NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS
FACULTY OF MATHEMATICS

Master in Logic, Algorithms and Computation

MASTER THESIS

Transfer-Kmeans: a new supervised clustering approach

Pelagia P. Teloni
A.M.: 201207

Supervisors: Aris Pagourtzis, Associate Professor NTUA
Anastasia Krithara, Associate Researcher NCSR Demokritos

Comittee: Aris Pagourtzis, Associate Professor NTUA
Stathis Zachos, Professor Emeritus NTUA
Dimitris Fotakis, Assistant Professor NTUA

ATHENS

JULY 2017

ABSTRACT

Supervised and unsupervised learning are two fundamental learning schemes whose difference lies in the presence and absence of a supervisor (i.e. entity which provides examples) respectively. On the other hand, transfer learning aims at improving the learning of a task by using auxiliary knowledge. The goal of this thesis was to investigate how the two fundamental paradigms, supervised and unsupervised learning, can collaborate in the setting of transfer learning. As a result, we developed transfer-Kmeans, a transfer learning variant of the popular Kmeans heuristic.

The proposed method enhances the unsupervised nature of Kmeans, using supervision from a different but related context as a seeding technique, in order to improve the heuristic's performance towards more meaningful results. We provide approximation guarantees based on the nature of the input and we experimentally validate the benefits of the proposed method using documents as a real-world example.

SUBJECT AREA: Machine Learning

KEYWORDS: clustering, transfer learning, domain adaptation, density ratio estimation, natural language processing

ΠΕΡΙΛΗΨΗ

Η επιτηρούμενη και η μη-επιτηρούμενη μάθηση είναι δύο θεμελιώδη σχήματα μάθησης, των οποίων η διαφορά έγγυται στην παρουσία και απουσία ενός καθηγητή (δηλαδή μιας οντότητας που παρέχει παραδείγματα) αντίστοιχα. Από την άλλη πλευρά, η μεταφορά μάθησης είναι μια ιδέα που στοχεύει να βελτιώσει την μάθηση ενός έργου χρησιμοποιώντας βοηθητική γνώση. Ο στόχος της παρούσας διπλωματικής είναι να διερευνήσει πως αυτά τα δύο θεμελιώδη παραδείγματα μάθησης, επιτηρούμενη και μη-επιτηρούμενη μάθηση, μπορούν να συνεργαστούν στο πλαίσιο της μεταφοράς μάθησης. Ως αποτέλεσμα, αναπτύξαμε τη μέθοδο transfer-Kmeans, μια παραλλαγή της δημοφιλής ευριστικής μεθόδου Kmeans, που βασίζεται στην μεταφορά μάθησης.

Η προτεινόμενη μέθοδος εμπλουτίζει την μη-επιτηρούμενη φύση του Kmeans χρησιμοποιώντας επιτήρηση από ένα διαφορετικό αλλά σχετικό χώρο ως τεχνική αρχικοποίησης των συστάδων, με σκοπό να βελτιώσει την απόδοση της ευριστικής αυτής μεθόδου. Παρέχουμε προσεγγιστικές εγγυήσεις σύμφωνα με την φύση της εισόδου και επαληθεύουμε πειραματικά τα οφέλη του transfer-Kmeans χρησιμοποιώντας κείμενα σε φυσική γλώσσα ως ρεαλιστική εφαρμογή.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Μηχανική Μάθηση

ΛΕΞΕΙΣ - ΚΛΕΙΔΙΑ: συσταδοποίηση, μηχανική μάθηση, προσαρμογή τομέα, εκτίμηση λόγου κατανομών, επεξεργασία φυσικής γλώσσας

PREFACE

This thesis would not have been possible without the support and generosity of several individuals. Foremost among those are my supervisors, dr. Aris Pagourtzis and dr. Anastasia Krithara, who have provided great inspiration, ideas, comments and guidance on a variety of topics, both within and outside of the realm of Algorithms and Machine Learning. In addition, I would like to extend my appreciation to my family: Tasoula, Panos, Eirini and Dimitris. Your patience and support was of paramount importance for me to complete this work, which I dedicate to you. Finally, I would like to thank my dear friends Yiannis, Giorgos, Kamil and Diego, for providing me fruitful distraction on other aspects of human condition.

Table of Contents

1 Introduction

alter techno is more reliable than *alter ego*

— Pascal Chabot

"*Rotwang, give the Machine-Man the likeness of that girl.*" This is the moment where, in the 1927 film *Metropolis* by Fritz Lang, the powerful ruler of the city Joh Fredersen instructs the mad scientist Rotwang to give his evil robot the appearance of Maria, an influential peace-preaching proletariat woman, in order to use her reputation to spread corruption among the suffering workers. This movie is an emblematic instance of what Isaac Asimov called "the Frankenstein complex", the belief that science could produce something that it could not control. Asimov, one of the fathers of the science fiction genre, envisioned an era where humanity would be served by robots. After all, the word "robot", coined around 1920 from the Czech author Karel Capek or his brother Josef, comes from the Czech word *robota*, which means "forced labour, compulsory service, drudgery". However Asimov, as a talented dramaturgist, involved in his narrative the robots' attempt to destroy their creators.

This plot archetype was and still is very popular, since it captures the audience's imagination and deepest fears. However, it has not remained just a fictional construction. With the publicised achievements of Artificial Intelligence (AI) research, the Frankenstein complex has infiltrated into the real life, having a widespread scientific and social impact. Many scientists have raised concerns about the risks posed by future AI technology and the majority of the general public feel more and more alienated from this technological progress.

The negative opinion about AI exhibits three important characteristics: it is justifiable, useful but also misguided. For the first characteristic, we can easily consider two reasons. To begin with, AI's scandalous goal is to simulate one of the attributes that presumably place humans at the top of the species' pyramid: human cognition. It is very difficult for us to conceive other entities sharing our place in this pyramid. However, the notion of difficulty here does not relate with that of impossibility; it simply means that such a conception is outside of our imaginative reach, the same way we cannot imagine a fourth dimension. Another reason why this negative opinion is justifiable can be the existence of an emotional and cultural trauma: humanity has a long history of science abuse, for instance with the advent of nuclear energy. For the second characteristic, proponents of the Frankenstein complex actually offer themselves

as an example of the scientific morality debate: how human values resonate in science? Such a debate must remain open and active in order to question the purpose of the ongoing technological progress and redefine frequently the relation of science with the rest of human endeavours. However, the negative view of AI runs the risk of becoming a parochialism since such a quest, to simulate the human mind, entails one of the most daunting and existential questions: *what is the human mind?*

Reflection on this question can be traced back to antiquity and it has been one of the meeting points of philosophical and scientific approaches. Many mind theories were created, yet none of them radically changed our lives. It was only in the early 1930s that a variant of this question prepared the ground for a jump to universality[1]: *is the human mind a machine?* That variant was not something new. In the 16th century, during the Scientific Revolution, the philosophical doctrine of Mechanism came to prevalence, manifesting itself in the works of Isaac Newton and René Descartes and holding the anti-teleological view that all physical bodies (including humans) can be completely described by mechanistic laws of motion or in a later modification, that all "vital" phenomena can be explained as physical and chemical facts. This doctrine inevitably gave birth to a sub-field called *Anthropic Mechanism* where everything about human beings can be explained in mechanical terms, as can everything about e.g. a clockwork. For the body, most mechanist theories could hold their claim, but what about the mind? Is the mind a machine, yet that complex that we are still unable to describe? Or is there something more, a so-called *spirit* that cannot be reduced to a quantitative reality? This sub-field is still active, having proponents in both sides and their fruitful debate mostly focus on the abstract notions of consciousness and free will. Taking a brief look into this debate and with a dangerous oversimplification, the argument of anti-mechanists is that a mechanistic mind view is incompatible with commonsense intuitions. On the other side, the answer of mechanists is that commonsense intuitions are simply wrong or such an incompatibility is a paralogism and does not exist.

Although the mechanical aspect of the mind had been contemplated for so long, what changed in the 1930s? It was the launch of the Computability theory, the glue that unified all the necessary elements for the birth of the discipline that we now know as computer science and paved the way towards the Digital Era we live in. The pioneers of this theory was a group of mathematicians and logicians, among them the predominant figures of Kurt Gödel, Alonzo Church and Alan Turing, who tried to explain the human experience of computation and suggested how artificial computing devices should be built. They formalized the notion of effective computability as a fixed finite procedure, assuming that only a finite number of states of mind is "taken into account" at each stage [CPS13]. The existence of a device simulating this notion was proven constructively by Turing and it was called the Universal Turing machine, an automaton that operated on logical principles. The debate whether the mind is a machine was still ongoing, but there was a consensus that the mind is more powerful than any given Turing machine and Gödel's incompleteness results[2], although not applicable

[1] The *jump to universality* as described in [Deu11] is a concept that describes the situation where a solution for a specific problem becomes useful for its own sake and can customize itself as a solution to other problems.

[2] Informally, Gödel showed that there exist true mathematical statements that are not provable (a proof cannot be computed). In Turing's view, these results also showed that there is no formal system of logic that can contain all possible methods of proof.

to human reason, established limitations of such a mathematical formalism.

What is in the human mind that could not be expressed as a Turing machine? Focusing only on human reasoning, Turing thought that it was the notion of *intuition*. The human mind (at least that of an idealized mathematician), although at every stage is identical to some Turing machine, when it searches around to find new solutions and methods of proof, it transforms itself from one Turing machine to some other Turing machine. Intuition was regarded as the non-mechanical process of choosing this other Turing machine. According to Turing, a machine that can transform itself is in effect a machine with the ability to *learn*, yet he provided no precise ways on how to accomplish such a transformation. Therefore after computability, learning was the next step along the path of exploring the human mind. From the 1980s, several mathematical models have attempted to explain this phenomenon, such as the PAC learning theory [Val84] which formalized the concept of an efficient learner. But unlike Turing machines and computability, none of them succeeded a jump to universality yet.

