TDK-dolgozat

Marussy Kristóf
2012.

MŰEGYETEM 1782

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Számítástudományi és Információelméleti Tanszék

Egy új módszer az idősorok pontosabb semi-supervised osztályzására

TDK-dolgozat

Készítette:
Marussy Kristóf
I. évfolyam

Konzulens:
Dr. Buza Krisztián,
egyetemi adjunktus

2012.

MŰEGYETEM 1782

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Computer Science and Information Theory

A new approach for more accurate semi-supervised time-series classification

TDK-paper

Written by:
Kristóf Marussy
year I.

Advisor:
Dr. Krisztián Buza,
lecturer

2012.

Contents

Contents 1

Abstract 3

Notations 5

1 Introduction 7

2 Background 9

- 2.1 Semi-supervised learning methods . . . 9
  - 2.1.1 Supervised and unsupervised tasks . . . 9
  - 2.1.2 Semi-supervised classification . . . 9
  - 2.1.3 Semi-supervised clustering . . . 14
- 2.2 Time-series data mining . . . 17
  - 2.2.1 Distance functions for time series . . . 17
  - 2.2.2 Hubness in time series databases . . . 20
  - 2.2.3 Semi-supervised time-series classification . . . 21

3 My approach: constrained SLINK with DTW 25

- 3.1 Description of the algorithm . . . 25
- 3.2 A graph theoretic view on time series nearest neighbour self-training . . . 25
  - 3.2.1 From self-training to cluster-and-label . . . 26
- 3.3 Properties of constrained SLINK . . . 26
  - 3.3.1 Similarity to the 1-nearest neighbour graph . . . 26
  - 3.3.2 Interaction with hubs . . . 27
  - 3.3.3 Transitive hubs . . . 27

4 Experimental evaluation 29

- 4.1 Protocol of evaluation . . . 29
- 4.2 Time-series databases . . . 29
- 4.3 Results . . . 30

5 Conclusions 35

- 5.1 Further work and open questions . . . 35

References 37

Rövid összefoglaló Az idősorok időben egymás után mért skalárok vagy vektorok sorozatai. Számos alkalmazási területen fordulnak elő olyan feladatok, melyek idősorokkal kapcsolatos problémákra vezethetőek vissza. A toll végének helyzete a papíron, ha egymásutáni időpillanatokban rögzítjük, jellemezhet egy leírt szót vagy aláírást. Megfelelő, kesztyűként viselhető szenzorok segítségével hasonlóan kódolhatóak a jelnyelv jelei is. Az orvosi alkalmazásokban az idősorok az agyhullámok (EEG) és EKG görbék természetes reprezentációi.

Az adatbányászatban supervised és unsupervised problémák fordulnak elő. A supervised feladatokban a tanuló algoritmus bemenetéül szolgáló adatok osztálycímkékkel vannak ellátva. Ezt a bemeneti halmazt használjuk fel egy modell tanítására, mellyel később előrejelzési és felismerési feladatokat oldhatunk meg. Az unsupervised tanulás (klaszterezés) esetén az osztálycímkék hiányoznak, vagy a tanulás fázisában nem érhetőek el. A semi-supervised protokoll esetében a tanító halmaznak csak egy – általában kicsiny – része címkézett. A címkézett adatok önmagukban nem feltétlenül jellemzik jól a lehetséges bemeneteket, így a jelen levő címkézetlen adatokat is fel kell használni a jó felismerő rendszer készítéséhez.

A dolgozatomban egy új semi-supervised tanulási módszert javasolok az idősorok osztályozására, mely az instance-based tanuláson és hierarchikus klaszterezésen alapul. A választás az instance-based paradigmára széleskörű alkalmazhatósága miatt esett: mindössze az idősorok páronkénti távolságainak ismeretét igényli. A távolságfüggvénynek a dynamic time warping (DTW) algoritmust választottam. Erről a távolságfüggvényről igazolt, hogy képes idősorok gyors és pontos supervised osztályzására (Ding et al., 2008; Keogh és C. A. Ratanamahatana, 2005).

Hogy munkám reprodukálását megkönnyítsem, az algoritmust 45, publikusan elérhető adatbázison (Keogh et al., 2006b) próbáltam ki. A kísérletekben módszeremet az egyik legelterjedtebb idősor felismerő rendszerrel hasonlítottam össze. Az eredmények szerint módszerem szignifikánsan pontosabban osztályozta az adatbázisok jelentős hányadát a state-of-the-art felismerő rendszerhez képest. Az algoritmus Java kódját nyilvánosan elérhetővé fogom tenni.

Abstract Time series are temporal sequences of scalar- or vector-valued measurements. Tasks which can be formalised as problems concerning time series appear in numerous domains. For example, a handwritten word or signature may be considered as a sequence of two dimensional vectors corresponding the positions of the pen tip on the paper in consecutive moments of time. Using a sensor apparatus worn as a glove, sign language signs can be encoded similarly. In medicine, time series are a natural representation of brainwaves (EEG) and electrocardiograph (ECG) curves.

There are supervised and unsupervised problems in machine learning. In supervised tasks, the training set (input data of the learning algorithm) is augmented with class labels. Based on this training set, a model is constructed that can be used for a prediction or recognition problem afterwards. The process of constructing the model is called training. In unsupervised learning, e.g. clustering, the class labels are absent (or unused in the training stage). In the semi-supervised learning protocol, only a—usually small—fraction of the training set is labeled. The labeled instances may not represent the problem domain well, thus the structure of the unlabeled training instances must be also exploited.

In this work, I propose a new semi-supervised learning method for time-series. My approach is based on the instance-based learning paradigm and hierarchical clustering. The instance-based approach was selected because of its versatility: it only depends on a pairwise dissimilarity measure between instances. As dissimilarity measure, I chose dynamic time warping (DTW), which was shown to have high accuracy and performance in time-series classification (Ding et al., 2008; Keogh and C. A. Ratanamahatana, 2005).

In order to assist reproduction of my work, I evaluated the algorithm on 45 publicly available data sets (Keogh et al., 2006b) from various real-word domains. In the experiments, I compared my approach against one of the most prominent state-of-the-art time-series classifiers. The results show that my approach significantly outperformed the state-of-the-art time-series classifier in terms of classification accuracy on a large fraction of the data sets To further support reproducibility, I will publish the Java-code of my algorithm.

Notations

n	number of instances
l	number of labeled instances
x_i	an instance
y_i	class label of instance x_i
$\hat{y}_i$	predicted class label of instance x_i
$X = \{x_i\}_{i=1}^{n}$	set of all instances
$L = \{(x_i, y_i)\}_{i=1}^{l}$	set of all labeled instances
$U = \{x_i\}_{i=l+1}^{n}$	set of all unlabeled instances
$\|x\|$	length of time series x
$x[t]$	value of time series x at time moment t
$x^{(i)}$	ith channel of vector-valued time series x
$\mathbb{E}_P[\xi]$	expected value of ξ, with respect to distribution P
S_p	skewness (standardised third moment) of p
$[\![\phi]\!] = \begin{cases} 1 & \text{if } \phi \text{ is true,} \\ 0 & \text{if } \phi \text{ is false} \end{cases}$	truth value of ϕ
$V(G)$	vertex set of graph G
$E(G)$	edge set of graph G
$(u \to v)$	directed edge between u and v
$(u \rightsquigarrow v)$	(directed) path between u and v
G^T	transitive closure of graph G
$g_N^k(x_i)$	k-occurrence score of k
$g_G^k(x_i)$	good k-occurrence score of k
$g_{TN}^k(x_i)$	transitive k-occurrence score of k
$g_{TG}^k(x_i)$	transitive good k-occurrence score of k
$\widetilde{BN}_k$	normalised total bad k-hubness
$\widetilde{TBN}_k$	normalised total bad transitive k-hubness

1 Introduction

As sensor networks and other intelligent systems become more and more ubiquitous, the amount of data available grows dramatically day by day.

Large volumes of raw data by themselves are nearly meaningless. Only with the aid of data mining and machine learning can meaningful *information* be extracted from databases. An important area of machine learning is *classification*, which aims to associate class labels with instances from databases.

Classification is a *supervised* task. Classifiers need sizeable *training sets*, which contain instances labeled with class labels. This labeling requires significant effort of human *experts*.

In domains with a long history of data collection and data mining applications, vast labeled databases are readily accessible. However, there are also potential new domains in which labeled datasets are scarce and expensive.

Semi-supervised learning tries to take advantage of both labeled and unlabeled datasets, which could reduce amount of labeled data needed for an accurate recognition system. This means semi-supervised machine learning techniques can be adapted to domains where large labeled datasets are currently unavailable.

Many applications are interested in parameters that change with time. Time-series are a natural representation of such data. Therefore, semi-supervised learning for time-series is of great utility. It is quite surprising that the number of semi-supervised learning methods for time series are relatively low compared to semi-supervised learning methods for other data types.

2 Background

2.1 Semi-supervised learning methods

2.1.1 Supervised and unsupervised tasks

In machine learning, tasks are either *supervised* or *unsupervised*. Learning algorithms build a system that classifies or clusters data or predicts unknown variables.

The problem is said to be unsupervised if the training set contains *unlabeled* data. On the other hand, if the training set is enhanced with *labels* —desired predictions— the problem is said to be a supervised one.

While unlabeled data is abundant in most domains, labeled data is scarce and expensive, because labeling data requires considerable effort of a human expert. Therefore, it is of great interest to reduce the amount of labeled data needed while maintaining accuracy.

Learning algorithms that take advantage of both labeled and unlabeled data are said to be *semi-supervised*.

Comprehensive literature reviews on the topic include Seeger (2001) and X. Zhu (2006).

2.1.2 Semi-supervised classification

The training set for *classification* problems consists of pairs *instances* and *class labels* $X = \{x_i, y_i\}_{i=1}^{n}$. Given an instance x we wish to accurately predict its class label $\hat{y}$.

More formally, a *classifier* function is sought which minimises

$$c_{\text{true}} = \mathbb{E}_{(x,y)\sim P}[c(x, y, f(x))], \tag{2.1}$$

where the expectation is taken over all possible x from the joint distribution P of instances and their true class labels, not only those in the training set. In most problems we do not know the true distribution P and can only estimate it. The labeled data X, which is a set of samples from P, can be seen as aid—supervision—from a teacher in the construction of f,

When the class labels are discrete, a commonly used cost function is the 0–1 loss

$$c(x, y, \hat{y}) = [\![y \neq \hat{y}]\!], \tag{2.2}$$

i.e. the cost of misclassification is 1 and the cost of correct classification is 0.

In semi-supervised classification problems unlabeled training data $U = \{x_i\}_{i=l+1}^{n}$ is available as well as labeled training data $L = \{(x_i, y_i)\}_{i=1}^{l}$. Usually, the number of labeled instances l is much smaller than n because of the scarcity of labeled data.

If the labeled data L represented the distribution P well we could simply ignore the unlabeled data U and perform supervised learning by choosing f from a family of classifiers F such that the *training set error*

$$c_L = \frac{1}{l}\sum_{i=1}^{l} c(x_i, y_i, f(x_i)) \approx c_{\text{true}} \tag{2.3}$$

is minimised. In such situations, applying semi-supervised learning instead of supervised learning would yield no improvement.

