functions. More expressive kernels would allow us to model more complex time series and thus extend the clustering power of our framework.

Appendix A

More experimental results

In the following, we report the experimental results for $m = x \cdot logT$ inducing points, where $x = [1, 2, 4, 5]$. We report the best and the worst ARI scores among the scores for the different values of α, and the ARI score for the constant value $\alpha = 1$.

Metric Adjusted Rand Index (ARI)	k-means	SGPR+k-means (m=logT)		
Dataset		**a=1**	**worst**	**best**
Beef	0.084	**0.11**	**0.091**	**0.11**
ECG200	0.13	**0.16**	0.08	**0.16**
Arrowhead	0.11	**0.17**	0.063	**0.17**
Meat	0.47	0.23	0.12	**0.48**
Plane	0.72	0.67	0.44	0.68
ProximalPhalanxTW	0.41	0.4	0.27	0.4
Fish	0.25	0.22	0.11	0.22
DiatomSizeReduction	0.28	**0.32**	0.25	**0.39**
DistalPhalanxTW	0.44	0.43	0.21	**0.48**
Trace	0.72	0.7	0.49	**0.81**
Lightning7	0.22	**0.31**	0.13	**0.31**
ECGFiveDays	0.11	0.079	0.079	**0.19**
SonyAIBORobotSurface	0.26	**0.3**	0.12	**0.65**
OSULeaf	0.12	0.087	0.057	0.094
Coffee	0.24	**0.3**	0.07	**0.71**

Table A.1: Adjusted Rand Index scores for $m = logT$.

Metric Adjusted Rand Index (ARI)	k-means	SGPR+k-means (m=2logT)		
Dataset		**a=1**	**worst**	**best**
Beef	0.084	**0.097**	**0.087**	**0.12**
ECG200	0.13	0.1	0.1	**0.22**
Arrowhead	0.11	**0.15**	0.075	**0.18**
Meat	0.47	0.17	0.16	0.2
Plane	0.72	0.69	0.69	**0.77**
ProximalPhalanxTW	0.41	**0.42**	**0.42**	**0.45**
Fish	0.25	0.21	0.16	0.21
DiatomSizeReduction	0.28	**0.47**	**0.42**	**0.49**
DistalPhalanxTW	0.44	**0.44**	0.31	**0.48**
Trace	0.72	**0.76**	0.61	**0.86**
Lightning7	0.22	**0.29**	**0.23**	**0.29**
ECGFiveDays	0.11	0.088	0.088	**0.12**
SonyAIBORobotSurface	0.26	**0.57**	0.14	**0.7**
OSULeaf	0.12	**0.12**	**0.12**	**0.14**
Coffee	0.24	0.07	0.07	0.22

Table A.2: Adjusted Rand Index scores for $m = 2 \cdot logT$.

Metric Adjusted Rand Index (ARI)	k-means	SGPR+k-means (m=4logT)		
Dataset		**a=1**	**worst**	**best**
Beef	0.084	**0.096**	**0.084**	**0.12**
ECG200	0.13	0.12	0.11	**0.14**
Arrowhead	0.11	**0.21**	0.073	**0.21**
Meat	0.47	0.44	0.42	0.46
Plane	0.72	**0.73**	0.66	**0.74**
ProximalPhalanxTW	0.41	**0.43**	0.33	**0.44**
Fish	0.25	**0.25**	0.13	**0.25**
DiatomSizeReduction	0.28	**0.77**	**0.43**	**0.86**
DistalPhalanxTW	0.44	0.37	0.32	0.41
Trace	0.72	**0.81**	**0.75**	**0.89**
Lightning7	0.22	**0.28**	**0.24**	**0.3**
ECGFiveDays	0.11	**0.11**	0.1	**0.17**
SonyAIBORobotSurface	0.26	**0.26**	**0.26**	**0.5**
OSULeaf	0.12	**0.12**	0.093	**0.15**
Coffee	0.24	0.12	0.1	0.16

Table A.3: Adjusted Rand Index scores for $m = 4 \cdot logT$.