Learning is an experience that is not fully understood in order to be properly defined. Precisely because of its abstract flavour, many philosophical attempts exist such as this by Ellen Fridland in [MnSDB15] where she defines learning as *a process, where as a result of experience or reasoning, the behaviour, mental processing or representations of subjects change in some way that contributes to the satisfaction of their goal(s).* The terms in this definition stay unqualified to preserve a broad range of interpretations, yet remaining informative in the sense that learning can be described by two conditions: flexibility and success. The flexibility condition is expressed through a change and the success condition through the satisfaction of a goal. A particular kind of flexibility is *transferability*: a change that is not bound to a specific context but it can be applied in various settings and circumstances. For example, in order to learn the concept BLUE one needs to be able to think various blue things: a blue sky, a blue dress or a blue chair. Transfer learning, albeit in its infant mechanistic variant, will be the main focus in this work.

Due to such a definitional challenge, it is no surprise that mathematicians have not yet unanimously agreed on a formulation about learning. However, they are starting to get influenced by another rapidly developing field: neuroscience. Why just talking about the human mind, when we can have a look at how the human brain works ? The study of the human neocortex - with its complex networks of neurons, how they generate and interact with each other and their large-scale behaviour by integrating inputs from thousands of synapses - has inspired new inventions and is considered to be the most promising way towards a formation of AI [HA16].

Despite all the ongoing research so far, little progress has been made towards machines that think. There exist ways to create programs that detect spam e-mails, recommend related items or perform face recognition, but in fact, we don't even have a test that could validate that it is a program that generates the knowledge and not the programmer. Although AI has not reach its goal, it's getting closer. And this goal no longer seems scandalous but as it has been discussed so far, it reflects our efforts to understand us better. As a counter-point of the

Frankenstein complex, it is important to observe that even in the present stage of technological evolution, the relation between humans and machines is no longer the master-slave bipolar but it can be described as a collaboration. Gilbert Simondon demonstrated this by using memory as an example: human memory is selective and stores only emotionally-trigerring details whereas machine's memory can retain every detail but is incapable of selectivity [Cha13]. Accepting us humans as "translators of information between machines" will be a successful manifestation of the non-alienating positivity and productivity of a technical mentality [Sim06], a mode of knowledge that has and will enhance our ability to find explanations even further. Such an endeavour will be full of errors but as David Deutsch mentions in [Deu11] "*Without error-correction all information processing, and hence all knowledge-creation, is necessarily bounded. Error-correction is the beginning of infinity.*"

1.1 Thesis goal

Computational learning theory is the field that deals with the notion of learning from a theoretical computer science perspective. Its two main branches are: statistical and algorithmic learning theory. In statistical learning theory, as mentioned in [Han00], learning is considered as the stochastic process of generalizing from a random finite sample of data. In algorithmic learning theory, the sample of data is not assumed to be random. Its main focus is learning in the limit: the bigger the sample, the better the learning. In this thesis, we will implicitly follow the statistical approach.

Standard problems in computational learning theory can be classified into two fundamental learning paradigms: supervised and unsupervised learning. Their core difference can be understood by the existence of an entity called *teacher*: in supervised learning, there exists a teacher which provides examples of some concept to a *learner* and the learner's goal is to learn the concept. For example, let's assume that we want to learn a function $f : A \to B$ so that for any input $x \in A$ we would like to predict the output $f(x) \in B$. In supervised learning, the teacher will provide us a finite set of labeled examples $\{x_i, f(x_i)\}_{i=1}^{m} \subset A \times B$, which is a set of desired input-output pairs. Our goal is to predict the value of f on an unseen instance $x_{m+1} \in A$. In unsupervised learning, the teacher does not exist. Therefore the learner is given as input a finite set of unlabeled examples where the learning source is the intrinsic structure of inputs and the goal of the learner is to find hidden patterns in this structure, for instance by clustering or dimensionality reduction of the inputs. Let us refer to the input provided to the learner as *train set* and the instances on which the learner makes predictions as *test set*.

The combination of the above paradigms has given rise to interesting learning methods such as semi-supervised and transductive learning. What these methods have in common is that some part of the input is provided with a teacher while the other part is not. The input consists of labeled and unlabeled examples. However they differ in their goal: in semi-supervised learning the goal, as in supervised learning, is given the input to learn a general rule on unseen instances. In transductive learning, the goal is to predict the labels of the unlabeled part of the input. Therefore in transductive learning, the test set is available during training

time.

So far it has been implied that train and test sets are drawn from the same set $A \times B$, or more generally from the same fixed but unknown distribution (in this way, we can model uncertainty in the examples, e.g. a teacher can make mistakes). By removing this assumption and allowing train and test sets to come from different distributions, we enter the field of transfer learning. Introduced in 1996 by Thrun [Thr96], transfer learning aims to apply the knowledge learned in one context to enhance the learning in a different but somewhat related context. For instance, if we already know a way to separate documents about football and documents about cars, can we use this knowledge to separate documents about hockey and motorcycles?

As a sub-field of machine learning, transfer learning has gained significant attention in recent years. To begin with, it is a new learning paradigm that allows the formulation of many problems as well as enhancing the capabilities of already existing machine learning approaches. In addition, it has a particular practical importance: in classical machine learning implementations, the work of the teacher is done by humans. For instance, a set of documents have been manually annotated as hockey and motorcycles, which are then provided as input to a program in order to learn to separate between these two concepts. However, the amount of available data in our days is so big or become outdated so fast that annotating all of them manually is a slow, expensive and error-prone process. To address such constraints, we could re-use already annotated data, although from a somewhat different context, e.g. the football-cars domain.

The goal of this thesis was to investigate how the two fundamental paradigms, supervised and unsupervised learning, can collaborate in the setting of transfer learning. As a result, we developed tr-Kmeans, a transfer learning variant of the popular Kmeans heuristic, also called Lloyd's method [Llo82]. This heuristic is a typical example of unsupervised learning. With supervision from a different but related context, we attempted to improve the heuristic's performance towards more meaningful results.

1.2 Thesis structure

The rest of thesis is organized as follows. In chapter 2 we offer a short survey of the existing work on transfer learning. Section 2.1 provides some useful notations alongside a high-level definition on transfer learning and sections 2.2-2.3 review the basic problem settings and learning strategies for transfer learning and based on this categorization some known transfer learning algorithms are discussed.

In chapter 3, we provide some background notations and definitions that will be used in this thesis. In section 3.1 we formally define the problem under investigation, namely the Domain Adaptation problem and we review some known theoretical results on the generalization capabilities of a learner. In section 3.2 we provide the notion of context-similarity that we will adopt in this work and discuss on existing approaches to compute this measure. Finally in section 3.3 we elaborate on the pre-processing step of human-created data, a crucial step

in our experimental analysis provided in chapter 5.

In chapter 4 we discuss in detail our proposed approach, the tr-Kmeans heuristic. In section **??** we argue how and where it would be valuable to transfer knowledge and in section **??** we provide some approximation guarantees of the method.

In chapter 5 we conduct some experiments to illustrate the value of our method, both on synthetic 5.2 and real 5.3 datasets. Finally in chapter 6 we summarize the observed conclusions and we propose directions of future research.

2 Related Work

This chapter offers a survey of the existent work related to transfer learning. In section 2.1 we provide some useful notations and a high-level definition on transfer learning. In section 2.2 we review the basic problem settings and learning strategies for transfer learning and based on this categorization we discuss some well known transfer learning algorithms. Finally in section 2.3 we focus on our setting, namely transductive transfer learning, and present a by-no-means-complete literature review on this subdomain.

2.1 Useful Notation and Definitions

Following the notation in [PY10], let us define a *domain* to be the tuple $D = \{I, P(I)\}$ where I is an input space (namely a feature set[1]) and $P(I)$ is a marginal probability distribution on this set. A domain can be considered as a generator of inputs. Given a specific domain D, a *task* is a tuple $T = \{O, P(O|I)\}$ where O is the set of all possible outputs (often called labels or information classes) and $P(O|I)$ is a conditional probability distribution which models what needs to be learned. It is worth mentioning that $P(O|I)$ can contain $f(x) = y$, $x \in I$ and $y \in O$ as a special case (i.e. $P(f(x)|x) = 1$). Finally let us define a *context* to be the tuple $C = \{D, T\}$ which is governed by the joint distribution $P(I, O) = P(I) \cdot P(O|I)$. We are now ready to give an informal definition on transfer learning:

Definition 2.1.1. *[Transfer Learning - informal] Given a source context C_s and a target context C_t, where $C_s \neq C_t$, transfer learning aims to improve the learning of the target conditional distribution $P_t(O_t|I_t)$ using the knowledge obtained from C_s.*

So a learner, given inputs from the target domain D_t and knowledge learned from at least one source context C_s, will produce a so-called hypothesis function $h : I_t \to O_t$ which will best approximate the target conditional distribution $P_t(O_t|I_t)$. A more inclusive discussion of learning is postponed until chapter 3, but for the moment let's observe the following: what the learner actually does is based on the input, it searches a function space $H = \{f|f : I_t \to O_t\}$

[1] In literature, the input or instance space is often intertwined with the feature space. In this thesis, we also follow this simplification. However, in [BDBCP07] they separate the actual input from its feature representation via a representation function, effectively formalizing the pre-processing in typical machine learning applications.