However, in many practical problems the training set error c_L—even if there is no overfitting—could be drastically different from the true error c_{true}. In this case, it may be possible to better approximate P by exploiting the structure of both L and U.

Assumptions about unlabeled data

In order to learn from unlabeled data we must make some assumptions about it structure. Unlabeled data is indeed beneficial if our model assumptions match the true structure of the data

The fact that unlabeled data can reduce the accuracy of a learner has been observed by many researchers (Cozman et al., 2003; Elworthy, 1994; Singh et al., 2008). Deciding whether a given model is correct is generally very difficult because it would require a large amount a labeled data.

Generative models

Semi-supervised generative models make the assumption that data is generated by some parametric model with joint instance and class label probability

$$p(x, y) = p(y)p(x|y) \tag{2.4}$$

where $p(x|y)$ is an identifiable mixture distribution.

More specifically, the optimal parameters $\hat{\theta}$ are determined such that the likelihood—or equivalently, the log-likelihood—of labeled data L and unlabeled data U is maximised, i.e.

$$\hat{\theta} = \arg\max_{\theta} \left[\sum_{i=1}^{l} \log p_\theta(x_i, y_i) + \sum_{i=l+1}^{n} \log p_\theta(x_i) \right]. \tag{2.5}$$

The optimization can be performed by the expectation-maximization (EM) algorithm (Dempster et al., 1977). One must take care choosing the initial configuration because EM is prone to stopping at local maxima.

If—given $p(x)$—the model components $p(x|y)$ can be uniquely identified the model assumption is correct and unlabeled data can improve accuracy (Castelli and Cover, 1996). By contrast, Cozman et al. (2003) give a formal derivation why model quality deteriorates when we add unlabeled data to a mixture model with incorrect assumptions.

Cluster-and-label

Some generative models, e.g. generative models with Gaussian components, can be regarded as models that first perform *probabilistic*—*fuzzy*—clustering on all data $\{x_i\}_{i=1}^n$. By probabilistic clustering, we refer to a clustering in which for each instance a discrete probability distribution $p_i(c)$ describes the probability of instance x_i belonging to cluster c. As a second step, the aforementioned semi-supervised generative models form a mapping between clusters and classes with the aid of labeled data L, such that the probability of instance x_i belonging to class y is $p(y|x_i) = p_i(c(y))$, where $c(y)$ is the cluster corresponding to class y. The existence of such bijection is postulated by the model assumption.

This scheme can be extended to non-probabilistic clustering, too.

1. Unsupervised or semi-supervised (see subsection 2.1.3) clustering is performed on $\{x_i\}_{i=1}^n$.

```
SELF-TRAINING$(L, U)$
 1  $L_0 = L$
 2  $U_0 = U$
 3  $t = 0$
 4  repeat
 5      $M = \text{SUPERVISED-LEARNING}(L_t)$
 6      $x_{best} = \arg\max_{x \in U_t} \text{CERTAINTY}(M, x)$
 7      $\hat{y} = \text{CLASSIFY}(M, x_{best})$
 8      $L_{t+1} = L_t \cup \{(x_{best}, \hat{y})\}$
 9      $U_{t+1} = U_t \setminus \{x_{best}\}$
10      $t = t + 1$
11  until $|U_t| == 0$
12  return $M$
```

(a) Simple self-training algorithm.

```
YAROWSKY$(L, U, \zeta)$
 1  $L_0 = L$
 2  $t = 0$
 3  repeat
 4      $M = \text{SUPERVISED-LEARNING}(L_t)$
 5      $L_{t+1} = L$
 6      for $i = l + 1$ to $n$
 7          if $\text{CERTAINTY}(M, x_i) > \zeta$
 8              $\hat{y} = \text{CLASSIFY}(M, x_i)$
 9              $L_{t+1} = L_{t+1} \cup \{(x_i, \hat{y})\}$
10      $t = t + 1$
11  until $L_{t-1} == L_t$
12  return $M$
```

(b) Yarowsky's algorithm.

Listing 2.1: Self-training algorithms.

2. Clusters are mapped to classes by some algorithm. A possible mapping can be constructed by majority vote, i.e. each cluster gets mapped to class of which the most labeled instances it contains.

For example, Dara et al. (2002)—with self-organizing maps (SOM)—and Demiriz et al. (1999)—with another genetic algorithm—applied this principle for semi-supervised classification.

Due to the algorithmic nature of cluster-and-label its anlysis may be difficult. However, it can perform well if particular clustering algorithm captures the true structure of the data.

Self-training

Self-training is perhaps the most commonly used semi-supervised algorithm. It is especially prevalent in the natural language processing community where it is mostly used in word sense disambiguation tasks.

Self-training is 'wrapper' method around a supervised classifier, i.e. one may use self-training to enhance nearly any existing recognition system. The only requirement is that the *base classifier* is capable of reporting a certainty value for each classification it makes. Thus the model assumption states that the certainty output of the supervised classifier is a good approximation of the true uncertainty. For probabilistic classifiers, the expected probability of correct classification $p(\hat{y}|x)$ is a natural candidate for being such certainty measure.

In each iteration t of self-training, the base classifier is trained on the labeled set L_t from the previous iteration, resulting in model M. The labeled set L_{t+1} is grown with some examples that are classified with high certainty. It is also possible that examples that some examples that are not part of the original labeled training set L get removed from L_{t+1}. Hypothesised class labels of instances which were initially unlabeled may also change. The process is repeated until some *stopping criterion* is satisfied, e.g. the labeled set does not change any more or no example is left unlabeled. '/

The simplest self-training algorithm grows the labeled set with the most certain instances until it runs out of unlabeled examples. This approach, which is occasionally called *label propagation*, is illustrated by listing 2.1(a).

Yarowsky's algorithm (Yarowsky, 1992), which is widely used for word sense disambiguation, is a more elaborate form of self-training. The number instances which are placed in the training set L_{t+1} is controlled by parameter ζ. An instance must have a certainty greater that ζ —or be initially labeled—to be placed and retained in L_{t+1}. Moreover, hypothesised class labels of initially unlabeled instances may change during training. Listing 2.1(b) shows this process.

General analysis of a wrapper method like self-training is extremely difficult. However, there are some result on analysis of convergence for some specific base learners (Abney, 2004; Culp and Michailidis, 2007; Haffari and Sarkar, 2007).

Instance-based self-training Instance-based learning is a family of learning algorithms which classify instances directly based on the training set L, not by statistical models derived from it.

A dissimilarity function $d(x_1, x_2)$ is defined on the data to describe the *dissimilarity*—or distance—of instances. Classification output for a new instance x is constructed from instances in the labeled set L and dissimilarity values $(d(x, x_i))_{i=1}^{l}$.

Instance-based learning methods may be sensitive to noise in the training set and the choice of dissimilarity function. Moreover, storing a large number of instance L may also be problematic. Despite these limitations, instance-based algorithms can be useful because their relative simplicity and, consequently, their ease of analysis (Aha et al., 1991).

A simple instance-based learning method is the *nearest neighbour* classifier, which is equivalent to the IB1 algorithm (Aha et al., 1991), The output of the classifier is the class label of the least dissimilar training instance,

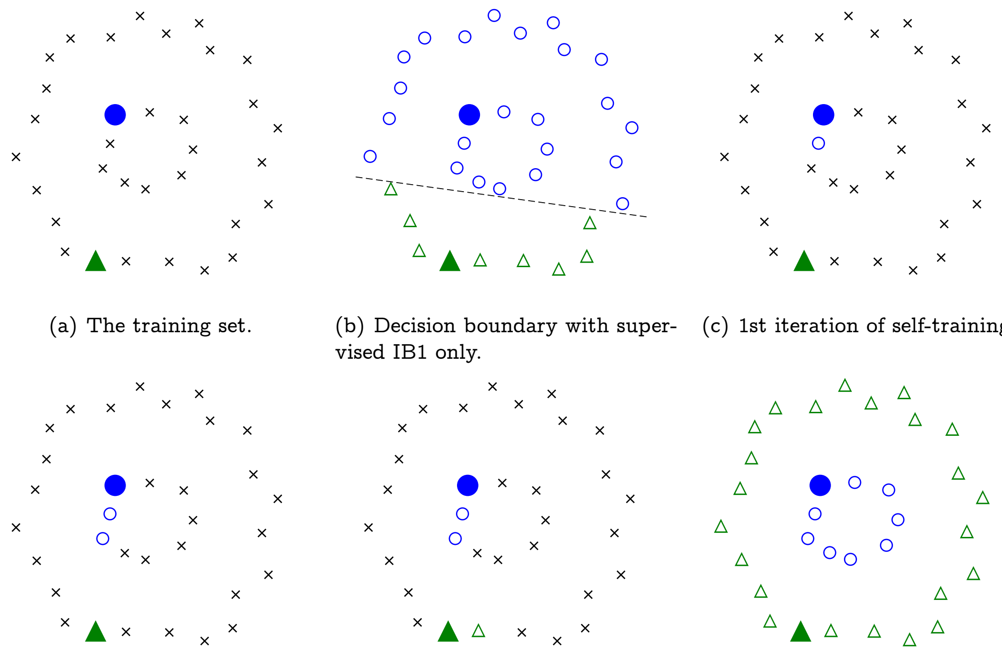
$$f(x) = y_i, \text{ where } i = \arg\min_{1 \leqslant i \leqslant l} d(x, x_i). \tag{2.6}$$

Instance-based learners may be employed as base classifiers in self-training algorithms. The certainty measure can be chosen to decrease as the distance between a instance and the labeled set $d(x, L_t) = \min_{x_i \in L_t} d(x, x_i)$ increases. In other words, if $d(x, L_t)$ is small, we can classify x with high certainty. Therefore, in each iteration the instance that gets added to L_{t+1} that lies closest to L_t. Figure 2.1 shows an example, where the base learner is IB1, the instances are points in the two-dimensional plane and d is the Euclidean distance.

Self-training with editing Instances and labels added to the training set L_{t+1} in iteration t of self-training are *consistent* with training set L_t, i.e. a classifier trained on L_t correctly predicts the labels of $L_{t+1} \setminus L_t$ with high certainty. However, if there are multiple examples in $L_{t+1} \setminus L_t$, they should also be consistent with each other. Adding multiple instances to the training set can be beneficial in many situations. For example, when the class distribution $p(y)$ is known *a priori* and we wish to maintain a similar training set class distribution $p(\hat{y}|x \in L_t)$. Additionally, inserting more instances to the labeled set per iteration reduces the number of iterations and training time.

One way of ensuring consistency of $L_{t+1} \setminus L_t$ is *editing*. In the first step, candidate examples are chosen from the set of instances which are classified with high certainty such that $p(\hat{y}|x \in L_{t+1}) \sim p(\hat{y}|x \in L_t)$. Next, the candidate set is edited by pruning inconsistent instances.