Metric Adjusted Rand Index (ARI)	k-means	SGPR+k-means (m=5logT)		
Dataset		**a=1**	**worst**	**best**
Beef	0.084	**0.1**	**0.086**	**0.13**
ECG200	0.13	0.084	0.084	**0.15**
Arrowhead	0.11	**0.22**	**0.12**	**0.22**
Meat	0.47	0.37	0.33	0.38
Plane	0.72	0.71	0.63	**0.74**
ProximalPhalanxTW	0.41	**0.41**	0.33	**0.42**
Fish	0.25	**0.26**	0.12	**0.26**
DiatomSizeReduction	0.28	**0.56**	0.11	**0.84**
DistalPhalanxTW	0.44	0.43	0.34	0.43
Trace	0.72	**0.85**	0.61	**0.89**
Lightning7	0.22	**0.3**	**0.22**	**0.3**
ECGFiveDays	0.11	0.1	0.1	**0.17**
SonyAIBORobotSurface	0.26	0.25	0.25	**0.48**
OSULeaf	0.12	**0.12**	0.096	**0.15**
Coffee	0.24	0.16	0.11	0.19

Table A.4: Adjusted Rand Index scores for $m = 5 \cdot logT$.

Bibliography

[1] *DTW-Based Motion Comparison and Retrieval*, pages 211–226. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, ISBN 978-3-540-74048-3. https://doi.org/10.1007/978-3-540-74048-3_10.

[2] *Dynamic Time Warping*, pages 69–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, ISBN 978-3-540-74048-3. https://doi.org/10.1007/978-3-540-74048-3_4.

[3] Aach, J. and G. Church: *Aligning gene expression time series with time warping algorithms.* Bioinformatics, 17 6:495–508, 2001.

[4] Abdulla, W., D. Chow, and G. Sin: *Cross-words reference template for dtw-based speech recognition systems.* TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region, 4:1576–1579 Vol.4, 2003.

[5] Abraham, C., PA Cornillon, E. Matzner-Løber, and Nicolas Molinari: *Unsupervised curve clustering using b-splines.* Scandinavian Journal of Statistics, 30:581 – 595, September 2003.

[6] Aggarwal, Charu C. and Chandan K. Reddy: *Data clustering.* https://www.oreilly.com/library/view/data-clustering/9781466558229/K15510_C015.xhtml.

[7] Aghabozorgi, Saeed, Ali Seyed Shirkhorshidi, and Teh Ying Wah: *Time-series clustering – a decade review.* Information Systems, 53:16–38, 2015, ISSN 0306-4379. https://www.sciencedirect.com/science/article/pii/S0306437915000733.

[8] Aghabozorgi, Saeed and Ying Wah Teh: *Stock market co-movement assessment using a three-phase clustering method.* Expert Systems with Applications, 41(4, Part 1):1301–1314, 2014, ISSN 0957-4174. https://www.sciencedirect.com/science/article/pii/S0957417413006404.

[9] Aghabozorgi, Sr, Ali Seyed Shirkhorshidi, and Teh Wah: *Time-series clustering - a decade review.* Information Systems, 53, May 2015.

[10] Agrawal, Rakesh, Christos Faloutsos, and Arun Swami: *Efficient similarity search in sequence databases.* Volume 730, pages 69–84, January 1993.

[11] Alles, Irina: *Time Series Clustering in the Field of Agronomy.* PhD thesis, 2013.

[12] Bagnall, Anthony and gj Janacek: *Clustering time series with clipped data.* Machine Learning, 58:151–178, February 2005.

[13] Billheimer, D.: *Functional data analysis, 2nd edition edited by j. o. ramsay and b. w. silverman.* Biometrics, 63:300–301, March 2007.

[14] Boyle, Phillip: *Gaussian Processes for Regression and Optimisation.* PhD thesis, 2007.

[15] Bui, T. and Richard Turner: *Tree-structured gaussian process approximations.* Advances in Neural Information Processing Systems, 3:2213–2221, January 2014.

[16] Burt, David, Carl Edward Rasmussen, and Mark Van Der Wilk: *Rates of convergence for sparse variational Gaussian process regression.* In Chaudhuri, Kamalika and Ruslan Salakhutdinov (editors): *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 862–871. PMLR, 09–15 Jun 2019. `http://proceedings.mlr.press/v97/burt19a.html`.