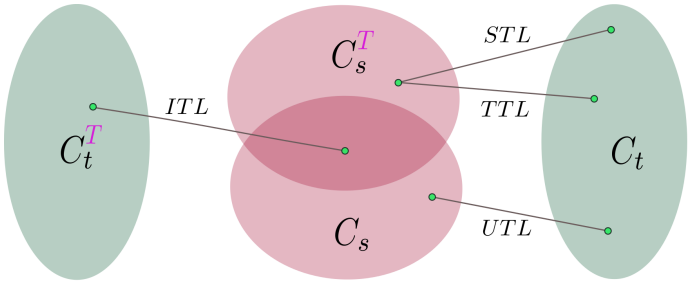


Figure 2.1.1: The four subsettings of transfer learning: green circles denote the target context and pink circles the source context. The superscript T denotes the existence of labels (teacher) in each context.

to find the best candidate function $h \in H$ for T_t. Given that $|H| = |O_t|^{|I_t|}$, learning can be impossible in practice. Therefore, we restrict the search space to be $H_A \subset H$, where A is a set of parameters and $H_A = \{f_a | a \in A \text{ and } f_a \in H\}$. Intuitively A can be thought of as the *inductive bias* or the assumptions of the learner about the true distribution of the training set, e.g. the (unknown) function to be learned is a polynomial. As explained in [Mit80], a learner without bias cannot generalize and will only memorize the given inputs. We will refer to the restricted search space H_A as the *hypothesis space* of the learner. It is worth mentioning that there is no guarantee that the target function f will be a member of H_A. Transfer learning aims to improve the search inside H_A.

2.2 Transfer Learning

Whether labeled examples (i.e. a teacher) are given in each context, transfer learning can be categorized into four main subsettings: *inductive*, *transductive*, *semi-supervised* and *unsupervised transfer learning*. Inductive transfer learning (ITL) requires a teacher in the target context: a few labeled examples are required in the target context in order to induce a target predictive function, but in addition labeled or unlabeled examples from the source context are available to boost the learning performance. Semi-supervised (STL) and transductive transfer learning (TTL) require a teacher only in the source context: the learner's input consists of labeled examples from the source context and unlabeled examples from the target domain. As mentioned earlier, in semi-supervised transfer learning the test set is not available during training time, in contrast with transductive transfer learning. It is natural to claim that the setting of inductive transfer learning serves as a rough upper-bound to the performance of a learner based on transductive transfer learning or semi-supervised learning. One final note for these two last settings is about the labels: it is required that either the labels in both contexts are the same or there is a learnable correspondence between the label spaces (e.g. see [KP13] for a treatment of different label spaces). Finally, unsupervised transfer learning (UTL) is the setting where no teacher exists in any context: unlabeled examples from both contexts are available to the learner and the goal is to estimate the underlying distribution (often called density estimation) or find hidden patterns in this structure, for instance by clustering or dimensionality reduction of the inputs.

As mentioned in the Definition 2.1.1, the core idea of transfer learning is that knowledge from one context can potentially improve the learning performance in a related, but different

context. One useful question is *where* we can apply this knowledge. By taking a careful look into the basic components of a learner, the answer to this question can reveal some of the frequently used strategies in the transfer learning literature.

A learner or learning algorithm receives inputs along with a feature representation, potentially with the presence of labels and has a set of assumptions, i.e. model parameters. Based on these, we can identify four major transfer learning strategies: input-based or re-weighting methods, feature-based or projection methods, parameter-based and label-based methods. Although this taxonomy is by no means complete[2], it demonstrates the current trends in the TL literature.

2.2.1 Inductive Transfer Learning

In Inductive Transfer Learning, the train set consists of some labeled target-domain instances and source-domain instances (labeled or not). We will denote the train set as X, its target part X_t which will always be labeled and its source part as X_s.

In [DYXY] they propose `trAdaBoost`, an ITL meta-algorithm based on the assumption that feature and label spaces are the same. They iteratively improve (boost) a learner by checking its performance on X_t. Both source and target train data are labeled and they are re-weighted in each iteration, however with a different goal: for X_s the goal of the re-weighting mechanism is to reduce or increase its impact in the next iterations, where impact here is translated as the data's contribution to the average empirical error (the empirical error $\hat{R}$ is measured on the train set, see 3). For X_t the goal of the re-weighting mechanism is to make the incorrectly-classified target train instances receive more attention in the next iterations. After a fixed number of iterations, the algorithm outputs the improved learner for which they provide convergence and generalization guarantees.

In [KHA09] they propose `trBagg`, also an ITL meta-algorithm which combines multiple weak-learners into an aggregated one. At first, they generate from X a fixed number of sampled-train sets via bootstrap sampling (i.e. sampling uniformly with replacement). We can assume that for each sample-train set, each train instance (either from the source or the target domain) is assigned a weight which corresponds to its number of appearances in this set. Weak learners are trained on these sample-sets, a process which can be done in parallel. After this learning phase, the method iteratively find a subset of these learners whose majority voting on the X_t has smaller empirical error (on X_t) than a fallback learner trained only on X_t. This filtered subset will be the aggregated learner which the method gives as an output. It is argued that this instance-weighting method affects the variance[3] factor of the aggregated learner, a factor which highly influences its generalization error.

In [HMT05] they are given a lot of labeled source train data X_s and a limited number

[2] For more information, we refer the interesting reader to [PY10]. In this thesis, only statistical methods will be discussed, since geometrical methods don't rely on the distribution of the input, therefore the domain difference is of no importance to them (however, an interesting approach can be seen in [DJX+09]).

[3] If a learner is sensitive to small changes in the train set, then we say that it is a high-variance learner. Therefore reducing its variance factor, we make the learner less sensitive to changes in the train set and therefore we can argue that it has better generalization capabilities.

of target train data X_t. Their goal is roughly to adapt a grammar-driven parser trained on newspaper text to a biomedical domain. Under the constraints imposed by X_t they try to estimate the conditional distribution $P_t(y|x)$ using the maximum entropy principle: from all the probability distributions that satisfy these constraints, chose the one with the maximum entropy. In this approach, often called minimum divergence modelling, there is a reference distribution $P_r(y|x)$ which is used to incorporate prior knowledge into the model. Usually this is the uniform distribution over all the possible labels an instance can take, so the model measures the divergence of $P_t(y|x)$ from the uniform distribution (which is the distribution with the maximal entropy). Their idea is to take advantage of the plentiful labeled source data to model $P_s(y|x)$ which they incorporate as a prior in their model to estimate $P_t(y|x)$. Therefore they apply transfer learning in the parameters of the model.

In [DI07] they propose EA: a method that tries to alleviate the difference between the domains so that standard supervised methods can be employed. Given labeled train data from both domains and under the assumption that the feature space is shared, they replicate each feature with domain-specific versions of it. So given there are only two domains, source and target, let $x \in \mathbb{R}^m$ be an instance in the source domain where the input space is the set of m-dimensional real numbers. Then it is transformed to the vector $\langle x, x, \mathbf{0} \rangle$ in $\mathbb{R}^{3m}$ where $\mathbf{0} \in \mathbb{R}^m$. Accordingly, if x belongs to the target domain then it is transformed to the vector $\langle x, \mathbf{0}, x \rangle$. The first m-block of features is common for all instances whereas the second and the third are activated only for source and target instances respectively. So features that are shared in both domains will be given more attention by the learner in the new augmented feature space.

In [RBL+07] they are given labeled target data X_t and many unlabeled source data X_s. Roughly they use the unlabeled X_s to learn a higher-level representation which in effect will be more succinct and will capture commonalities among the instances in the source domain. For instance, if the instances are images given in pixels, a higher-level representation might detect certain strong correlations between rows of pixels, and therefore learn that most images have many edges. Then they apply this new representation to the labeled X_t and employ a standard supervised method on this new labeled train set. It is argued that given that the domains are of the same type or modality (e.g. images, text, audio), the new feature representation learned in the source domain can boost the performance of a learner in the target domain.

2.2.2 Semi-supervised Transfer Learning

In Semi-supervised Transfer Learning, the train set consists of unlabeled target-domain instances and labeled source-domain instances. We will denote the train set as X, its target part X_t which will always be unlabeled and its source part as X_s which will always be labeled. The goal is to predict the label of a new target instance.

In [KP13] they propose TL-PLSA, a generative STL method under the assumptions the feature space is shared but the label spaces are different, although there is a shared subset of them between the domains. They assume two different kinds of higher-level representation

(which in the context of text domain is usually referred as a set of topics): one set of topics for instances and one for features. In particular, the method assumes that an instance x is generated as follows: pick an instance-topic with probability $P(z)$ and select an instance conditioned on the given instance-topic with probability $P(x|z)$. For each feature f of the instance x: select a feature-topic conditioned on the instance-topic with probability $P(k|z)$ and select the feature conditioned on the given feature-topic with probability $P(f|k)$. So overall $P(x, f) = \sum_{z,k} P(k, z) \cdot P(x|z) \cdot P(f|k)$ since it is assumed that instances and features are conditionally independent from the respective topics. All the parameters in the summation are estimated so as to maximize the predictive probability of the observed features. By letting instance-topics z correspond to the label spaces, we can observe that instance-topics are separated to source z_s and target z_t with a subset of them being shared z_{st}. For the source instance-topics, the parameters can be initialized based on the labeled source domain, therefore applying transfer learning in the model's parameters. In addition, given the number of source, target and shared classes, they can identify which instance-topics are shared, in order to transfer knowledge only where it is appropriate.

Another example of parameter-based STL is presented in [HLCN+15], where they assume that the feature and label space is common. In particular, they deploy the simplifying Naive Bayes assumption which states that for an instance $x = \{x_1, \dots, x_n\}$, its features are independently chosen, meaning that for a label y and for $i = 1, \dots, n$ it holds that $P(x|y) = \prod_i P(x_i|y)$. If we had an estimate of the label-priors $P(y)$ and the conditionals $P(x_i|y)$ for every feature and every label, then we could label a new instance $y^* = \operatorname{argmax}_y P(y|x_{new}) \propto \operatorname{argmax}_y P(y) \cdot \prod_i P(x_i|y)$. They initialize the required parameters with the labeled source data and they iteratively improve the estimation on the unlabeled target data so as to maximize the predictive probability of the observed features in the target domain.