SETRED (Li and Zhou, 2005) is a semi-supervised learning algorithm that uses this approach to improve training set quality. Nearest-neighbour classification is employed as a base learner, while a graph-based inconsistency measure called *cut edge weight* statistic is used for editing.



(a) The training set. (b) Decision boundary with supervised IB1 only. (c) 1st iteration of self-training.

(d) 2nd iteration of self-training. (e) 3rd iteration of self-training. (f) Classification with self-training.

Figure 2.1: Semi-supervised IBl on the two-dimensional plane. There are two classes, circles (in blue) and triangles (in green). Bold symbols correspond to labeled instances in L, while elements of U are marked with crosses. Subfigures (c)–(e) show the first three iterations of Self-Training (listing 2.1(a)). The ground truth is shown on subfigure (f), which is also the output of the final classifier.

Co-training and multiview training Co-training (Blum and Mitchell, 1998) assumes that instances can be described by vectors of *features* and these features can be partitioned into two disjoint subsets, such that

- the two sets are conditionally independent given the class;
- each of the sets is sufficient for training a classifier.

Each of the subsets is used for training a different classifier.

In multiview training, there are more than two models, often constructed with different learning algorithms. However, the subsets of attributes need not be disjoint.

In iteration t, a different model is trained on the labeled training set L_t using each subset of attributes. High quality unlabeled training examples are determined by voting and are added to the labeled set L_{t+1}.

Therefore, co-training and multiview training can be regarded as a case of self-training where the base classifier is a specific kind of *ensemble* classifier.

Related concepts

On-line semi supervised learning In on-line semi-supervised learning, the unlabeled data is U is revealed to the learner only sequentially. At time moment t only instances $U_t = \{x_i\}_{i=l+1}^{l+t}$ are available as well as labeled data L. The classifier f_t, which was constructed given U_t and

L, then has to classify unlabeled instance x_{l+t}, such that the on-line classification error

$$\sum_t c(x_{l+t}, y_{l+t}, f_t(x_{l+t})) \tag{2.7}$$

is minimised.

As an illustrative example, consider an autonomous mobile robot working on a planet far, far away (X. Zhu, 2009). The robot takes pictures of rocks and sends results back to Earth. Because bandwidth is severely limited due to the great distance, only a fraction of images can be sent back to the control center. Therefore, the robot is equipped with a classifier system, which labels rocks as either interesting or not interesting. The classifier was taught to recognise interesting samples from previous missions.

Because the robot visits areas not visited by any past mission, it may miss some interesting rock samples if only supervised learning is used. On-line semi-supervised learning may be used to decide whether a photo should be sent to Earth and to construct a better classifier by retaining knowledge from previous results.

Active learning Another related concept to semi-supervised learning is active learning. Given unlabeled data U, the learning algorithm may enquire about the labels of some instances. Such queries are then answered by a human expert.

A combination of active and semi-supervised learning is possible, when given both U and L the learning algorithm may also pose questions to the expert (McCallum and Nigam, 1998). It is also possible to perform active semi-supervised learning in an on-line fashion (Goldberg et al., 2011).

For an excellent literature survey on active learning techniques please refer to Settles (2012).

2.1.3 Semi-supervised clustering

When performing conventional clustering, we wish to partition a set on instances $X = \{x_i\}_{i=1}^n$ into several groups. The number of groups k may be given in advance, or to be determined by the clustering algorithm.

In case of conventional approaches, there are no class labels available to the clusterer. However, when measuring the performance of a clustering algorithm, one can compare the output groups with a grouping produced by placing the instances with equal class labels in the same groups. Such disparity measures include purity, information gain, gain ratio (Quinlan, 1986) and the χ^2-measure. For an extensive study on disparity measures see White and Liu (1994).

In case of semi-supervised or constrained clustering problems, some information other that the unlabeled instances X is available and acts as supervision. This supervision may be provided in many forms:

- Must-link (ML) constraints force two instances to be placed in the same cluster.
- Cannot-link (CL) constraints prohibit placing two instances in the same cluster.
- δ-constraints force clusters to be well-separated. Formally, given a *distance* function $d(x_1, x_2)$ defined on instances, if x_1 and x_2 belong to a different cluster, $d(x_1, x_2)$ must be at least δ. (Davidson and Ravi, 2005)
- ϵ-constraints force instances in a cluster to be close to each other. If an instance x_1 belongs to the cluster S, there must be another instance $x_2 \in S$ such that $d(x_1, x_2) < \epsilon$. (Davidson and Ravi, 2005)

- ϵ-path-constraints are an extension of ϵ-constraints. For any pair of distinct points x_1, x_1 in the same cluster S, there must be a sequence of instances belonging to S $(x_1, a_1, a_2, \dots, a_r, x_2)$ such that the distance of adjacent instances in the sequence is smaller than ϵ. (Davidson and Ravi, 2007)
- A labeled set L may be provided either for the generation of aforementioned constraints or to calculate and minimise disparity between classes and clusters.

Constraints can be either used to determine an objective function to be optimised (Basu et al., 2004; Klein et al., 2002) or strictly enforced during clustering (Wagstaff and Cardie, 2000; Wagstaff et al., 2001). It is also possible to learn *distance measures* from constraints (Reuter et al., 2011).

It is possible to provide constraints such that there is no clustering that satisfies them. For example, a ML and CL constraint involving the same pair (x_1, x_2) of points leads to a contradiction. Even if a correct clustering exists, some combination of constraints lead to NP-complete problems and therefore are computationally infeasible (Davidson and Ravi, 2005, 2007).

Semi-supervised hierarchical clustering

Hierarchical clustering aims build a hierarchy of clusters. This hierarchy is usually illustrated with a *dendrogram*. Figure 2.2(b) shows an example.

There are two basic types of hierarchical clustering methods.

- *Agglomerative* clustering starts with trivial clusters of single instances and constructs the dendrogram by merging clusters.
- *Divisive* clustering, in contrast, follows a 'top down' approach and divides clusters into smaller ones.

Cluster merging or division is done *greedily*. Agglomerative clustering selects the pair of clusters to in order to minimise a measure of cluster dissimilarity, Similarly, divisive clustering partitioned a cluster to maximise dissimilarity.

Given a distance function of instances $d(x_1, x_2)$, several cluster distances $d(C_1, C_2)$ are possible, for example:

- In *single-linkage* clustering (SLINK), the cluster distance is the minimum of pairwise instance distances,

$$d_{\text{single}}(C_1, C_2) = \min_{\substack{x_1 \in C_1 \\ x_2 \in C_2}} d(x_1, x_2). \tag{2.8}$$

- *Complete-linkage* (CLINK) and *average-linkage* can be defined analogously:

$$d_{\text{complete}}(C_1, C_2) = \max_{\substack{x_1 \in C_1 \\ x_2 \in C_2}} d(x_1, x_2), \tag{2.9}$$

$$d_{\text{average}}(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{\substack{x_1 \in C_1 \\ x_2 \in C_2}} d(x_1, x_2). \tag{2.10}$$

The inclusion of must-link (ML) and cannot-link (CL) constraints was shown to improve clustering accuracy and robustness (Kestler et al., 2006; Miyamoto and Terami, 2010).

Listing 2.2 shows a simple agglomerative hierarchical clustering algorithm with ML and CL constrains while Figure 2.2(d) illustrates the resulting dendrogram.

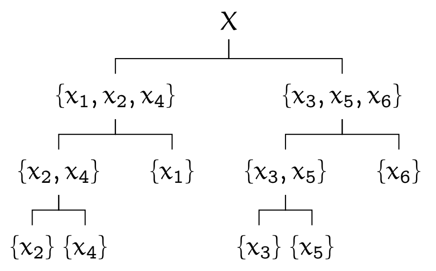
```
AGGLOMERATIVE-CLUSTERING($X, ML, CL$)
foreach $x \in X$
    MAKE-CLUSTER($x$)
foreach $\{x_1, x_2\} \in ML$
    $C_1 = $ FIND-CLUSTER($x_1$)
    $C_2 = $ FIND-CLUSTER($x_2$)
    MERGE-CLUSTERS($C_1, C_2$)
while clusters can be merged
    Find $C_1$ and $C_2$ such that $d(C_1, C_2)$ is minimal
    and COMPATIBLE($C_1, C_2, CL$) returns TRUE.
    MERGE-CLUSTERS($C_1, C_2$)

COMPATIBLE($C_1, C_2, CL$)
foreach $x_1 \in C_1$
    foreach $x_2 \in C_2$
        if $\{x_1, x_2\} \in CL$ return FALSE
return TRUE
```

Listing 2.2: Semi-supervised agglomerative clustering with instance-level constraints.

$$\begin{array}{c|cccccc|}
 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
x_1 & 0 & 1 & 2 & 3 & 4 & 5 \\
x_2 & 1 & 0 & 9 & 0.1 & 17 & 4 \\
x_3 & 2 & 9 & 0 & 8 & 0.5 & 2 \\
x_4 & 3 & 0.1 & 8 & 0 & 20 & 6 \\
x_5 & 4 & 17 & 0.5 & 20 & 0 & 3 \\
x_6 & 5 & 4 & 2 & 6 & 3 & 0
\end{array}$$

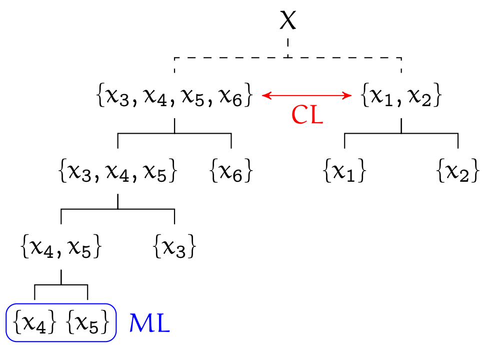
(a) Distance function $d(\cdot,\cdot)$.



(b) Resulting supervised dendrogram.

$$\begin{array}{c|cccccc|}
 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
x_1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
x_2 & \cdot & \cdot & \cdot & CL & \cdot & \cdot \\
x_3 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
x_4 & \cdot & CL & \cdot & \cdot & ML & \cdot \\
x_5 & \cdot & \cdot & \cdot & ML & \cdot & \cdot \\
x_6 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{array}$$

(c) Instance-level constraints.



(d) Resulting semi-supervised dendrogram.

Figure 2.2: Hierarchical agglomerative clustering with single-linkage.

2.2 Time-series data mining

Time series are temporal sequences of scalar- or vector-valued measurements.

We shall denote the *length* of the time series x by $|x|$. The time series which has length 0 is the *empty* time series ϵ.

For $t = 1, 2, \ldots, |x|$ and time series x, the value of the measurement taken in time moment t is $x[t]$.

If the time series is vector-valued—multivariate—, let us denote the scalar-valued time series which is constructed by taking the ith components of the measurement vectors by $x^{(i)}$, which will be called the ith *channel* of x. Thus the value of the d dimensional vector-valued time series at time moment t is $x[t] = (x^{(1)}[t], x^{(2)}[t], \ldots, x^{(d)}[t])$.