[17] Campbell, Joseph P.: *Speaker Recognition*, pages 165–189. Springer US, Boston, MA, 1996, ISBN 978-0-306-47044-8. `https://doi.org/10.1007/0-306-47044-6_8`.

[18] Cassisi, Carmelo, Placido Montalto, Marco Aliotta, Andrea Cannata, and Alfredo Pulvirenti: *Similarity Measures and Dimensionality Reduction Techniques for Time Series Data Mining*. September 2012, ISBN 978-953-51-0748-4.

[19] Chandola, V.: *Tr 09-004 detecting anomalies in a time series database.* 2009.

[20] Dau, Hoang Anh, Eamonn Keogh, Kaveh Kamgar, Chin Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana,

Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML: *The ucr time series classification archive*, October 2018. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[21] Deng, Charlotte: *Dynamic time warping for predictive modeling using sensor data*, May 2018. https://medium.com/@hdeng727/dynamic-time-warping-for-predictive-modeling-us-sensor-data-39021aeccc80.

[22] Duvenaud, David: *The kernel cookbook:*. http://www.cs.toronto.edu/~duvenaud/cookbook/index.html.

[23] Esling, Philippe and Carlos Agon: *Time-series data mining*. ACM Computing Surveys (CSUR), 45:12, November 2012.

[24] Faloutsos, Christos, M. Ranganathan, and Yannis Manolopoulos: *Fast subsequence matching in time-series databases*. ACM SIGMOD Record, 23, June 2000.

[25] Gardner, Jacob R, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson: *Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration*. In *Advances in Neural Information Processing Systems*, 2018.

[26] Gibbs, M.: *Bayesian Gaussian Processes for Classification and Regression*. PhD thesis, 1997.

[27] Gold, Omer and Micha Sharir: *Dynamic time warping and geometric edit distance: Breaking the quadratic barrier*. ACM Trans. Algorithms, 14(4), August 2018, ISSN 1549-6325. https://doi.org/10.1145/3230734.

[28] Gullo, Francesco, Giovanni Ponti, Andrea Tagarelli, and Sergio Greco: *A time series representation model for accurate and fast similarity detection*. Pattern Recognition, 42(11):2998–3014, 2009, ISSN 0031-3203. https://www.sciencedirect.com/science/article/pii/S0031320309001290.

[29] Gullo, Francesco, Giovanni Ponti, Andrea Tagarelli, Giuseppe Tradigo, and Pierangelo Veltri: *A time series approach for clustering mass spectrometry data*. Journal of Computational Science, 3(5):344–355, 2012, ISSN 1877-7503. https://www.sciencedirect.com/science/article/pii/S1877750311000627, Advanced Computing Solutions for Health Care and Medicine.

[30] Gupta, L., Dennis Molfese, Ravi Tammana, and Panagiotis Simos: *Nonlinear alignment and averaging for estimating the evoked potential.* IEEE transactions on bio-medical engineering, 43:348–56, May 1996.

[31] Guralnik, Valery and Jaideep Srivastava: *Event detection from time series data.* In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, page 33–42, New York, NY, USA, 1999. Association for Computing Machinery, ISBN 1581131437. https://doi.org/10.1145/312129.312190.

[32] Hautamäki, Ville, Pekka Nykänen, and P. Fränti: *Time-series clustering by approximate prototypes.* In *ICPR*, 2008.

[33] Hensman, James, Nicolò Fusi, and Neil D. Lawrence: *Gaussian processes for big data.* In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, page 282–290, Arlington, Virginia, USA, 2013. AUAI Press.

[34] Iorio, Carmela, Gianluca Frasso, Antonio D'Ambrosio, and Roberta Siciliano: *Parsimonious time series clustering using p-splines.* Expert Systems with Applications, 52:26–38, 2016, ISSN 0957-4174. https://www.sciencedirect.com/science/article/pii/S0957417416000063.

[35] Itakura, F.: *Minimum prediction residual principle applied to speech recognition.* IEEE Transactions on Acoustics, Speech, and Signal Processing, 23(1):67–72, 1975.