In [HPS13] they propose SLDAB, an iterative non-probabilistic method which can be categorized as a label-based method. They iteratively build a learner which simultaneously optimizes the source classification error and preserves a low discrepancy distance between the source and the target distribution. The latter condition ensures the generalization capabilities of the learned hypothesis. In each iteration, target data are iteratively self labeled and distributions over both datasets are maintained, in order to measure their divergence. They employ as divergence a measure computed by a maximum graph matching procedure and they provide convergence and generalization guarantees for their method.

In [SS07] they provide a feature-based TTL method where they try to transform instances so that the distributions become more similar and therefore standard supervised methods can be employed under this transformation. In particular, they attempt to minimize efficiently the source classification error by identifying a subset of good features where the distributions exhibit low discrepancy distance. Since no labels are provided for X_t, they use the poor estimation from a shift-unaware learner. The result learner, since it will be trained on generalizable features, it is argued to exhibit good generalization capabilities in the target domain.

2.2.3 Unsupervised Transfer Learning

In Unsupervised Transfer Learning, the train set consists of unlabeled instances from both domains.

In [BHLS13] they try to find a new feature space where the source and target distributions, estimated by the unlabeled samples, are as similar as possible. Although they demonstrate the power of their method in the presence of source labels, it is evident that it could be used for unsupervised tasks as well. In this way, target patterns can be found using structural information from the source domain as well.

Another feature-based UTL method is proposed in [DYXY08]. Their objective is to find a good clustering for the given unlabeled target data with the help of a clustering on the unlabeled source data as well as a clustering of the shared feature space. A feature-clustering in effect groups together features that exhibit similar behaviour among instances. In effect they provide a transfer learning variant of the *co-clustering* algorithm proposed in [DMM03], which tries to minimize the loss in mutual information[4] between instances and features before and after the co-clustering. Their idea is to simultaneously perform two co-clustering operations in the source and target instances, where both operations will share the same feature clustering. They experimentally tune a trade-off parameter which balances the influence between the domains and they prove that their iterative method exhibits convergence guarantees.

2.3 Transductive Transfer Learning

In Transductive Transfer Learning, the train set consists of unlabeled target-domain instances and labeled source-domain instances. We will denote the train set as X, its target part X_t which will always be unlabeled and its source part as X_s which will always be labeled. The goal is to predict the labels of the X_t. So unlike the Semi-supervised Transfer Learning, the test set is available at training time.

In [TCWX09] they propose `ANB`, a weighted transfer version of the famous Naive Bayes Classifier (see [Lew98] for a detailed reference). Their core idea is to identify a set of domain-independent features so when a learner is trained on the source instances, it will not be biased by source-specific features. They pick the generalizable features if two criteria are met: they occur frequently in both domains and they have similar occurring probability. Furthermore, they deploy the simplifying Naive Bayes assumption that all features are conditionally independent. So they initiate a learner with the labeled restricted source data and they iteratively improve the estimation using also the unlabeled target data so as to maximize the predictive probability of the observed features in the target domain. In each iteration, they use only the generalizable features for the source instances and all the features on the target instances in order to enhance the learner's generalization capabilities.

In [BGJV09] they propose `CGC`, an iterative method that uses source information in order to

[4] Mutual Information between two random variables X, Y is a measure of how much one random variable tells us about another. It is defined as $I(X,Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \cdot \log \frac{p(x,y)}{p(x) \cdot p(y)}$. Higher mutual information values indicate more certainty that one random variable depends on another.

supervise a clustering on the target domain. Since the source data are already labeled, we can assume that an optimal clustering for them is provided. In each iteration, the criterion of homogeneity between clusters in the target domain is therefore extended to incorporate the alignment between the domains. In particular the cross-domain alignment is found by constructing a complete bipartite graph that has one set of vertices corresponding to the source centroids and another set corresponding to the target centroids. The edges of the graph are assigned a user-defined similarity measure and therefore to find the best alignment is equivalent to find the maximum weighted bipartite match in the graph. Since feature spaces might be different, they identify a shared set of features on which they project the rest of the features and incorporate this knowledge inside the cross-domain similarity measure (resulting in a parameter-based TL method).

In [DrXYY07] they propose `NBTC`, another weighted transfer version of the Naive Bayes Classifier. Label and feature spaces are considered shared and they assume all features are conditionally independent (Naive Bayes Assumption). As in previous works, they initiate a learner on the labeled source data and iteratively improve this poor estimation using also the unlabeled target data so as to maximize the predictive probability of the observed features in the target domain. Although similar methods have already been mentioned, they all had an experimentally-tuned trade-off parameter to balance the influence of the source and target data in each iteration. However, this method attempts to automate the choice of this parameter by setting it as a function of a divergence measure between the estimated probability distributions of the observed data. Therefore, if the distributions are very similar, then source and target data equally influence the learning method, otherwise the method will be biased towards one domain and the improvement will be lessened.

In [XDXY07] they propose `bridged refinement`, a TTL method which aims at refining the target labels of X_t. Label and feature spaces are assumed to be the same as well as the conditional distributions $P_s(O|I), P_t(O|I)$ among domains. Inspired by the Page Rank [PBMW99], they assume a mutual reinforcement principle among instances: if the neighbours of an instance x have high confidence of belonging to a label y, then x may receive also high score to y. This confidence can be regarded as an estimation of $P(x|y)$. The method consists of 2 iterations: the first (pre-processing phase) is applied to the whole dataset X and the second (refinement phase) to X_t. In each iteration, a confidence score is updated for each instance and for each label based on the confidence of the K (tunable parameter) closest neighbours (under some measure). The initial confidences for the first iteration are set from a domain-unaware classifier trained on X_s and applied to X_t, whereas the initial confidences for the second iteration are the output of the first iteration (projected on X_t). At the end of the second iteration, we label each instance in X_t with the label that has the highest confidence.

In [DXYY07] they propose `CoCC`, a TTL iterative method where they use co-clustering as a bridge to propagate source information in the target domain. Label and feature spaces are assumed to be the same as well as the conditional distributions $P_s(O|I), P_t(O|I)$ among domains. Their core idea is to use the labels provided in X_s in order to constrain the feature clusters which are shared among the domains. Since in the target domain, instance and

feature clusters are identified simultaneously, their idea is argued to allow the clusters of X_t to be mapped to a corresponding source label. In effect, given that I is the feature set and O the label space, they develop a co-clustering method that simultaneously minimizes a loss function in the mutual information between the instance-feature target co-clustering (X_t, I) and between the label-feature co-clustering (I, O) before and after the co-clustering. They experimentally tune the number of feature clusters and a trade-off parameter which balances the influence of (I, O) to (X_t, I) and they prove that their iterative method exhibits convergence guarantees.

3 Background

In this chapter, we introduce some notations and definitions that will be used in this thesis. In section 3.1 we formally define the problem under investigation, namely the Domain Adaptation problem and we review some known results on generalization bounds. In section 3.2 we provide the notion of context-similarity we will adopt in this work and discuss on existing approaches to compute this measure. Finally in section 3.3 we elaborate on the pre-processing step of human-created data, a crucial step in our experimental analysis provided in chapter 5.

3.1 Learning under Domain Adaptation

In this thesis, we focus on transductive transfer learning and we restrict to the case where only the source and target domains differ. This problem is often called *Domain Adaptation*. In particular, we consider two contexts, a *source* context $C_s = \{D_s, T_s\}$ and a *target* context $C_t = \{D_t, T_t\}$ where source and target input spaces are the same $I_s = I_t = I$ and tasks are considered the same $T_s = T_t$. Therefore the set of all possible outputs (namely the label space) is the same for both contexts $O_s = O_t = O$ as well as the conditional probability distributions $P_s(y|x) = P_t(y|x)$, for $x \in I$ and $y \in O$. Furthermore, we make the assumption that only the marginal distributions differ, that is: $P_s(x) \neq P_t(x)$. This situation is often termed *covariate shift*.

Although the assumptions may seem restrictive, they are quite natural in many real-world problems such as document classification. As a motivation for this setting, let us imagine we have been given documents in English coming from two contexts: movie-reviews and book-reviews. Movie-reviews have already been annotated as positive-negative but book-reviews are still left to be determined. In both cases the task is to automatically recognize if the review is positive or negative, so labels are the same. Since all documents are in the same language, then input spaces are also the same. Finally, we can conclude that if two documents are very similar (for instance they contain synonym words), it would be natural to categorize them in the same class, therefore conditional probability distributions should also be the same. A toy example is shown in Figure 2: squares are coloured as purple, and a triangle seams more similar to a square then a circle, so we might chose to color the triangles as purple as well.

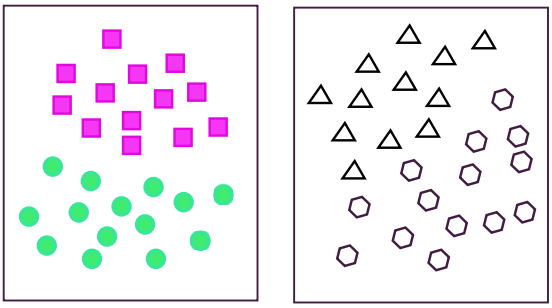


Figure 3.1.1: What color would you paint triangles and polygons?