2.2.1 Distance functions for time series

Euclidean distance

A straightforward distance measure between two scalar-valued time series, x_1 and x_2 of equal length is their Euclidean distance

$$d_{\mathrm{EU}}(x_1, x_2) = \sqrt{\sum_t \big(x_1[t] - x_2[t]\big)^2}. \tag{2.11}$$

Dynamic Time Warping

Euclidean distance is only defined for time series of equal length. Moreover, we may want to allow slight temporal differences between two time series, because a real-world phenomenon may not always start at the same time and happen with constant speed.

Dynamic Time Warping (DTW) allows for *elongation* of time series. More formally, the DTW distance of time series is

$$d_{\mathrm{DTW}}(x_1, x_2) = c\big(|x_1|, |x_2|\big) \tag{2.12}$$

where c is recursively defined as

$$c(0,0) = 0, \tag{2.13}$$

$$c(i,0) = c(0,j) = \infty, \tag{2.14}$$

$$c(i,j) = d_{\mathrm{inner}}(x_1[i], x_2[j]) + \min \left\{ \begin{array}{l} c(i-1, j-1), \\ c(i, j-1) + c_{\mathrm{el}}, \\ c(i-1, j) + c_{\mathrm{el}} \end{array} \right\} \tag{2.15}$$

and $d_{\mathrm{inner}}(\cdot,\cdot)$ is the *inner* distance function while c_{el} is the cost associated with elongation.

The inner distance function expresses the cost of matching two measurements. In the case of univariate time series, usually *absolute difference* is selected,

$$d_{\mathrm{inner}}(a, b) = |a - b|. \tag{2.16}$$

For multivariate time series, the Euclidean distance

$$d_{\mathrm{inner}}(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_{\mathrm{EU}} = \sqrt{\sum_{i=1}^{d} \big(a_i - b_i\big)^2} \tag{2.17}$$

can be used.

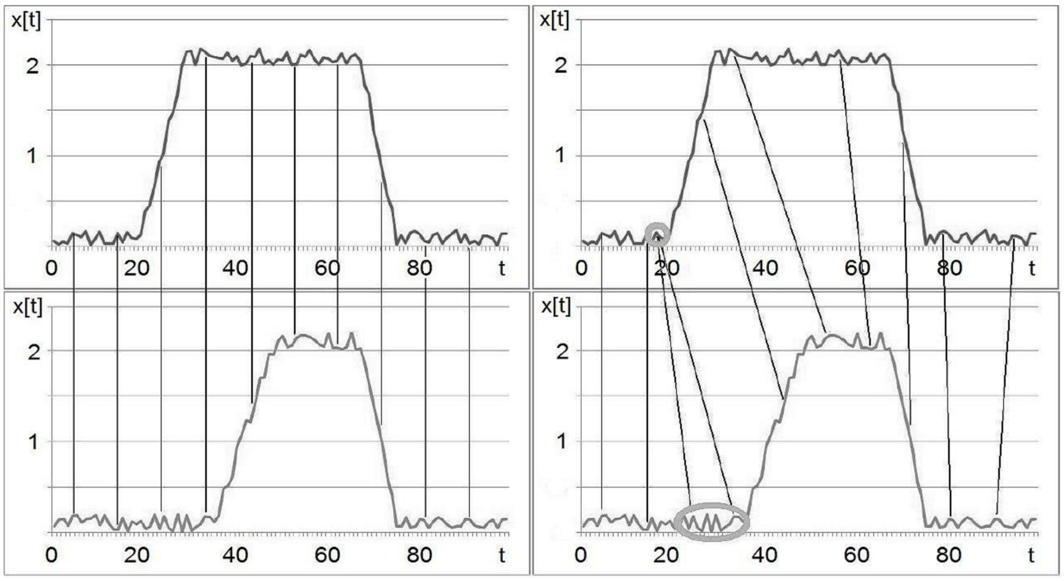


Figure 2.3: Comparison of Euclidean distance and Dynamic Time Warping. Euclidean distance (on the left) tries to match time series segments exactly. In contrast, Dynamic Time Warping (on the right) can—correctly—determine that the peak between 20–80 on top is an elongation of the peak between 40–80 on bottom. This illustration is adapted from K. A. Buza (2011).

```
DTW($x_1, x_2$)
1  $c[0,0] = 0$                       // Initialise the matrix c with (2.13).
2  for $i = 1 \to |x_2|$
3      $c[0,i] = \infty$              // Initialise element in the first row by (2.14).
4  for $i = 1 \to |x_1|$
5      $c[i,0] = \infty$              // Initialise element in the first column by (2.14).
6      for $j = 1 \to |x_2|$
           // The rest of the row is calculated by the recursive formula (2.15).
7          $c[i,j] = d_{\text{inner}}(x_1[i], x_2[j]) + \min\{c[i-1,j-1],$
                                                          $c[i,j-1] + c_{\text{el}},$
                                                          $c[i-1,j] + c_{\text{el}}\}$
8  return $c[|x_1|, |x_2|]$            // The distance is the bottom-right element of c.
```

Listing 2.3: Dyamic Time Warping.

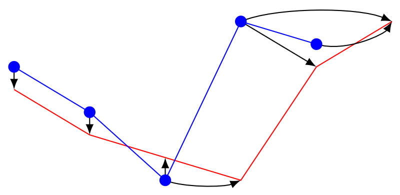
Calculating DTW The DTW distance of two time series can be computed by elementary dynamic programming. This algorithm is displayed in Listing 2.3. Execution takes $O(|x_1| \cdot |x_2|)$ time, which is quadratic in the average length of time series.

The presented algorithm calculates a matrix of distance values c. The values which contribute to the final distance form a *warping path* (Figure 2.4(a)). It is possible to reduce runtime by constraining the warping path to a specific region of the matrix and avoiding calculation of elements which lie outside that area. Such limits also improve classification accuracy (C. (Ratanamahatana and Keogh, 2005).

One technique for reducing the resource demand of DTW is the use of a *warping window*,

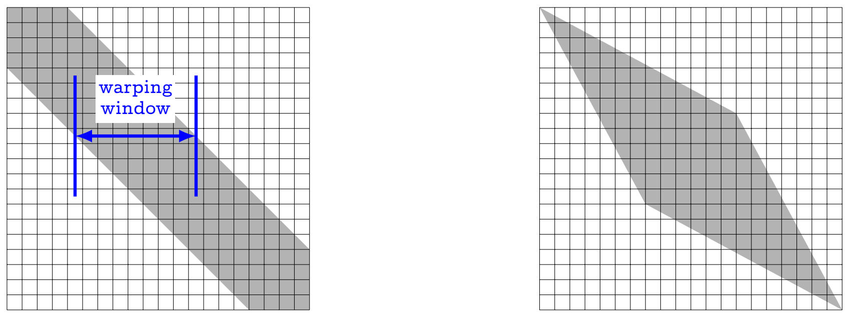
$$\begin{array}{c|cccccc}
 & & 3 & 1 & -2 & 5 & 4 \\
\hline
 & 0 & \infty & \infty & \infty & \infty & \infty \\
2 & \infty & 1 & 2 & 6 & 9 & 11 \\
0 & \infty & 4 & 2 & 4 & 9 & 13 \\
-1 & \infty & 8 & 4 & 3 & 9 & 14 \\
-2 & \infty & 13 & 7 & 3 & 10 & 15 \\
3 & \infty & 13 & 9 & 8 & 5 & 6 \\
5 & \infty & 15 & 13 & 16 & 5 & \leftarrow 6
\end{array}$$

(a) Warping path.



(b) Matching measurements.

Figure 2.4: Calculation of Dyamic Time Warping.



(a) Sakoe–Chiba band.

(b) Itakura parallelogram.

Figure 2.5: Restricted Dynamic Time Warping examples.

also known as the *Sakoe–Chiba band* (Figure 2.5(a)). The width of the window is usually around 5 percent of the length of the time series and can be selected by cross-validation.

Another set of constraints form the *Itakura parallelogram* (Figure 2.5(b)), which allows greater elongation in the middle of the time series than in the beginning and end.

Properties Dynamic Time Warping is symmetric—$d_{DTW}(x_1, x_2) = d_{DTW}(x_2, x_1)$—and nonnegative if the inner distance is symmetric and nonnegative.

DTW, in the general case, does not form a *metric space* with a set of time series. Neither the *identity of indiscernibles* nor the *triangle equality* is satisfied, i.e. there may exist distinct time series $x_1 \neq x_2$ such that

$$d_{DTW}(x_1, x_2) = 0 \tag{2.18}$$

and there may exist time series x_1, x_2, x_3 such that

$$d_{DTW}(x_1, x_2) > d_{DTW}(x_1, x_3) + d_{DTW}(x_2, x_3). \tag{2.19}$$

Some learning and indexing techniques might require the distance function to satisfy the metric axioms (Hjaltason and Samet, 2003). In this case, one of the metrics which are conceptually similar to DTW can be used, such as edit distance with real penalty (Chen and Ng, 2004) or move-split-merge (Stefan et al., 2012).

Indexing Dynamic Time Series is upper bounded by the computationally cheaper Euclidean distance.

Efficient lower bounds also exist, such as those suggested by Keogh (2002); Keogh et al. (2006a); Kim et al. (2001); Yi et al. (1998). These lower bounds, combined with appropriate indexing can be used for fast nearest-neighbour searches.

This property makes DTW a suitable choice for instance-based time-series classification.

It is also possible to employ the lower bounds in an *anytime* framework (Q. Zhu et al., 2012). In the beginning a distance matrix is initialised with lower bounds. Values in the distance matrix are incrementally refined by the calculation of true DTW distance. The refining step can be halted at any time. The resulting distance matrix is expected to be a good approximation of the true DTW distance matrix.

2.2.2 Hubness in time series databases

A univariate time series of length l can be represent as a vector of l real numbers. This suggests that a database with time series of length l is an l-dimensional vector space. In practical applications, consecutive measurements usually highly correlate—a change of a quantity usually does not happen too fast—real world time series databases still have relatively high 'time' or intrinsic dimensionality (Radovanovic et al., 2010b). The dimensionality is even greater in multivariate time series databases. This fact gives rise to a collection of phenomena known as the *curse of dimensionality*.

For a data set $X = \{x_i\}_{i=1}^n$, let $g_N^1(i)$ denote the number of instances in X for which x_i is *nearest neighbour*, i.e.

$$g_N^1(i) = \sum_{\substack{j=1 \\ j \neq i}}^{n} \left[\!\left[\operatorname*{arg\,min}_{\substack{1 \leqslant k \leqslant n \\ k \neq j}} d(x_j, x_k) = i \right]\!\right]. \tag{2.20}$$

The quantity $g_N^k(i)$ can be similarly defined to be the number of instances in X that have x_i among their k nearest neighbours.

The *skewness* of g_N^k is its standardised third moment,

$$S_{g_N^k} = \frac{\mathbb{E}_i\left[\left(g_N^k(i) - \mu_{g_N^k}\right)^3\right]}{\sigma^3_{g_N^k}}. \tag{2.21}$$

In high dimensional databases, g_N^k is skewed to the right. This means there is a small set of instances which are k-nearest neighbours to a large number of other instances. Members of this set are called *hubs* (Radovanovic et al., 2010a).