[36] Javed, Ali, Byung Suk Lee, and Donna M. Rizzo: *A benchmark study on time series clustering.* Machine Learning with Applications, 1:100001, 2020, ISSN 2666-8270. https://www.sciencedirect.com/science/article/pii/S2666827020300013.

[37] Kaufmann, Leonard and Peter Rousseeuw: *Clustering by means of medoids.* Data Analysis based on the L1-Norm and Related Methods, pages 405–416, January 1987.

[38] Keogh, Eamonn, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra: *Locally adaptive dimensionality reduction for indexing large time series databases.* In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD '01, page 151–162, New York, NY, USA, 2001. Association for Computing Machinery, ISBN 1581133324. https://doi.org/10.1145/375663.375680.

[39] Keogh, Eamonn, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra: *Dimensionality reduction for fast similarity search in large time series databases.* Knowledge and Information Systems, 3, January 2002.

[40] Keogh, Eamonn and Shruti Kasetty: *Kasetty, s.: On the need for time series data mining benchmarks: A survey and empirical demonstration. data mining and knowledge discovery 7(4), 349-371.* Data Mining and Knowledge Discovery, 7:349–371, January 2003.

[41] Kingma, Diederik P. and Jimmy Ba: *Adam: A method for stochastic optimization*, 2017.

[42] Lin, Jessica, Eamonn Keogh, Stefano Lonardi, and Bill Chiu: *A symbolic representation of time series, with implications for streaming algorithms.* In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '03, page 2–11, New York, NY, USA, 2003. Association for Computing Machinery, ISBN 9781450374224. https://doi.org/10.1145/882082.882086.

[43] MacKay, D. J. C.: *Bayesian interpolation.* Neural Computation, 4(3):415–447, 1992. http://www.inference.phy.cam.ac.uk/mackay/PhD.html.

[44] Michelen, Rory: *Using b-splines and k-means to cluster time series*, Jun 2020. https://towardsdatascience.com/using-b-splines-and-k-means-to-cluster-time-series-16468f588ea6.

[45] Morel, Marion, Catherine Achard, Richard Kulpa, and Séverine Dubuisson: *Time-series averaging using constrained dynamic time warping with tolerance.* Pattern Recognition, 74, August 2017.

[46] Müller, Meinard: *Information Retrieval for Music and Motion*. January 2007, ISBN 3540740473.

[47] Niennattrakul, Vit and Chotirat Ratanamahatana: *Shape averaging under time warping.* pages 626 – 629, June 2009.

[48] Niennattrakul, Vit and Chotirat Ann Ratanamahatana: *On clustering multimedia time series data using k-means and dynamic time warping.* In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, pages 733–738, 2007.

[49] Özkoç, Esma Ergüner: *Clustering of time-series data.* 2020.

[50] Panuccio, Antonello, Manuele Bicego, and Vittorio Murino: *A hidden markov model-based approach to sequential data clustering*. pages 734–742, August 2002, ISBN 978-3-540-44011-6.

[51] Petitjean, François and Pierre Gançarski: *Summarizing a set of time series by averaging: From steiner sequence to compact multiple alignment*. Theoretical Computer Science, 414(1):76–91, 2012, ISSN 0304-3975. https://www.sciencedirect.com/science/article/pii/S030439751100822X.

[52] Petitjean, François, Alain Ketterlin, and Pierre Gançarski: *A global averaging method for dynamic time warping, with applications to clustering*. Pattern Recognition, 44(3):678–693, 2011, ISSN 0031-3203. https://www.sciencedirect.com/science/article/pii/S003132031000453X.

[53] Quinonero-Candela, JQ and Carl Rasmussen: *A unifying view of sparse approximate gaussian process regression*. Journal of Machine Learning Research, v.6, 1935-1959 (2005), 6, December 2005.

[54] Quiñonero-Candela, Joaquin, Carl Ramussen, and Christopher Williams: *Approximation methods for gaussian process regression*. April 2007.

[55] Rasmussen, Carl Edward and Christopher K. I. Williams: *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005, ISBN 026218253X.