In this setting, let us consider two sets $X_s = \{x_l^s\}_{l=1}^{N_s}$ and $X_t = \{x_u^t\}_{u=1}^{N_t}$ composed of N_s source-domain and N_t target-domain instances, sampled from the unknown marginal distributions $P_s(x)$ and $P_t(x)$ respectively. Let us further assume that $x^s, x^t \in \mathbb{R}^m$, that is each instance is a m-dimensional real vector, where m represents the dimensionality of the common input space $I = \mathbb{R}^m$. The same set of K classes $O = \{o_k\}_{k=1}^K$ characterizes both domains D_s and D_t. A set of true labels $Y_s = \{y_l^s\}_{l=1}^{N_s}$ for X_s is available, where $y^s \in O$, thus it is possible to define a source labeled set $Tr_s = \{X_s, Y_s\} = \{(x_l^s, y_l^s)\}_{l=1}^{N_s}$ for D_s drawn from the probability distribution $P_s(x, y) = P_s(x) \cdot P_s(y|x)$. Given Tr_s and X_t, our goal is to find a set of labels $Y_t = \{y_u^t\}_{u=1}^{N_t}$ for X_t such that $\{X_t, Y_t\}$ best approximates the unknown distribution $P_t(x, y) = P_t(x) \cdot P_t(y|x) = P_t(x) \cdot P_s(y|x)$.

In order to provide a formal definition of the problem, let us review some terminology from Learning Theory in [Vap98]: let G be a generator of random vectors $x \in I$ chosen independently from a fixed but unknown distribution $P(x)$, where I is an input space. Let T be a teacher which returns for every input x an answer $y \in O$ according to some fixed but unknown function $f : I \to O$, where O is the set of all possible information classes. Therefore we consider the existence of a target function f and a labeled train set $Tr = \{(x_i, y_i)\}_{i=1}^n$.

So a learning machine M, given the train set Tr will produce a so-called hypothesis function $h : I \to O$ which will best approximate the target function f. We have already mentioned in 2.1 that a learning machine needs to be associated with a set of parameters A that will in effect characterize its bias or its *hypothesis space* H_A. Therefore let M_A be a learning machine capable of computing a set of functions $H_A = \{h_a(x) | h : I \to O \text{ and } a \in A\}$ where A a set of parameters. Then the learning problem is: *Given generator G, teacher T, a train set Tr and a learner M_A, the goal of M_A is to choose from H_A a function (hypothesis) that bests approximates the answers of T.* An optimal prediction depends on how much an error can cost. This concept can be quantified with the use of a loss function $L(\hat{y}, y) : O \times O \to \mathbb{R}$ which measures how much the prediction $\hat{y} = h_a(x)$ of the hypothesis function differs from the real answer $y = f(x)$. Therefore we can define the *risk* or *expected error* of choosing a hypothesis to be $R(a) = \underset{(x,y) \leftarrow P(x,y)}{E} [L(h_a(x), y)]$ (we will often omit the subscript a unless otherwise stated). Therefore the learning problem can be formulated as follows:

Definition 3.1.1. *[Learning problem] Given a generator G, a teacher T, a train set Tr, a learner M_A and a loss function L, the goal of M_A is to find a function $M_A(Tr) =$*

$h_{a^*}(x) \in H_A$ *such that* $a^* = \underset{a \in A}{\operatorname{argmin}} R(a)$.

Unfortunately, since $P(x, y)$ is unknown, the expected error cannot be computed. However, we can define the *empirical error* to be $\hat{R}(a) = \frac{1}{n} \sum_{i=1}^{n} L(h_a(x_i), y_i)$ which is the average cost of the hypothesis h_a on the train set. We say that a hypothesis is *consistent* with Tr if the empirical error is zero, meaning that the chosen hypothesis made no mistakes on the train set. Although we can measure this quantity, how close will it be to the expected error? Before showing this, let us mention a useful notion related with the hypothesis space. Given an input space X and a hypothesis space H defined over X, let us call *dichotomy* a partition of the sample set X into two disjoint subsets. We say that X is *shattered* by H if for every dichotomy of X there exists a hypothesis $h \in H$ which is consistent with this dichotomy. Therefore, the size of the largest finite subset of X shattered by H is called *the Vapnik-Chervonenkis dimension* of H and is denoted as $VC(H)$. If arbitrarily large finite subsets of X can be shattered by H then $VC(H) = \infty$. Intuitively, VC dimension is a quantitative way to measure the capacity (or complexity or richness) of a learner. It has been shown in [Vap98]:

Theorem 3.1.1. *Let $S \subseteq X$ be a train set of size N over an input space X identically and independently drawn from a distribution $P(x, y)$, let M_A be a learner characterized by a hypothesis space H_A such that $VC(H_A) = v$ and let $R(a)$, $\hat{R}(a)$ the expected and empirical errors respectively. Then for any $0 \leq \delta \leq 1$ it holds that with probability $1 - \delta$*

$$R(a) \leq \hat{R}(a) + \sqrt{\frac{1}{N}[v(\log \frac{2N}{v} + 1) - \log \frac{\delta}{4}]}$$

The result above holds for finite $v \ll N$ and for binary functions $h \in H_A$. There are many proposed theories to extend this result that are out of scope for this thesis but the useful point here is that under some conditions, we can bound the generalization error with the empirical error and the capacity of the learner.

Going back to our setting, the train set consists of a labeled part Tr_s and an unlabeled part X_t coming from different distributions and our goal is to predict on the unlabeled part on the input. Therefore we can adapt Definition 3.1.1 as follows:

Definition 3.1.2. *[Domain Adaptation problem] Given a generator G, a teacher T, a source labeled train set Tr_s, a target unlabeled train set X_t, a learner M_A and a loss function L, the goal of M_A is to find a function $M_A(Tr_s, X_t) = h_{a^*}(x) \in H_A$ such that $a^* = \underset{a \in A}{\operatorname{argmin}} R_t(a) = \underset{a \in A}{\operatorname{argmin}} \underset{(x,y) \leftarrow P_t(x,y)}{E} [L(h_a(x), y)]$.*

In the Domain Adaptation scenario, there is no known similar bound with 3.1.1 (to the best of our knowledge). However, this is an active area of research in Transfer Learning that has already produced some partial results. For instance, in [BCK+08] they show generalization bounds in the case of Unsupervised and Inductive Transfer Learning, where in the latter they specialize on learners that try to minimize a convex combination of empirical source and target

errors. In both cases, apart from the VC-dimension, another measure that is used in the bounds is the similarity between the contexts and in particular of the probability distributions governing them. For their case, they define similarity based on a hypothesis space-specific distance measure.

To stress the significance of the notion of context-similarity, we can observe that the core idea of all the discussed methods in chapter 2 is based on how they define (implicitly or explicitly) and exploit context-similarity. Therefore, it is of no surprise that this notion will also play a crucial role in our method. For reasons that will be elaborated in section 4.3 and based on our assumption that only the marginal distributions differ, we will base the notion of similarity on the ratio $w(x) = P_t(x)/P_s(x)$. Since the marginal probability densities are unknown, we cannot compute exactly $w(x)$. So the question now is how to accurately estimate it. In the following section we review some existing approaches on the Density[1] Ratio Estimation.

3.2 Density Ratio Estimation

It has been argued that directly estimating the ratio is much more effective (both in time and accuracy) than estimating the densities separately and then computing the ratio (see [HMSW04] where they discuss on the hardness of density estimation for high-dimensional data). In literature there exist several methods that allow us to directly estimate this ratio. Despite not having theoretical approximation guarantees, they are experimentally tested and mostly focus on scalability (both in the sample size and the dimensionality of the input space). In principle, they provide a specific model for the ratio and they determine its parameters so that a specific function is minimized, resulting in a convex optimization problem. Below we briefly review some of these methods but we refer the interesting reader to [SSK12] for more details. In Table 3.1 a succinct summary of these characteristics is presented.

In KLIEP [SNK+08] they model the ratio with a linear model $\hat{w}(x) = \sum_{l=1}^{b} a_l \cdot \phi_l(x) = \langle a \cdot \phi(x) \rangle$ and they estimate its parameters a_l by maximizing the log-loss function $J = \int \log \hat{w}(x) p_t(x) dx$ which in effect minimizes the *Kullback-Leibler* divergence between $P_t(x)$ and $\hat{P}_t(x) = \hat{w}(x) \cdot P_s(x)$. Kullback-Leibler divergence, also known as relative entropy, is defined as defined as:

$$KL(p, q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

where p, q are probability densities. The method also deals with the automated selection of the basis functions ϕ_l in order to maximize J and it is suggested to use spherical Gaussian kernels as the search space of them. The Gaussian centers are set at the test samples in order to allocate many kernels at high test input density regions where it is expected that the density ratio tends to take large values. However, this method needs to solve a convex optimization problem which is non-linear, making it computationally rather expensive and with scalability issues.

To solve the non-linearity, in LSIF [KHS09], they study KLIEP by minimizing the squared-

[1] In this thesis, probability distribution and probability density function are used interchangeably.

loss function $J = \frac{1}{2}\int(\hat{w}(x) - w(x))^2 p_t(x)dx$. The resulting optimization problem is quadratic and a convergence analysis is provided. Again they suggest as promising model candidates the spherical Gaussian kernels for ϕ_l. Due to accumulation of numerical errors, they further propose an approximation version of LSIF called unconstrained LSIF (uLSIF) which is experimentally argued to perform better computationally. They also provide convergence as well as approximation bounds for uLSIF, the latter being dependable on tunable parameters. Again here there is no focus on scalability issues.

To deal with scalability, in LL-KLIEP [TKH+09] they argue that KLIEP idea can be naturally applied to log-linear models, so they use the normalized model $\hat{w}(x) = \frac{exp(\langle a \cdot \phi(x) \rangle)}{\frac{1}{N_s}\sum_{x' \in X_s} exp(\langle a \cdot \phi(x') \rangle)}$. Their motivation comes from the observation that in KLIEP, in each step of the gradient ascent (an algorithm that solves the optimization problem) the summation over all test samples needs to be computed, which is prohibitively slow in large-scale problems. They make KLIEP feasible in the setting of large-scale test dataset, yet dimensionality issues was not the focus. The main advantage is computational efficiency since the computation time of LL-KLIEP is nearly independent of the amount of test data (under some pre-processing assumptions). Also they experimentally confirmed that the accuracy of the proposed method is good enough.