Coverage graphs

The k-*nearest neighbour coverage* graph—a.k.a. k-nearest neighbour graph—is useful tool for analysing hubness. Consider the dataset $X = \{x_i\}_1^n$ and the directed graph $G_N^k = (X, E_N^k)$ which has an edge $(x_i \to x_j)$ between x_i and x_j if and only if x_j has x_i among its k nearest neighbours.

The k-occurrence score of an instance is its out-degree in G_N^k. Thus we can give a new definition of g_N^k:

$$g_N^k(x_i) = \operatorname{out\,deg}_{G_N^k} x_i. \tag{2.22}$$

Good hubness

Good hubs in a dataset are hubs which belong to the same class as their neighbours. More formally, we can define the $g_G^k(x_i)$ *good* k-*occurrence score* of x_i as the number of instances $\{x_j\}$ in the database that have x_i among their k nearest neighbours and satisfy $y_i = y_j$. *Good hubs* have $g_G^k / g_N^k \approx 1$.

Good hubness can be exploited to increase classification efficiency (K. Buza et al., 2011a). By discarding the labeled data set except good hubs, the performance of instance-based learning may be dramatically improved. This is possible by constructing a *cover* of the k-nearest neighbour graph. Every instance in $X \setminus C$ should have at least one instance in C among its k-nearest neighbours. In other words, vertices of G_N^k which are not in C should be separated from C by only a single edge.

A good cover contains as many good hubs as possible. Constructing such cover, in the general case, an NP-complete problem. However, cover construction for 1-nearest neighbour graphs is tractable (K. Buza et al., 2011a).

Bad hubness

Just as godd hubs can correctly classify many instances, *bad hubs*, which have $g_G^k / g_N^k \ll 1$, are responsible for a surprisingly large number of misclassifications.

The presence of bad hubs also results in *cluster assumption violation* (CAV). In other words, some instances that are 'close' to each other do not belong to the same class. This, among other unfavorable effects, limits the application of clustering based semi-supervised learning methods.

To provide a numerical measure of bad hubness, we can introduce the normalised total bad hubness $\widetilde{BN}_k$, which is the sum of all bad k-occurrences $g_N^k - g_G^k$ normalised with the sum of all k-occurrences g_N^k. Formally,

$$\widetilde{BN}_k = 1 - \frac{\sum_{i=1}^{n} g_G^k(x_i)}{\sum_{i=1}^{n} g_N^k(x_i)} \tag{2.23}$$

Radovanovic et al. (2010b) divided time series databases into three *zones* based on the skewness of 10-occurrence scores $S_{g_N^{10}}$ and normalised total bad 10-hubness $\widetilde{BN}_{10}$:

- Datasets in Zone 3 have no or very little bad hubness. The cluster assumption is not significantly violated.
- In Zone 2, $\widetilde{BN}_{10}$ is high. However, g_N^{10} is not skewed to the left—$S_{g_N^{10}}$ is low or negative. This means bad hubs are not strong enough to dramatically impact supervised classification.
- In Zone 1, large bad hubness is combined with large $S_{g_N^{10}}$. Strong bad hubs violated the cluster assumptions significantly and classification becomes a difficult task.

2.2.3 Semi-supervised time-series classification

Positive–unlabeled learning

In positive–unlabeled (PU) learning, the problem which the recognition system aims to solve is a *binary classification* problem. Instances may belong to the *positive* or *negative* class. Positive instances may be a lot rarer than negative ones. The learning algorithm is given a training set of positive $P = \{x_i\}_{i=1}^{l}$ and unlabeled $U = \{x_i\}_{i=l+1}^{n}$ examples. The majority of unlabeled examples probably belong to the negative class.

As an example, consider a database of patients' medical history. Some patients are in a *risky* health condition, they form the class of positive instances. The other, non-risky patients belong to the negative class. Medical doctors—domain experts—cannot be absolutely sure that a patient's condition is non-risky. Hence only positive examples are available. Naturally, we aim to recognise risky patients before their health significantly

```
POSITIVE-UNLABELED-1-NN($P, U$)
 1  $P_0 = P$
 2  $U_0 = U$
 3  $t = 0$
 4  repeat
        // The unlabeled example which is closest to the set of currently
        // positive-labeled examples $P_t$ will be considered positive.
 5      $x_{best} = \arg\min_{x \in U_t} d(P_t, x)$
 6      $L_{t+1} = L_t \cup \{x_{best}\}$
 7      $U_{t+1} = U_t \setminus \{x_{best}\}$
        // Form a training set by labeling the yet unlabeled instances as negative.
 8      $X_{t+1} = \{(x, +) : x \in P_t\} \cup \{(x, -) : x \in U_t\}$
        // Train a nearest-neighbour classifier.
 9      $M_{t+1} =$ SUPERVISED-1NN-CLASSIFIER($X_t$)
10      $t = t + 1$
11  until ¬SHOULD-STOP()
12  return $M_t$
```

Listing 2.4: Positive-unlabeled instance-based learning.

deteriorates. A positive–unlabeled learner may create a better recognition system than a supervised only learner, given enough patients' data whose condition is currently unknown.

The first studies on semi-supervised time series classification include Wei and Keogh (2006), which was concerned with positive–unlabeled learning. The authors proposed and algorithm which is a modification of instance-based self training (subsection 2.1.2) with early stopping. Listing 2.4 shows the modifications.

Stopping heuristic A critical part of the POSITIVE-UNLABELED-1-NN algorithm is the criterion for stopping, SHOULD-STOP. Clearly, the number of hypothesised positive instances increases monotonically with the iterations number because $|P_{t+1}| = |P_t| + 1$. If we continue the learning process until we run out of unlabeled examples—$|U_t| = 0$—we will consider all the initially unlabeled examples as positive. Thus the training set X for the nearest-neighbour classifier (line 8 in Listing 2.4) will only contain positive-labeled examples. Such training set is useless, because the resulting classifier will have a constant output of '+'.

Wei and Keogh (2006) have observed a correlation between abrupt decrease of the *closest pair*'s distance in P_t

$$d_P^{min}(t) = \min_{\substack{x_i, x_j \in P_t \\ x_i \neq x_j}} d(x_i, x_j) \tag{2.24}$$

and deterioration of classification accuracy of M_t. In other words, negative examples tend to be more 'dense' than positive examples. Although a model's true accuracy cannot be exactly determined at the training phase—if that was possible we would not need semi-supervised learning—this correlation can still be exploited in stopping heuristic construction.

A formal description of the aforementioned heuristic was developed by C. A. Ratanamahatana and Wanichsan (2008).

Positive-unlabeled learning by clustering Another approach for positive-negative time-series learning was studied by Nguyen et al. (2011).

Their method first clusters the training examples by k-means clustering. Then principal component extraction is performed on the clusters. Principal components are used to define a new dissimilarity function for cluster.

This new dissimilarity function helps the identification of *reliable negative* (RN), *likely positive* (LP) and *ambiguous* (AMBI) clusters. A procedure similar to nearest neighbour self-training is carried out on the clusters, with RN clusters are labeled negative, LP clusters as labeled positive and AMBI clusters as unlabeled examples.

The idea of applying semi-supervised learning to clusters of instances as opposed to single instances is somewhat similar to *harmonic mixtures* (X. Zhu and Lafferty, 2005). In that variation of semi-supervised learning, fuzzy clustering is performed to reduce the resource demand of a semi-supervised algorithm. Because there are less clusters than instances in a non-trivial clustering, processing clusters is computationally cheaper than processing single instances.

Multiclass self-training

C. A. Ratanamahatana and Wanichsan (2008) also studied the standard instance-based self-training classifier for time series. I selected their approach as a baseline to which I compared my algorithm in Chapter 4.

Because there are examples available from all classes, there is no need for a stopping heuristic in multiclass self-training. The self-training process is iterated until there are no more unlabeled examples.

Self-training with HMMs Another approach of time series self-training uses Hidden Markov Models[1] (Zhong, 2005).

Unfortunately, Hidden Markov Models (HMMs) cannot be directly trained in a semi-supervised setting by maximisation of the labeled and unlabeled data likelihood. Unlabeled data, in general, does not improve the performance of the Baum-Welch algorithm that is employed to perform this maximisation (Elworthy, 1994).

However, self-training may be used to train Hidden Markov Models:

- For each class y, a HMM with parameter vector θ_y can be trained.
- The certainty measure for and unlabeled instance is set to the posterior probability, i.e.

$$\textsc{Certainty}(x) = p(x|\theta_{\hat{y}}), \tag{2.25}$$

  where $\hat{y}$ is the hypothesised class label of x.
- Self-Training (Listing 2.1(a)) is performed on the dataset.

This approach has shown promising results on small databases. However, to our best knowledge, no further results are available for larger databases.

[1] Hidden Markov Models are probabilistic, generative models for sequences. They are utilised for time series generation as well as supervised classification.

3 My approach: constrained SLINK with DTW

3.1 Description of the algorithm

Consider the semi-supervised classification problem, in which a set of labeled time-series $L = \{(x_i, y_i)\}_{i=1}^{l}$ and a set of unlabeled time-series $U = \{x_i\}_{i=l+1}^{n}$ is available to a learner. We wish to construct a classifier that can classify any time-series—not only elements of U—with high accuracy, i.e. is capable of *induction*.

I propose a novel semi-supervised time series classification method, constrained SLINK with DTW:

1. The labeled and unlabeled examples in the training set are clustered with constrained single-linkage (SLINK) hiearchical agglomerative clustering. The following two extenstions are used, which, to our best knowledge, have never been used together for semi-supervised time series classification:
   - There is a CL constraint for each pair of labeled examples.
   - The dissmilarity function for time series is Dynamic Time Warping (DTW).
2. The resulting top-level clusters are labeled by their corresponding 'seeds'.
3. The final classifier is 1-nearest neighbour trained on the resulting labeling. This classifier can be applied to *unseen* test data.

3.2 A graph theoretic view on time series nearest neighbour self-training

The properties of minimum spanning tree algorithms form a theoretical motivation of constrained SLINK with DTW.

Consider the set of all—labeled or unlabeled—examples $X = \{x_i\}_{i=1}^{n}$. Let $G = (X, V)$ be an undirected *complete* graph width edge weights $w_{ij} = w(\{x_i, x_j\}) = d(x_i, d_j)$.

Define a *spanning forest* as a set of acyclic undirected subgraphs—trees—$\mathcal{T} = \{T_i\}_{i=1}^{j}$ that satisfy the following properties:

- The trees are disjoint, i.e. $\forall i : x_i \in V(T_a) \wedge x_i \in V(T_b) \implies a = b$.
- The trees together span the whole set of examples, i.e. $\bigcup_{i=1}^{l} V(T_i) = X$.
- The ith tree contains the i labeled example, i.e. $\forall 1 \leqslant i \leqslant l : x_i \in V(T_i)$.

A spanning forest is a *minimum spanning forest* if the sum of its edge weights $W(\mathcal{T}) = \sum_{i=1}^{l} \sum_{e \in E(T_i)} w(e)$ is minimal.