[56] Ratanamahatana, Chotirat, Jessica Lin, Dimitrios Gunopulos, Eamonn Keogh, Michalis Vlachos, and Gautam Das: *Mining Time Series Data*, pages 1049–1077. July 2010.

[57] Sakoe, H. and S. Chiba: *Dynamic programming algorithm optimization for spoken word recognition*. IEEE Transactions on Acoustics, Speech, and Signal Processing, 26(1):43–49, 1978.

[58] Salvador, S. and P. Chan: *Fastdtw: Toward accurate dynamic time warping in linear time and space*. 2004.

[59] Sardá-Espinosa, Alexis: *Comparing time-series clustering algorithms in r using the dtwclust package*. January 2018.

[60] Schultz, David and Brijnesh Jain: *Nonsmooth analysis and subgradient methods for averaging in dynamic time warping spaces*, 2017.

[61] Senin, Pavel: *Dynamic time warping algorithm review*. January 2009.

[62] Smola, Alex J. and Peter Bartlett: *Sparse greedy gaussian process regression.* In *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press, 2001.

[63] Snelson, Edward and Zoubin Ghahramani: *Sparse gaussian processes using pseudo-inputs.* In *Advances in Neural Information Processing Systems 18*, pages 1257–1264. MIT press, 2006.

[64] Soheily Khah, Said, Ahlame Douzal, and Eric Gaussier: *A comparison of progressive and iterative centroid estimation approaches under time warp.* Volume 9785, pages 144–156, August 2016, ISBN 978-3-319-44411-6.

[65] Subhani, Numanul, Luis Rueda, Alioune Ngom, and Conrad J. Burden: *Multiple gene expression profile alignment for microarray time-series data clustering.* Bioinformatics, 26(18):2281–2288, July 2010, ISSN 1367-4803. https://doi.org/10.1093/bioinformatics/btq422.

[66] Tavenard, Romain, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods: *Tslearn, a machine learning toolkit for time series data.* Journal of Machine Learning Research, 21(118):1–6, 2020. http://jmlr.org/papers/v21/20-091.html.

[67] Titsias, Michalis: *Variational model selection for sparse gaussian process regression.* 2008.

[68] Titsias, Michalis: *Variational learning of inducing variables in sparse gaussian processes.* In Dyk, David van and Max Welling (editors): *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 567–574, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR. http://proceedings.mlr.press/v5/titsias09a.html.

[69] Tralie, Christopher and Elizabeth Dempsey: *Exact, parallelizable dynamic time warping alignment with linear memory*, 2020.

[70] Vial, Jérôme, Hicham Noçairi, Patrick Sassiat, Sreedhar Mallipatu, Guillaume Cognon, Didier Thiébaut, Béatrice Teillet, and Douglas N. Rutledge: *Combination of dynamic time warping and multivariate analysis for the comparison of comprehensive two-dimensional gas chromatograms: Application to plant extracts.* Journal of Chromatography A, 1216(14):2866–2872, 2009,

ISSN 0021-9673. https://www.sciencedirect.com/science/article/pii/S0021967308015471, 32nd International Symposium on Capillary Chromatography and 5th GCxGC Symposium.

[71] Volny, Petr, David Novak, and Pavel Zezula: *Employing subsequence matching in audio data processing.* April 2012.

[72] Wang, L. and Tao Jiang: *On the complexity of multiple sequence alignment.* Journal of computational biology : a journal of computational molecular cell biology, 1 4:337–48, 1994.

[73] Wang, Xiaoyue, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh: *Experimental comparison of representation methods and distance measures for time series data.* Data Mining and Knowledge Discovery, 26, December 2010.

[74] Warren Liao, T.: *Clustering of time series data—a survey.* Pattern Recognition, 38(11):1857–1874, 2005, ISSN 0031-3203. https://www.sciencedirect.com/science/article/pii/S0031320305001305.

[75] Wikipedia contributors: *Limited-memory bfgs — Wikipedia, the free encyclopedia*, 2020. https://en.wikipedia.org/w/index.php?title=Limited-memory_BFGS&oldid=996198021, [Online; accessed 4-June-2021].