In GM-KLIEP [YS09] they argue that KLIEP idea can be naturally applied to Gaussian mixture models. Their motivation comes from the following observation: in a typical implementation of KLIEP a spherical Gaussian kernel model is employed and the Gaussian width is shared by all the basis functions ϕ_l. This means that when the true importance function is correlated, the performance of KLIEP could be degraded. To cope with this problem, they propose to use a Gaussian mixture model $\hat{w}(x) = \sum_{l=1}^{b} \pi_l \mathcal{N}(x|m_l, \Sigma_l)$ and learn the covariance matrix of the Gaussian components at the same time (through an iterative estimation procedure). So, they learn the importance function more adaptively even when the true importance function contains high correlation. Since the optimization function is the same as in KLIEP, this method inherits scalability issues and may suffer from numerical instability in certain data samples.

In order to deal with high-dimensional input spaces, in D^3-HSA [YS11] they employ the idea of performing density ratio estimation only in a subspace where the two distributions are significantly different and in particular where they are are maximally different under the *Pearson* divergence. Pearson divergence is a squared-loss variant of the Kullback-Leibler divergence and it is defined as

$$PE(p,q) = \frac{1}{2}\int q(x)\left(\frac{p(x)}{q(x)} - 1\right)^2 dx$$

where p, q are probability densities. They refer to this subspace as *hetero-distributional subspace* (HDS). Intuitively, in such a subspace the train and test samples will be maximally separated. After finding such an HDS, they perform the aforementioned uLSIF in this low-dimensional subspace.

Density ratios have two natural limitations: they can be unbounded even for simple cases (see [CMM10]) and they are asymmetric and thus the "direction" needs to be determined by a user. In our case, the direction will be set in section 4.3 so the only thing to tackle is that they can be unbounded. For that reason, we chose to use the RuLSIF method [YSK+11].

Table 3.1: Characteristics of Density Ratio Estimation methods

Name	Loss Function	Model	Scalability
KLIEP	log loss $J = \int \log \hat{w}(x) p_t(x) dx$	linear Gaussian kernels	not efficient for high dimensional spaces not efficient if many test data
LSIF	squared loss $J_0(\alpha) = \frac{1}{2} \int (\hat{w}(x) - w(x))^2 p_t(x) dx$	linear Gaussian kernels	approx. uLSIF due to numerical issues
LL-KLIEP	log loss $J = \int \log \hat{w}(x) p_t(x) dx$	log linear	scalable on sample size
GM-KLIEP	log loss $J = \int \log \hat{w}(x) p_t(x) dx$	Gaussian mixture	has KLIEP's scalability issues numerical issues
D^3-HSA	squared loss $J_0(\alpha) = \frac{1}{2} \int (\hat{w}(x) - w(x))^2 p_t(x) dx$	linear Epanechnikov kernels	efficient scalable on dimensionality
RuLSIF	squared loss $J(\theta) = \frac{1}{2} E_{q_\alpha(x)}[(g(x;\theta) - w_\alpha(x))^2]$	linear Gaussian kernels	smoothed extension of uLSIF

In this method, they introduce the notion of the α-*relative density ratio* as

$$w_a(x) = \frac{p(x)}{\alpha \cdot p(x) + (1-a) \cdot q(x)} = \frac{p(x)}{q_a(x)}$$

where it is easy to observe that for $\alpha = 0$ it reduces to the usual definition of the ratio. They model this relative ratio with a linear model $\hat{w}(x) = \sum_{l=1}^{b} \theta_l \cdot \phi_l(x) = \langle \theta \cdot \phi(x) \rangle$ where $b = n_s$ the number of source samples and they estimate its parameters θ_l by maximizing the function $J = \frac{1}{2} \int q_\alpha(x) \left(r_a(x) - 1\right)^2 dx$ which in effect minimizes the relative analogue of the Pearson divergence. Pearson divergence is argued to be more efficient than the Kullback-Leibler, due to the absence of the non-linear log function. This method is regarded as an extension of uLSIF to the α-relative density ratio and they also provide extensive convergence analysis. As before, it is suggested to use the family of spherical Gaussian kernels as the basis functions.

Since none of the above methods is equipped with theoretical approximation guarantees, we performed experiments in chapter 5 and focused our attention to a comparison between uLSIF and RuLSIF. Despite being simple and readily available in [Lab], our two candidates were chosen based on their unique properties: uLSIF is arguably the fastest proposed method whereas RuLSIF handles unbounded ratios smoothly and in a tunable way.

Experiments usually supplement the theoretical study, allow the investigation of tunable parameters but they also indicate practical limitations and technicalities of the proposed methods. As it is typical in the machine learning literature we conducted experiments on synthetic as well as on real data. Synthetic data are crafted input instances generated from known distributions and are usually employed to justify the predictive power of a learner on good instances as well as its limitations in bad instances. In this thesis, we created 3 synthetic experiments that allowed us to investigate further the behaviour of our method as well as the selection of the most suitable density ratio estimator. The generation and discussion on them can be found in chapter 5.

On the other hand, real data are input instances generated by unknown distributions. They

are employed as an application of the proposed method in a realistic scenario in order to investigate accuracy and scalability issues, since real data typically come in large amounts from a high-dimensional space. Despite being arbitrary input instances, their modality is usually known, e.g. the data are documents, images or audio. In this thesis we chose to experiment with documents. Such human-created instances, usually require a pre-processing step that transforms them in a computer-friendly representation.

3.3 Document Pre-processing

Earlier we mentioned that the input consists of data which are m-dimensional feature vectors. However, this is just a theoretical formulation and it fails to describe the nature of all available data, especially the ones generated by humans. In order to conduct experiments, a transformation needs to take place so that real data can be represented by explicit feature vectors and in effect the learning systems can be operational. Such a pre-processing phase can be considered as data-compression: we try to transform data from a highly-expressive, complex and unstructured representation, such as natural language in the case of documents, to a more restrictive and structured one, that is easier for a computer to manipulate. The goal in this phase is to minimize the loss of information by preserving the semantic richness of the data, yet efficiently resulting in a more manageable form.

This phase is actually one of the main active research topics in the Information Retrieval field. For documents, many models have been proposed such as the Boolean model (set-theoretic approach), the Vector Space model (algebraic approach) and the Probabilistic model (probabilistic approach). We refer the interesting reader to [RM11] (chapter 6) for a detailed overview and focus our attention to the Vector Space model. Although it is quite simple and inevitably results in loss of information, yet we select it because of its popularity and because it is equipped with a natural mechanism for distance and similarity. Since our proposed method in chapter 4 involves a clustering phase, this mechanism will prove to be very useful when we will experimentally validate the method in chapter 5.

The Vector Space model [SWY75] is based on the idea that the meaning of a document is conveyed by the words used in it. This model does not take into account the order of the words, it simply considers documents as bags of words. In particular, a document $\mathbf{x}$ is represented as a feature vector $\mathbf{x} = \{x_1, \dots, x_m\}$ where each feature x_i is a weight representing the *importance* of the word i in the document. Therefore m denotes the cardinality of a set V containing all the possible words and often referred to as *vocabulary*. Typically, the weight w_i is the number of appearances of word i in the document x, denoted as $w_i = n_{x,i}$. If the word i does not appear in the document x, then $w_i = 0$, so we guarantee that all documents have the same dimensionality m.

Since V was defined as a set containing all the possible words, its cardinality will not provide the manageability we aim for. Therefore we can initially restrict the vocabulary to contain only words that appear in the documents provided as input. In the case of Domain Adaptation, this includes the documents of the source and the target domain. Since in this setting, our goal is to predict the labels on the given target documents, with this simplification no document

will be left without a valid representation. In addition, the universality of vocabulary V realizes our assumption that the feature space is shared among domains. In addition, very frequent words such as *the*, *is*, *at*, *which*, and *on* (often referred to as *stopwords*) usually provide no useful information to discriminate between documents, so they could also be removed. Finally some other common heuristics that seem to work well in practice for document classification and reduce the dimensionality of the feature space (i.e. the size of the vocabulary) are : remove rare words, for example words that appear less then 3 times in the whole corpus of documents and instead of using the words, use their respective word stems[2].

Previously, we defined as word-importance the number of the word's appearances in a document. However, this definition may seem a bit inadequate. If a word appears frequent in one document but is absent in another document generated from the same domain, how well can we argue that this feature characterizes the domain? Following this line of thought, some other ways have been proposed to better capture the *importance* of a word by taking into account not only the document under examination but the whole corpus of documents. One of the most popular candidates is the *tf-idf* weighting. Tf-idf stands for term frequency-inverse document frequency and it is a two-fold statistical measure: if a word appears frequently in a document (term frequency tf), it must be important but if it appears in many documents (document frequency df), then it must not be a unique identifier of the domain. So for every word $i \in V$:

$$w_{x,i} = tf_i \cdot idf_i = \frac{n_{x,i}}{n_x} \cdot \log\left(\frac{|D|}{n_i}\right)$$

where n_x is the total number of words in document x, n_i is the number of documents where word i appears and $|D|$ is the size of the corpus, i.e. the total number of documents. The logarithm used here is merely to smooth out the presence of rare words, since if a word appears in very few documents, then $|D|/n_i$ would be boosted too much.

In Domain Adaptation, we inevitably reach the following dilemma: shall we compute the document frequency on both target and source domains, or for each domain separately? In other words, what happens if a word appears often in many source documents but not so often in target documents?