Let us define the graph $G^\star = (X \cup \{\star\}, E^\star)$, which is an extension of G with a super-vertex $\star$. This super-vertex is connected to the labeled examples with 0-weight edges, i.e.

$$E^\star = E \cup \{\{x_i, \star\} : 1 \leqslant i \leqslant l\}, \tag{3.1}$$

$$x_{i\star} = w(\{x_i, \star\}) = 0. \tag{3.2}$$

The tree which is formed by taking the union of the trees in a minimum spanning forest $\mathcal{T}$ of G and the new edges from $\star$ has no greater sum weight than a minimum spanning tree of $G^\star$. Therefore, it is itself a minimum spanning tree $T^\star$ of $G^\star$.

The instance-based self-training algorithm presented in subsection 2.1.3 (Figure 2.1) can be viewed as a specific way of finding a minimum spanning tree of $G^\star$. This tree contains all the outgoing edges from $\star$. Thus, it is a minimum spanning forest of G.

More concretely, if all the edge weights w_{ij} in G are strictly positive, instance based self-training—which I selected as a baseline to compare my algorithm to—is equivalent to running Prim's algorithm with $\star$ as the root node. In the first l iterations, the algorithm adds the labeled instances $\{x_i\}_{i=1}^{l}$ to the tree. Every iteration onwards, the set nodes in the growing tree equals the set of already labeled instance.

The role of $\star$ in this process is entirely superficial: after the lth iteration it plays no further role. In fact, we could simply start with the graph G and consider multiple root nodes. This is exactly what instance-based self-training does. The minimum spanning forest can be constructed by adding a new edge connecting the instance x_{best} to the closest labeled instance is P_t.

3.2.1 From self-training to cluster-and-label

The *other* famous minimum spanning tree algorithm for graphs is Kruskal's method. Note that the forest which is gradually joined by Kruskal's algorithm is a set of clusters in some level of a *single-linkage* (SLINK) *hierarchical agglomerative clustering* dendrogram.

In contrast with nearest neighbour self-training, $\star$ cannot be simply ignored. In the first l iterations, the algorithm will join all the labeled instances into one cluster.

No two clusters which contains a labeled instance may be merged, because, in the presence of $\star$, they would be the *same* cluster.

When removing $\star$ from the graph, we must take care not to create clusters with more than one labeled instance in them. This is equivalent to a *cannot-link* (CL) constraint between each labeled instance.

If there were l labeled instances among the training examples, hierarchical clustering with the above CL constraints will terminate with l clusters. Unlabeled instances should be labeled with the same class label as the 'seed'—labeled instance—in their cluster.

3.3 Properties of constrained SLINK

3.3.1 Similarity to the 1-nearest neighbour graph

Single-linkage clustering or, equivalently, Kurskal's algorithm for databases with prevalent *hubness* has some interesting properties.

The first edge encountered in Kruskal's algorithm which contains some instance x_i is the edge between x_i and its nearest neighbour x_j. When this edge $\{x_i, x_j\}$ is processed, x_i is a single-instance cluster. If x_i is unlabeled, it will be added to x_j's cluster. If x_i is labeled, it will still be added to x_j's cluster. Thus, the only case when the edge $\{x_i, x_j\}$ is not added to the minimum spanning forest is when x_i is labeled and x_j's cluster already contains another labeled instance.

Therefore, the resulting forest will contain 'almost all' edges of the—undirected—1-nearest neighbour graph.

3.3.2 Interaction with hubs

Because of the similarity of the minimum spanning forest and the 1-nearest neighbour graph, constrained SLINK with DTW for time series benefits from good hubs greatly.

'Almost all' instances will have the same label as their nearest neighbour. Consequently, 'almost all' hubs in the labeled or unlabeled training set will be good hubs in the final labeling.

With this knowledge, we can state what is assumed by constrained SLINK with DTW about the time-series database: *'almost all' hubs are good hubs.*

3.3.3 Transitive hubs

In constrained SLINK with DTW, instances will generally have the same class label as their nearest neighbours. This means that hubs will also have the same label as their nearest neighbour, and those hubs which have hubs as their nearest neighbours will also have the same label as thier nearest neighbour, etc.

The effect of hubs is *transitive*: a hubs class label will determine the class label of 'almost all' nodes to which it is connected by a chain of nearest neighbours.

Let us introduce the *transitive closure* $G^T_{\text{1-NN}}$ of the 1-nearest neighbour graph $G_{\text{1-NN}}$ in which there is a—directed—edge between vertices x_i and x_j if and only if x_j is accessible from x_j in $G_{\text{1-NN}}$. Formally,

$$(x_i \to x_j) \in G^T_{\text{1-NN}} \iff (x_i \rightsquigarrow x_j) \in G_{\text{1-NN}}. \tag{3.3}$$

We can introduce *transitive* 1-*occurrence* analogously to 1-occurrence values:

$$g^1_{TN}(x_i) = \operatorname{out deg}_{G^T_{\text{1-NN}}} x_i, \tag{3.4}$$

$$g^1_{TG}(x_i) = \sum_{(x_i \to x_j) \in G^T_{\text{1-NN}}} [\![y_i = y_j]\!]. \tag{3.5}$$

Generalisation to transitive k-occurrence scores is straightforward.

Because the minimum spanning forest generated by constrained SLINK with DTW and the 1-nearest neighbour graph is similar, g^1_{TN} and g^1_{TG} should give insight how accurate this semi-supervised classification technique will be. In my empirical evaluation in Chapter 4, I confirm this hypothesis.

The assumption about data taken by constrained SLINK with DTW can by phrased more accurately: 'almost all' hubs are *transitively* good hubs.

4 Experimental evaluation

4.1 Protocol of evaluation

I compared the performance of my algorithm with a state-of-the-art supervised time-series classifier CITE and a state-of-the-art semi-supervised time-series classifier. Both classifiers are based instance-based learning—nearest-neighbour classification—and Dynamic Time Warping.

The test runs were repeated 30 times so that a paired two-tailed t-test could be performed. Results were deemed statistically significant when $p < 0.05$ was satisfied.

In the experiments, I wanted to simulate a scenario in which large database is available with relative few labeled instances. Not only good accuracy of labeling the unlabeled instances, but also good capability of induction—i.e. accuracy on instances unknown in the training stage—was demanded.

In a single trial, the classifiers had access to a randomly selected 90 percent of the database. However, class labels of only a randomly selected 10 and 20 percent of the database were available, respectively. The rest of the training instances had their class labels discarded.

The 10 percent of the database which was unavailable to the learner simulated the new, unknown instances. The *test set* was formed by these new instances along the instances which had their labels before training stage.

The reported measure of accuracy is the *misclassification rate*, which is the number of misclassified instances divided by the size of the raining set,

$$r_{\text{misclass.}} = \frac{\sum_i [\![\hat{y}_i \neq y_i]\!]}{|T_{\text{test}}|}. \tag{4.1}$$

I also report the performance of the state-of-the-art supervised classifier on the data with no class labels discarded from the training set. This is an *upper bound* on the accuracy of a semi-supervised learner and corresponds to the situation when an *oracle* has labeled all the unlabeled data correctly.

4.2 Time-series databases

To aid reproducibility, I performed the trials with 45 publicly accessible real-world datasets[1] from the UCR time series repository (Keogh et al., 2006b). The time series databased involved in the experiments are shown in Table 4.2.

[1] Recently, two more datasets—Non-Invasive Fetal ECG Thorax1 and 2—has been added to the repository. However, I did not perform the trials on them because DTW distance matrices were not readily available and require extremely resource-intensive preprocessing to calculate.

4.3 Results

As a baseline, I selected the state-of-the-art Dynamic Time Warping based supervised time series classifier. I compared the performance of the multiclass algorithm introduced by C. A. Ratanamahatana and Wanichsan (2008) and my approach to this baseline.

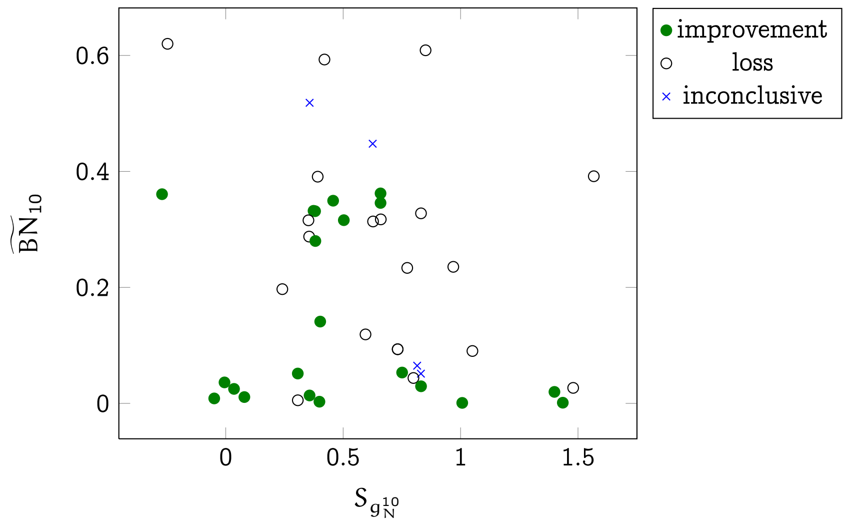
Semi-supervised learning only improves classification accuracy if the assumptions taken by the model match the data closely. As expected from its interaction with *hubness* and *transitive* hubness, my method outperformed the baseline—and in a number of cases, the other algorithm—when a dataset had low *bad hubness* and *transitive bad hubness*. If bad hubs were prominent in the data, my algorithm could still outperform the baseline in a few datasets where the overall number of hubs was low.

These observations are illustrated in Figure 4.1. Low values of $\widetilde{\mathrm{BN}}$ correspond to low bad hubness, while low valued of $S_{\mathcal{G}_N}$ correspond to low hubness overall. Both the 10-hubness and the transitive 1-hubness statistics show a correlation between low bad hubness and improved accuracy.

The total number of wins and losses against the baseline are summarised in Table 4.1. The self-training based algorithm introduced by C. A. Ratanamahatana and Wanichsan (2008) is shown in the SS-1-N column, while my algorithm—constrained SLINK with DTW—is shown in the SS-Slink column.

	10% labeled		20% labeled	
	SS-1-N	SS-Slink	SS-1-N	SS-Slink
improvements	10	20	11	19
losses	30	19	28	19
inconclusive	5	6	6	7

Table 4.1: Summary of wins and losses against the baseline.