[76] Wikipedia contributors: *Polynomial regression — Wikipedia, the free encyclopedia*, 2020. https://en.wikipedia.org/w/index.php?title=Polynomial_regression&oldid=996324225, [Online; accessed 3-June-2021].

[77] Wikipedia contributors: *Autoregressive integrated moving average — Wikipedia, the free encyclopedia*, 2021. https://en.wikipedia.org/w/index.php?title=Autoregressive_integrated_moving_average&oldid=1020508954, [Online; accessed 18-June-2021].

[78] Wikipedia contributors: *Autoregressive–moving-average model — Wikipedia, the free encyclopedia*, 2021. https://en.wikipedia.org/w/index.php?title=Autoregressive%E2%80%93moving-average_model&oldid=997824723, [Online; accessed 18-June-2021].

[79] Wikipedia contributors: *Cluster analysis — Wikipedia, the free encyclopedia*, 2021. https://en.wikipedia.org/w/index.php?title=

Cluster_analysis&oldid=1026425559, [Online; accessed 17-June-2021].

[80] Wikipedia contributors: *Discrete wavelet transform — Wikipedia, the free encyclopedia*, 2021. https://en.wikipedia.org/w/index.php?title=Discrete_wavelet_transform&oldid=1026374626, [Online; accessed 18-June-2021].

[81] Wikipedia contributors: *Dynamic time warping — Wikipedia, the free encyclopedia*, 2021. https://en.wikipedia.org/w/index.php?title=Dynamic_time_warping&oldid=1002338461, [Online; accessed 26-May-2021].

[82] Wikipedia contributors: *Gaussian process — Wikipedia, the free encyclopedia*, 2021. https://en.wikipedia.org/w/index.php?title=Gaussian_process&oldid=1026424395, [Online; accessed 2-June-2021].

[83] Wikipedia contributors: *Kernel method — Wikipedia, the free encyclopedia*, 2021. https://en.wikipedia.org/w/index.php?title=Kernel_method&oldid=1012624126, [Online; accessed 2-June-2021].

[84] Wikipedia contributors: *Multivariate normal distribution — Wikipedia, the free encyclopedia*, 2021. https://en.wikipedia.org/w/index.php?title=Multivariate_normal_distribution&oldid=1025292039, [Online; accessed 2-June-2021].

[85] Wikipedia contributors: *Radial basis function kernel — Wikipedia, the free encyclopedia*, 2021. https://en.wikipedia.org/w/index.php?title=Radial_basis_function_kernel&oldid=1010095405, [Online; accessed 2-June-2021].

[86] Wikipedia contributors: *Rand index — Wikipedia, the free encyclopedia*, 2021. https://en.wikipedia.org/w/index.php?title=Rand_index&oldid=1027033499, [Online; accessed 21-June-2021].

[87] Wikipedia contributors: *Regression analysis — Wikipedia, the free encyclopedia*, 2021. https://en.wikipedia.org/w/index.php?title=Regression_analysis&oldid=1018847859, [Online; accessed 3-June-2021].

[88] Wikipedia contributors: *Time series — Wikipedia, the free encyclopedia*, 2021. https://en.wikipedia.org/w/index.php?title=Time_series&oldid=1028371500, [Online; accessed 28-June-2021].

[89] Wilk, Mark van der: *Sparse Gaussian Process Approximations and Applications*. PhD thesis, 2018.

[90] Wilson, Andrew and Hannes Nickisch: *Kernel interpolation for scalable structured gaussian processes (kiss-gp)*. March 2015.

[91] Wu, Xindong, Vipin Kumar, Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, G. Mclachlan, Shu Kay Angus Ng, Bing Liu, Philip Yu, Zhi Hua Zhou, Michael Steinbach, David Hand, and Dan Steinberg: *Top 10 algorithms in data mining*. Knowledge and Information Systems, 14, December 2007.

[92] Xing, Zheng zheng, Jian Pei, and Eamonn Keogh: *A brief survey on sequence classification*. SIGKDD Explorations, 12:40–48, November 2010.

[93] Yi, Wei: *Sparse and variational gaussian process-what to do when data is large*, May 2021. `https://towardsdatascience.com/sparse-and-variational-gaussian-process-what-to-do-when-data-is-large-2d3959f430e7`.