In this case, we can either use a local tf-idf weighting for each domain respectively or we can use a global tf-idf weighting where both domains are combined. We argue that a local tf-idf weighting is more suitable for two reasons. First, the pre-processing of the domains can be done independently reducing the computational cost and allowing for adaptability on various source and target domains. Secondly, a local tf-idf weighting encapsulates the domain-difference in the feature-representation of the data. To illustrate this better, let us consider the extreme case of a one-dimensional feature space. In the case of global tf-idf, this single feature/word has the same document frequency on every document therefore its importance is solely determined by its term frequency. Let us assume the word appears the same number of times in a target and a source document. Then in the global case, these

[2] A stem is a portion of a word that is left after the removal of its affixes (prefixes and suffixes), e.g. *hetero* is the word stem from which we can derive words like heterogeneous, heterogamy, heterodox. Many stemming approaches exist like brute-force look up, stripping the affixes, using word n-grams etc. In this thesis, we chose to use the popular Porter stemmer [Por80] that utilizes suffix stripping.

documents will look identical. However, in the case of local tf-idf, the feature might appear less important in the source document but more important in the target document if it is observed often in the source but not so often in the target domain. Therefore the documents will not look identical, effectively capturing the domain-difference. This argument is further validated experimentally in in chapter 5.

4 Proposed Approach

In this chapter, we present the proposed method of this work. In Section 1. we introduce the clustering problem and review the current literature. In Section 2. we motivate and present a Domain Adaptation method and in Section 3. we offer an analysis of this method under certain assumptions.

4.1 Introduction

"Clustering is the grouping of similar objects" [Har75, p.1], therefore given a set of objects, the problem is to find subsets or clusters which are homogeneous and/or well separated. There exists a vast literature around this problem and its applications can be found in almost any scientific discipline, such as psychology, data mining, bioinformatics and computer graphics. Depending on the application, research has studied different types of clusterings: hierarchical (where nested sub-clusters inside a cluster are allowed), partitional (where no sub-clusters are allowed), hard (where each object belongs only in one cluster), soft (where each object can belong to one or more clusters), complete (where all objects are assigned to some cluster), partial (where some objects might not be assigned to some cluster) etc. In this work, we focus on complete hard partitional clusterings.

In addition, depending on the nature of objects, there exists different types of clusters: prototype-based (where a cluster is defined as a set of objects closer to the prototype or center that defines the cluster than to the center of any other cluster), density-based (where a cluster is defined as a dense region of objects surrounded by a region of low density), graph-based (where objects are represented as a graph and a cluster is defined as a connected component of the graph, i.e. a group of objects that are connected with each other and have no other connection outside the group) etc. In this work, we focus on prototype-based clusters, where every cluster can be succinctly represented by a *center*, such that objects belonging to this cluster will be more similar to its center than to any other center. The notion of center effectively captures a common application of clustering: to approximate a large set of objects by a small set of representatives. Yet, how shall we define similarity?

In literature, there exist many notions of similarity and a standard way of expressing it is through a set of distances between pairs of objects. The selection of a suitable distance function

is highly correlated with the nature of the objects, since "a serious difficulty in choosing a distance lies in the fact that a clustering structure is more primitive than a distance function and that knowledge of clusters changes the choice of distance function" [Har75, p.58]. For two m-dimensional vectors a, b, some examples of similarity measures are the cosine similarity $cos(a,b) = (\sum_{i=1}^{m} a_i * b_i) * (|a||b|)^{-1}$, the manhattan distance $m(a,b) = \sum_{i=1}^{m} |a_i - b_i|$ and the chebychev distance $c(a,b) = \max_{i \in [m]} |a_i - b_i|$ but perhaps the most popular one, is the *euclidean distance* $d(a,b) = (\sum_{i=1}^{m} (a_i - b_i)^2)^{1/2}$ which some argue that mostly corresponds to our everyday experience and perceptions.

Given a set of centers, let's assume we assign each object to its closest center with respect to the euclidean distance. Therefore a set of clusters is created and there are different ways to evaluate the cost of such a clustering. The choice of the cost allows us to create useful clustering problems, such as the Kmeans, the Kmedians and the K center problems. For the Kmeans problem, which will be the focus of this work, the cost is defined as the sum of the squared Euclidean distances from every object to its nearest center. Formally

Definition 4.1.1. *[Kmeans problem] Given a set of n points $X = \{x_1, \dots, x_n\}$ in $\mathbb{R}^m$ find a set of $K > 1$ points $B = \{b_1, \dots, b_K\} \subset \mathbb{R}^m$ such that*

$$\sum_{x \in X} d^2(x, B)$$

is minimized. The minimum value, also called potential function, is denoted as $\phi_{OPT}(X, K)$ and $d^2(x, B)$ is the squared Euclidean distance from x to the nearest point in B i.e. $d^2(x, B) = \min_{1 \le k \le K} d^2(x, b_k)$.

To provide some intuition for this problem, let us assume that we have some data generated by an equally weighted combination of Gaussian distributions, all with unit variance. The real parameters of the Gaussian distributions (namely the mean values) are unknown. By minimizing the sum of the squared euclidean distances of each point to its closest center is like estimating the parameters that most likely generated the given data, namely the mean values (for more detail, see [BBM02]). Furthermore it was shown in [DH04] that such a cost function tries to minimize the intra-cluster distance while maximizing the inter-cluster distance, two desired properties of any intuitively good clustering. Despite its simple definition, this clustering problem is a rather hard one. In fact, when the dimension m is not fixed, then the Kmeans problem has been shown to be NP-hard even for $K = 2$ (many reductions exist, e.g. [Das08]). In addition, if $m = 2$ and K is part of the input, then the so-called *planar-Kmeans* has also been shown to be NP-hard [MNV09]. When both K and the dimension m of the input are fixed, the problem can be solved exactly in polynomial time [IKI94]. If only K is fixed, then the problem admits polynomial-time approximation schemes (PTAS), e.g. see [SSK04].

The Kmedians problem is similar to the Kmeans with the only difference that the cost function is the sum of the distances (not squared) from every object to its nearest center. Kariv and Hakimi in [KH79b] showed that the general Kmedians problem is NP-complete even in the plane $m = 2$. Furthermore, if $m = 2$ and the euclidean distance is used, then

it was shown in [MS84] that Kmedians remains NP-complete. To make things worse, in [JMS02] it was shown that it is even hard to approximate Kmedians within a factor $1+2/e$. However if we restrict to the Euclidean space, Kmedians was shown to admit a PTAS (first in [ARR98] and later improved by [KR99] and [HPM04]). The best result so far is a $(3+\epsilon)$ approximation algorithm by [AGK+01].

The Kcenter problem requires that the maximum distance of any object to its nearest center is minimized, i.e. $\phi_{OPT}(X,K) = \min \max_{x \in X} d(x,B)$. Intuitively, this problem attempts to minimize the maximum radius of any cluster and therefore is highly sensitive to outliers. Even under the metric restriction of a metric space [1], when K and m are part of the input, Kcenter is NP-hard [KH79a], as well as its discrete variant (where centers are allowed to be only objects of X). It remains NP-hard even if $m = 2$ but K still part of the input [MS84] (also the discrete version). The news don't get any better if we want to find good approximation solutions. In [FG88] it was shown that it is NP-hard to approximate within a factor < 2 even under the L_∞ metric[2]. It is of no surprise that if the space is not metric, then it is NP-hard to approximate within any factor (see [Hoc97, p.378] if the triangle inequality is left out and [CGH+04] if the symmetry property is left out). Despite these pessimistic results, there exist 2-approximation algorithms (e.g. see [HS86], [Gon85] and [FG88]).

4.2 Proposed Approach

As stated in 2.1.1 the goal of Transfer Learning is to improve the efficiency and accuracy of learning in a target context C_t using knowledge obtained from a source context C_s. In our setting, the goal is to discover a good clustering of the target dataset X_t given an optimal clustering in a different but similar dataset X_s. The underlying cluster problem will follow the definition of the Kmeans problem 4.1.1. The proposed method of this work is a transfer learning variant of the widely-known Lloyd's method, therefore we motivate our result by first reviewing this algorithm.

4.2.1 Lloyd's method

As stated in [ORSS13], there is a wide unsatisfactory gap between the practical and the theoretical clustering literatures. Popular heuristics such as the *Lloyd's method*, although lacking in theoretical guarantees, they are still widely used because they outperform (in terms of time complexity) the implementations of theoretically-guaranteed alternatives. In particular, Lloyd's method is a simple algorithm that attempts to locally improve an arbitrary Kmeans clustering. Initially, K centers $B = \{b_1, \ldots, b_K\}$ are chosen independently and uniformly at random from the input dataset X. Therefore, an initial clustering $C = \{S_1, \ldots, S_K\}$ is formed, where each cluster S_i is assigned the set of points in X that are closer to the center b_i than to any other center. After the assignment, the centers are updated to be the center of mass of all the points in the respective cluster. The whole process iterates until the

[1] A metric space $(\mathcal{X}, \rho)$ consists of a set $\mathcal{X}$ and a distance function $\rho : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that satisfies the three properties of a metric: reflexivity $\rho(x,y) \geq 0$ with equality iff $x = y$, symmetry $\rho(x,y) = \rho(y,x)$ and triangle inequality $\rho(x,y) \leq \rho(x,z) + \rho(z,y)$.

[2] Given two vectors $a, b \in \mathbb{R}^m$, their infinity norm is defined as $L_\infty(a,b) = \max_i \in [m]|a_i - b_i|$.

clustering C no longer changes. The pseudocode of Lloyd's method is provided in Figure 4.2.1.