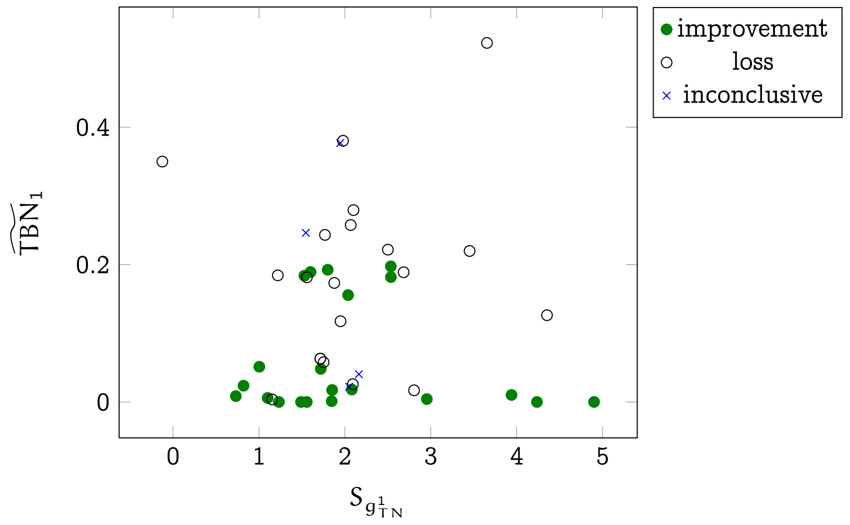


Figure 4.1: Effects of hubness on my algorithm's accuracy.

	$S_{g_N^{10}}$	$\widetilde{BN}_{10}$	$S_{g_{TN}^{1}}$	$\widetilde{TBN}_1$	•/○
50words	0.659	0.362	2.535	0.197	●
Adiac	0.357	0.518	1.946	0.377	
Beef	−0.248	0.620	−0.125	0.350	○
Car	1.567	0.392	1.768	0.243	○
CBF	1.435	0.001	4.235	0.000	●
ChlorineConcentration	0.503	0.316	2.954	0.004	●
CinC_ECG_torso	0.079	0.011	1.232	0.000	●
Coffee	−0.271	0.361	1.004	0.051	●
Cricket_X	0.380	0.331	1.800	0.192	●
Cricket_Y	0.457	0.349	1.600	0.189	●
Cricket_Z	0.374	0.332	1.529	0.184	●
DiatomSizeReduction	0.357	0.014	1.101	0.006	●
ECG200	0.241	0.197	1.949	0.118	○
ECGFiveDays	−0.005	0.036	0.730	0.009	●
FaceFour	0.402	0.141	1.719	0.048	●
FacesUCR	0.751	0.053	2.081	0.018	●
fish	0.831	0.328	2.684	0.189	○
Gun_Point	0.307	0.052	0.820	0.024	●
Haptics	0.851	0.609	3.654	0.523	○
InlineSkate	0.420	0.593	1.979	0.380	○
ItalyPowerDemand	0.831	0.051	2.163	0.040	
Lighting2	0.355	0.288	1.877	0.173	○
Lighting7	0.392	0.391	1.558	0.182	○
MALLAT	1.479	0.027	2.805	0.017	○
MedicalImages	0.352	0.316	1.219	0.184	○
Motes	0.732	0.093	1.714	0.063	○
MoteStrain	0.732	0.093	1.714	0.063	○
OliveOil	0.382	0.280	2.037	0.156	●
OSULeaf	0.626	0.448	1.545	0.246	
plane	−0.049	0.009	1.489	0.000	●
SonyAIBORobotSurfaceII	0.815	0.065	2.050	0.022	
SonyAIBORobotSurface	0.799	0.044	2.093	0.026	○
StarLightCurves	1.050	0.090	4.353	0.126	○
SwedishLeaf	0.969	0.235	3.451	0.220	○
Symbols	0.832	0.030	1.852	0.017	●
synthetic_control	1.400	0.020	3.940	0.010	●
Trace	0.035	0.025	1.556	0.000	●
TwoLeadECG	0.399	0.003	1.846	0.001	●
Two_Patterns	1.007	0.001	4.901	0.000	●
uWaveGestureLibrary_X	0.773	0.234	2.501	0.222	○
uWaveGestureLibrary_Y	0.627	0.314	2.068	0.258	○
uWaveGestureLibrary_Z	0.660	0.317	2.100	0.279	○
wafer	0.307	0.005	1.154	0.004	○
WordsSynonyms	0.659	0.346	2.536	0.182	●
yoga	0.595	0.119	1.752	0.058	○

Table 4.2: Summary of datasets. Hubness measurements and the performance of my approach compared to the baseline are also shown.

	10% initially labeled			20% initially labeled			90%
$r_{\text{misclass.}}$	1-NN[2]	SS-1-NN[3]	SS-SLINK[4]	1-NN	SS-1-NN	SS-SLINK	1-NN[5]
50words	0.422	○ 0.429	● 0.398	0.332	○ 0.336	● 0.309	0.202
Adiac	0.589	○ 0.621	0.590	0.493	○ 0.518	0.493	0.346
Beef	0.632	○ 0.657	○ 0.671	0.590	○ 0.616	○ 0.623	0.483
Car	0.449	○ 0.471	○ 0.458	0.366	○ 0.401	0.366	0.229
CBF	0.003	0.003	● 0.000	0.001	● 0.000	● 0.000	0.000
Chlor...on	0.369	○ 0.372	● 0.070	0.242	○ 0.245	● 0.035	0.003
CinC_...so	0.027	● 0.021	● 0.002	0.010	● 0.005	● 0.000	0.000
Coffee	0.389	○ 0.431	● 0.375	0.286	0.285	0.277	0.030
Cricket_X	0.442	○ 0.467	● 0.430	0.318	○ 0.326	● 0.298	0.166
Cricket_Y	0.428	○ 0.455	● 0.408	0.314	○ 0.331	● 0.290	0.185
Cricket_Z	0.429	○ 0.451	● 0.407	0.318	○ 0.325	● 0.294	0.165
Diato...on	0.020	● 0.017	● 0.017	0.012	0.012	○ 0.013	0.006
ECG200	0.211	○ 0.229	○ 0.218	0.174	○ 0.195	○ 0.195	0.153
ECGFiveDays	0.062	● 0.045	● 0.021	0.031	● 0.023	● 0.016	0.011
FaceFour	0.215	0.210	● 0.185	0.144	○ 0.164	● 0.121	0.059
FacesUCR	0.076	● 0.071	● 0.056	0.048	● 0.045	● 0.038	0.014
fish	0.370	○ 0.410	○ 0.424	0.299	○ 0.324	○ 0.323	0.203
Gun_Point	0.099	0.094	● 0.045	0.064	● 0.047	● 0.047	0.023
Haptics	0.627	○ 0.676	○ 0.701	0.594	○ 0.636	○ 0.662	0.521
InlineSkate	0.646	○ 0.686	○ 0.678	0.571	○ 0.598	○ 0.607	0.449
Italy...nd	0.061	0.063	0.064	0.053	○ 0.057	0.053	0.047
Lighting2	0.319	○ 0.329	0.313	0.263	0.265	○ 0.278	0.121
Lighting7	0.456	○ 0.497	○ 0.507	0.361	○ 0.390	○ 0.405	0.232
MALLAT	0.031	○ 0.037	○ 0.041	0.024	○ 0.026	○ 0.030	0.009
Medic...es	0.364	○ 0.375	○ 0.376	0.304	○ 0.315	0.304	0.205
Motes	0.104	○ 0.122	○ 0.128	0.084	○ 0.098	○ 0.105	0.047
MoteStrain	0.104	○ 0.122	○ 0.128	0.084	○ 0.098	○ 0.105	0.047
OliveOil	0.303	○ 0.319	0.294	0.247	● 0.230	● 0.218	0.083
OSULeaf	0.499	○ 0.528	0.502	0.409	○ 0.424	0.407	0.239
plane	0.048	● 0.040	● 0.039	0.008	● 0.001	● 0.000	0.000
SonyA...II	0.091	0.091	0.094	0.057	0.059	0.057	0.012
SonyA...ce	0.057	● 0.052	○ 0.106	0.043	0.046	○ 0.064	0.023
StarL...es	0.097	○ 0.120	○ 0.203	0.086	○ 0.102	○ 0.160	0.071
SwedishLeaf	0.293	○ 0.324	○ 0.367	0.226	○ 0.249	○ 0.267	0.157
Symbols	0.038	● 0.031	● 0.022	0.028	● 0.026	● 0.021	0.019
synth...ol	0.037	○ 0.050	● 0.025	0.019	0.017	● 0.015	0.008
Trace	0.148	● 0.039	● 0.000	0.027	● 0.000	● 0.000	0.000
TwoLeadECG	0.008	● 0.003	● 0.001	0.003	● 0.001	● 0.001	0.001
Two_Patterns	0.010	● 0.000	● 0.000	0.001	● 0.000	● 0.000	0.000
uWave..._X	0.253	○ 0.278	○ 0.282	0.226	○ 0.244	○ 0.253	0.198
uWave..._Y	0.333	○ 0.355	○ 0.368	0.304	○ 0.319	○ 0.321	0.258
uWave..._Z	0.332	○ 0.360	○ 0.379	0.307	○ 0.329	○ 0.346	0.259
wafer	0.008	○ 0.010	○ 0.009	0.005	○ 0.006	○ 0.006	0.003
Words...ms	0.398	○ 0.406	● 0.371	0.317	○ 0.327	● 0.294	0.187
yoga	0.138	○ 0.146	○ 0.151	0.105	○ 0.113	○ 0.112	0.053

Table 4.3: Summary of experiments. Statistically significant ($p < 0.5$) differences from supervised 1-NN are indicated by ● and ○ symbols. Smaller $r_{\text{misclass.}}$ misclassification rate is better.

[2] Baseline supervised 1-NN classifier trained on the initially labeled instances.

[3] Instance-based self-training classifier introduced by C. A. Ratanamahatana and Wanichsan (2008).

[4] The proposed approach: semi-supervised clustering, then class label assignment.

[5] Upper bound on self training, where an *oracle* labels the unlabeled training instances.

5 Conclusions

In this work, I focused on the semi-supervised time-series classification problem. This problem is especially important in domains where the usage of data mining and recognition systems is only emerging, because in those areas, labeled data is scarce. Semi-supervised methods can serve to greatly reduce required human expert effort in new time-series classification applications.

In Chapter 2, I reviewed some of the literature concerning semi-supervised learning a time-series data mining. These topics are usually absent from most Hungarian university curricula.

In Chapter 3, I proposed a novel instance-based semi-supervised classification method for time series, *constrained SLINK with DTW*. This approach is based on constrained clustering and Dynamic Time Warping and interacts well with the properties of time series databases, such as hubness. I also proposed *transitive hubness* as method to approximate the expected performance of my algorithm on a given time series database.

In Chapter 4, I showed that my approach significantly outperforms a state-of-the-art instance-based supervised and semi-supervised time-series classifier. I also observed a correlation between transitive hubness in a database and the performance of my algorithm.

5.1 Further work and open questions

Semi-supervised classification of time series is a difficult task: compared to vectors, time series have unusual structure, such as high correlation between consecutive measurements. Their high intrinsic dimensionality gives rise to good and bad hubness, which can 'confuse' algorithms as well as help them.

In supervised learning, it is possible to localise regions in the database where the bad hubs are (K. Buza et al., 2011b). By selectively ignoring bad hubs, classification accuracy can be improved. *Is there a way to recognise potential bad hubs in unlabeled data?* Perhaps this could improve accuracy of semi-supervised learning in datasets where there are many strong bad hubs.

In this works, I considered instance-based learning for time series with Dynamic Time Warping, which is a state-of-the-art supervised classifier for time series (Ding et al., 2008). *Are there other, better representations and distance functions for semi-supervised classification of time series?*

References

Abney, Steven P. (2004). ‘Understanding the Yarowsky Algorithm’. In: *Computational Linguistics* 30.3, pp. 365–395 (cit. on p. 12).

Aha, David W., Dennis F. Kibler and Marc K. Albert (1991). ‘Instance-Based Learning Algorithms’. In: *Machine Learning* 6, pp. 37–66 (cit. on p. 12).

Basu, Sugato, Mikhail Bilenko and Raymond J. Mooney (2004). ‘A probabilistic framework for semi-supervised clustering’. In: *KDD*, pp. 59–68 (cit. on p. 15).

Blum, Avrim and Tom M. Mitchell (1998). ‘Combining Labeled and Unlabeled Data with Co-Training’. In: *COLT*, pp. 92–100 (cit. on p. 13).

Buza, Krisztian Antal (2011). ‘Fusion Methods for Time-Series Classification’. PhD thesis, pp. 1–144 (cit. on p. 18).

Buza, Krisztian, Alexandros Nanopoulos and Lars Schmidt-Thieme (2011a). ‘INSIGHT: Efficient and Effective Instance Selection for Time-Series Classification’. In: *PAKDD (2)*. Vol. 6635, pp. 149–160 (cit. on p. 21).

— (2011b). ‘IQ estimation for accurate time-series classification’. In: *CIDM*, pp. 216–223 (cit. on p. 35).

Castelli, Vittorio and Thomas M. Cover (1996). ‘The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter’. In: *IEEE Transactions on Information Theory* 42.6, pp. 2102–2117 (cit. on p. 10).

Chen, Lei and Raymond T. Ng (2004). ‘On The Marriage of Lp-norms and Edit Distance’. In: *VLDB*, pp. 792–803 (cit. on p. 19).

Cozman, Fabio Gagliardi, Ira Cohen and Marcelo Cesar Cirelo (2003). ‘Semi-Supervised Learning of Mixture Models’. In: *ICML*, pp. 99–106 (cit. on p. 10).

Culp, Mark and George Michailidis (2007). ‘An iterative algorithm for extending learners to a semisupervised setting’. In: *The 2007 Joint Statistical Meetings (JSM)* (cit. on p. 12).

Dara, R., S.C. Kremer and D.A. Stacey (2002). ‘Clustering unlabeled data with SOMs improves classification of labeled real-world data’. In: *Neural Networks, 2002. IJCNN ’02. Proceedings of the 2002 International Joint Conference on.* Vol. 3, pp. 2237 –2242 (cit. on p. 11).

Davidson, Ian and S. S. Ravi (2005). ‘Clustering with Constraints: Feasibility Issues and the k-Means Algorithm’. In: *SDM* (cit. on pp. 14, 15).

— (2007). ‘The complexity of non-hierarchical clustering with instance and cluster level constraints’. In: *Data Min. Knowl. Discov.* 14.1, pp. 25–61 (cit. on p. 15).

Demiriz, Ayhan, Kristin Bennett and Mark J. Embrechts (1999). ‘Semi-Supervised Clustering Using Genetic Algorithms’. In: *In Artificial Neural Networks in Engineering (ANNIE-99)*, pp. 809–814 (cit. on p. 11).

Dempster, A. P., N. M. Laird and D. B. Rubin (1977). ‘Maximum likelihood from incomplete data via the EM algorithm’. In: *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B* 39.1, pp. 1–38 (cit. on p. 10).

Ding, Hui, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang and Eamonn J. Keogh (2008). ‘Querying and mining of time series data: experimental comparison of representations and distance measures’. In: *PVLDB* 1.2, pp. 1542–1552 (cit. on pp. 3, 35).

Elworthy, David (1994). ‘Does Baum-Welch Re-estimation Help Taggers?’ In: *ANLP*, pp. 53–58 (cit. on pp. 10, 23).

Goldberg, Andrew B., Xiaojin Zhu, Alex Furger and Jun-Ming Xu (2011). ‘OASIS: Online Active Semi-Supervised Learning’. In: *AAAI* (cit. on p. 14).

Haffari, Gholamreza and Anoop Sarkar (2007). 'Analysis of semi-supervised learning with the Yarowsky algorithm'. In: *23rd Conference on Uncertainty in Artificial Intelligence (UAI)* (cit. on p. 12).

Hjaltason, Gísli R. and Hanan Samet (2003). 'Index-driven similarity search in metric spaces'. In: *ACM Trans. Database Syst.* 28.4, pp. 517–580 (cit. on p. 19).

Keogh, Eamonn J. (2002). 'Exact Indexing of Dynamic Time Warping'. In: *VLDB*, pp. 406–417 (cit. on p. 19).

Keogh, Eamonn J. and Chotirat Ann Ratanamahatana (2005). 'Exact indexing of dynamic time warping'. In: *Knowl. Inf. Syst.* 7.3, pp. 358–386 (cit. on p. 3).

Keogh, Eamonn J., Li Wei, Xiaopeng Xi, Sang-Hee Lee and Michail Vlachos (2006a). 'LB_Keogh Supports Exact Indexing of Shapes under Rotation Invariance with Arbitrary Representations and Distance Measures'. In: *VLDB*, pp. 882–893 (cit. on p. 19).

Keogh, Eamonn J., Xiaopeng Xi, Li Wei and Chotirat Ann Ratanamahatana (2006b). *The UCR Time Series Classification/Clustering Homepage*. URL: http://www.cs.ucr.edu/~eamonn/time_series_data/ (cit. on pp. 3, 29).

Kestler, Hans A., Johann M. Kraus, Günther Palm and Friedhelm Schwenker (2006). 'On the Effects of Constraints in Semi-supervised Hierarchical Clustering'. In: *ANNPR*. Vol. 4087, pp. 57–66 (cit. on p. 15).

Kim, Sang-Wook, Sanghyun Park and Wesley W. Chu (2001). 'An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases'. In: *ICDE*, pp. 607–614 (cit. on p. 19).

Klein, Dan, Sepandar D. Kamvar and Christopher D. Manning (2002). 'From Instance-level Constraints to Space-Level Constraints: Making the Most of Prior Knowledge in Data Clustering'. In: *ICML*, pp. 307–314 (cit. on p. 15).

Li, Ming and Zhi-Hua Zhou (2005). 'SETRED: Self-training with Editing'. In: *PAKDD*. Vol. 3518, pp. 611–621 (cit. on p. 12).

McCallum, Andrew and Kamal Nigam (1998). 'Employing EM and Pool-Based Active Learning for Text Classification'. In: *ICML*, pp. 350–358 (cit. on p. 14).

Miyamoto, Sadaaki and Akihisa Terami (2010). 'Semi-supervised agglomerative hierarchical clustering algorithms with pairwise constraints'. In: *FUZZ-IEEE*, pp. 1–6 (cit. on p. 15).

Nguyen, Minh Nhut, Xiaoli Li and See-Kiong Ng (2011). 'Positive Unlabeled Leaning for Time Series Classification'. In: *IJCAI*, pp. 1421–1426 (cit. on p. 23).

Quinlan, J. Ross (1986). 'Induction of Decision Trees'. In: *Machine Learning* 1.1, pp. 81–106 (cit. on p. 14).

Radovanovic, Milos, Alexandros Nanopoulos and Mirjana Ivanovic (2010a). 'Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data'. In: *Journal of Machine Learning Research* 11, pp. 2487–2531 (cit. on p. 20).

— (2010b). 'Time-Series Classification in Many Intrinsic Dimensions'. In: *SDM*, pp. 677–688 (cit. on pp. 20, 21).

Ratanamahatana, Chotirat (Ann) and Eamonn J. Keogh (2005). 'Three Myths about Dynamic Time Warping Data Mining'. In: *SDM* (cit. on p. 18).

Ratanamahatana, Chotirat Ann and Dachawut Wanichsan (2008). 'Stopping Criterion Selection for Efficient Semi-supervised Time Series Classification'. In: *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*. Vol. 149, pp. 1–14 (cit. on pp. 22, 23, 30, 33).

Reuter, Timo, Philipp Cimiano, Lucas Drumond, Krisztian Buza and Lars Schmidt-Thieme (2011). 'Scalable Event-Based Clustering of Social Media Via Record Linkage Techniques'. In: *ICWSM* (cit. on p. 15).

Seeger, Matthias (2001). *Learning with Labeled and Unlabeled Data*. Tech. rep. University of Edinburgh (cit. on p. 9).

Settles, Burr (2012). *Active Learning*. Morgan & Claypool Publishers (cit. on p. 14).

Singh, Aarti, Robert D. Nowak and Xiaojin Zhu (2008). 'Unlabeled data: Now it helps, now it doesn't'. In: *NIPS*, pp. 1513–1520 (cit. on p. 10).

Stefan, Alexandra, Vassilis Athitsos and Gautam Das (2012). 'The Move-Split-Merge Metric for Time Series'. In: *IEEE Transactions on Knowledge and Data Engineering* 99.PrePrints (cit. on p. 19).

Wagstaff, Kiri and Claire Cardie (2000). 'Clustering with Instance-level Constraints'. In: *ICML*, pp. 1103–1110 (cit. on p. 15).

Wagstaff, Kiri, Claire Cardie, Seth Rogers and Stefan Schrödl (2001). 'Constrained K-means Clustering with Background Knowledge'. In: *ICML*, pp. 577–584 (cit. on p. 15).

Wei, Li and Eamonn J. Keogh (2006). 'Semi-supervised time series classification'. In: *KDD*, pp. 748–753 (cit. on p. 22).

White, Allan P. and Wei Zhong Liu (1994). 'Bias in Information-Based Measures in Decision Tree Induction'. In: *Machine Learning* 15.3, pp. 321–329 (cit. on p. 14).

Yarowsky, David (1992). 'Word-Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora'. In: *COLING*, pp. 454–460 (cit. on p. 12).

Yi, Byoung-Kee, H. V. Jagadish and Christos Faloutsos (1998). 'Efficient Retrieval of Similar Time Sequences Under Time Warping'. In: *ICDE*, pp. 201–208 (cit. on p. 19).

Zhong, Shi (2005). 'Semi-Supervised Sequence Classification With Hmms'. In: *IJPRAI* 19.2, pp. 165–182 (cit. on p. 23).

Zhu, Qiang, Gustavo E. A. P. A. Batista, Thanawin Rakthanmanon and Eamonn J. Keogh (2012). 'A Novel Approximation to Dynamic Time Warping allows Anytime Clustering of Massive Time Series Datasets'. In: *SDM*, pp. 999–1010 (cit. on p. 20).

Zhu, Xiaojin (2006). *Semi-Supervised Learning Literature Survey* (cit. on p. 9).

— (2009). *Semi-Supervised Learning*. Tutorial at PASCAL Machine Learning Summer School (MLSS). Video recording available at http://videolectures.net/mlss09us_zhu_ssl/ (cit. on p. 14).

Zhu, Xiaojin and John D. Lafferty (2005). 'Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning'. In: *ICML*. Vol. 119, pp. 1052–1059 (cit. on p. 23).