Algorithm: `Lloyd's method`
Input:
X: data set in $\mathbb{R}^m$
K: the number of clusters
Output:
C: a local minimum clustering
Method:

1. Initialization $B = \{b_i\}_{i=1}^{K}$ where $b_i \overset{r}{\longleftarrow} X$
2. Assignment $C = \{S_i : \forall i \in K\}$ where
$S_i = \{x \in X : b_i = \underset{b \in B}{\operatorname{argmin}}\ d(x, b)\}$ for $i = 1, \dots, k$
3. Update $b_i = |S_i|^{-1} \sum_{x \in S_i} x$ for $i = 1, \dots, k$
4. Convergence Repeat 2-3 until C no longer changes
5. Output C

Figure 4.2.1: Pseudocode of Lloyd's method

By taking a closer look, we can observe that during the Assignment (step 2.) the algorithm makes a locally optimal decision (that is, each point is assigned to its nearest center). Each point moves to another cluster if and only if its distance from the new center is smaller then the distance from its current center. Therefore we can argue that this greedy criterion guarantees to decrease the potential function ϕ. In fact, if ϕ does not decrease then the current clustering C did not change, therefore the algorithm terminates. We can also provide a similar argument for the Update phase (step 3.), yet we first need to state the following useful property (a review of its proof can be found in Lemma B.0.1):

Lemma 4.2.1. *[centroid property] Let $S \subseteq \mathbb{R}^m$ be a set of points with center of mass $s^* = |S|^{-1} \sum_{s \in S} s$, $d(a, b)$ the euclidean distance of $a, b \in \mathbb{R}^m$ and let $x \in \mathbb{R}^m$ be an arbitrary point. Then, it holds that*

$$\sum_{s \in S} d^2(s, x) = \sum_{s \in S} d^2(s, s^*) + |S| \cdot d^2(s^*, x)$$

For any set S, lemma 4.2.1 states that if we select as representative its center of mass s^* then that would be the best choice in order to minimize its Kmeans potential function $\phi(S)$. If any other point x of the input space is selected as representative, then $\phi(S)$ will increase by a quantifiable factor which depends both on the distance of x from the best choice and the size of the set. Going back to the Update phase of Lloyd's method, we observe that for every current cluster S_i the algorithm updates its center with the best possible representative with respect to ϕ, guaranteeing again that ϕ will decrease.

Since the potential function decreases in every iteration, the algorithm will converge to a local optimum. However, in the Initialization phase (step 1.) the centers are chosen uniformly at

random from X, therefore this local optimum can be quite poor and in fact, to the best of author's knowledge, there are no approximation guarantees of how poor can it be. In practice, multiple repetitions of the algorithm help to smooth out this effect (at the expense of computational time). Also, there exist better initialization schemas that allow to estimate how poor the resulted local optimum can be, as for example the famous work by Arthur and Vassilvitskii [AV07]. Their variation of Lloyd's method, called Kmeans++ employs a different Initialization phase and guarantees that, even from the first iteration, it will result in a local optimum at most $O(\log k)$ times worse than the optimal solution.

4.2.2 tr-Kmeans

As mentioned earlier, the method provided in this thesis is a variation of the Initialization phase of Lloyd's method. In the setting of Domain Adaptation, we integrate in this phase the knowledge of some auxiliary data. To introduce some helpful notation, let us assume the existence of a source and a target context $C_s = \{D_s, T_s\}, C_t = \{D_t, T_t\}$ governed by the unknown but fixed joint distributions $P_s(x, y), P_t(x, y)$ respectively. Let us consider two sample sets $X_s = \{x_l^s\}_{l=1}^{N_s}$ and $X_t = \{x_u^t\}_{u=1}^{N_t}$ composed of N_s source-domain and N_t target-domain instances, sampled from the unknown marginal distributions $P_s(x)$ and $P_t(x)$ respectively. It holds that $x^s, x^t \in \mathbb{R}^m$, that is each instance is a m-dimensional real vector, where m represents the dimensionality of the common input space $I = \mathbb{R}^m$. The same set of K classes $O = \{o_k\}_{k=1}^{K}$ characterizes both domains D_s and D_t. A set of true labels $Y_s = \{y_l^s\}_{l=1}^{N_s}$ for X_s is available, where $y^s \in O$. Thus the input consists of a labeled source set $Tr_s = \{X_s, Y_s\} = \{(x_l^s, y_l^s)\}_{l=1}^{N_s}$ and an unlabeled target set X_t with total size $O(m \cdot (N_s + N_t))$. Considering that labeling is a costly process, we may assume that $N_t \gg N_s$ therefore the size of the input is $O(m \cdot N_t)$.

Based on the definition of Domain Adaptation in 3.1.2, one natural limitation that can appear in the form of two extreme cases is: a) if $P_s(x, y) \equiv P_t(x, y)$ then adaptation is not necessary and b) if $P_s(x, y)$ and $P_t(x, y)$ are uncorrelated then adaptation is useless and can be misleading, a situation often referred to as *negative transfer*. Therefore, it is of great importance to identify when the transfer of knowledge can be useful or not. In particular, if a) is the case, then standard classification methods under the transductive learning framework can be employed whereas if b) is the case, then T_s should be ignored resulting in a typical unsupervised learning problem where standard clustering methods can be used.

This observation motivated our study and as a result, we have developed a transfer learning algorithm that intertwines these two learning paradigms to achieve higher efficiency and accuracy than its building components. Our basic idea is to discover target instances similar to the source domain in order to transfer knowledge and guide a centroid-based clustering algorithm. We call such good target instances as friends F, i.e. a subset of the target dataset X_t that is believed to contain the most similar target instances with the source dataset X_s. Once we have discovered friends, a source-trained classifier is naively applied to the set F in order to reveal a small portion of the desired clustering in the target domain. In essence, the classifier provides an initial *seeded* clustering of F (a labeling that partitions F into K subsets). The

centers are then initialized at the mass centres of the seeded clustering. In detail, the proposed Initialization phase consists of the following two steps:

1. *Friends identification*: this step partitions the target set X_t into two subsets: friends F and non-friends $X_t - F$ with respect to the source domain. *Friends* are target instances that appear to be similar to D_s (under some specific notion of similarity which we will soon define) whereas *non-friends* are target instances that are not friends of D_s. Intuitively, friends are target instances where transfer of knowledge is most likely to be valuable.

2. *Centers generation*: in this step, we pseudo-label friends F by applying a classifier trained on the source domain. Since friends are the target instances most similar to the source domain, we expect the naive classifier will give us results with high confidence. This labeling in effect partitions friends in K subsets. The centroids of Kmeans are then initialized at the mass centres of these subsets.

After the initialization phase, our method proceeds just like the Lloyd's method, i.e. by assigning all target instances (friends and non-friends) to their closest centroid and adjusting the centroids to be the center of mass of each resulted cluster, with the goal to minimize the sum of the squared distances from every target instance to its closest center. The only thing left to conclude our method description is the notion of similarity between source and target instances. For reasons that will become clear during the analysis of the method in section 4.3 we chose to use the ratio between the source and target distributions as similarity measure. That is

Definition 4.2.1. *[Friends set] Given $0 \leq e_F < 1$, we define friends as a set F such that*

$$x \in F \iff x \in X_t \text{ and } 1 - \epsilon_F \leq \frac{P_t(x)}{P_s(x)} \leq 1 + \epsilon_F$$

It is straightforward to observe, that the closer the ratio gets to one, the more similar the distributions. Furthermore, the user-defined parameter ϵ_F allows to control the bias induced by the source-trained learner. The more tolerance we allow in the ratio of the distributions, the worse will be our friends selection. We call the proposed method tr-Kmeans and the pseudocode is available in figure 4.2.2.

Such an initialization phase achieves two important things: the first is to alleviate randomness from the algorithm itself and place it to the external environment (i.e. randomness becomes part of the input) and the second is to establish a correspondence between source and target label spaces. The seeded clustering of F is in accordance with the source domain, therefore it can be used as prior information to find a good global clustering in the target domain. In effect, our initialization step is a transfer learning variation of the *Seeded-KMeans* proposed in [BBM02]. We observe that since the seeded clustering can be noisy due to the learner's generalization error, we pose no restrictions to the following iterations of the algorithm, which proceeds as the Lloyd's method.

Algorithm: tr-Kmeans
Input:
$\mathsf{Tr_s} = (\mathsf{X_s}, \mathsf{Y_s})$: labelled source data
K: the cardinality of label space O
$\mathsf{X_t}$: unlabelled target data
A: a supervised learner
ϵ_F: parameter describing the friends set
Output:
$\mathsf{Y_t}$: target labels on $\mathsf{X_t}$
Method:
1.Initialization
1a. Friends Selection $F \leftarrow \{x \in X_t : 1 - \epsilon_F \leq \frac{P_t(x)}{P_s(x)} \leq 1 + \epsilon_F\}$
1b. Centers generation
$h_A \leftarrow A(X_s)$
$Y_F \leftarrow h_A(F)$
$b_i = \mu(\{x \in F : Y_F(x) = i\})$ for $i = 1, \dots, k$
2.Assignment $C = \{S_i : \forall i \in K\}$ where
$S_i = \{x \in X_t : b_i = \underset{b \in B}{\operatorname{argmin}}\ d(x, b)\}$ for $i = 1, \dots, k$
3.Update $b_i = |S_i|^{-1} \sum_{x \in S_i} x$
4.Convergence Repeat 2-3 until C no longer changes
5.Output $Y_t = \{i \in K : x \in S_i, \forall x \in X_t\}$

Figure 4.2.2: Pseudocode of tr-Kmeans

We can expect that the gains of this method will be bi-directional: clustering helps the classification and classification helps the clustering. For the first direction, as it will be apparent in more detail in section 4.3, during the initialization step the pseudo-labeled friends hold labels from a source-trained classifier. This labeling might be noisy due to the inherent generalization error of the chosen classifier but also due to the domain-difference. Clustering phase will smooth out the noise induced by the domain-difference by applying its homogeneity criterion in the target domain. For the second direction, classification can be considered as a *teacher* on friends, giving a prior knowledge to an unsupervised method and transforming it to a semi-supervised one. By removing the randomness from the algorithm itself, not only we avoid the error-prone initialization but we expect that it will reduce its computational cost. As a final note, Kmeans was chosen because the number of clusters K is considered given (as we have access to the label space) and it is considerably one of the most famous and efficient clustering algorithms. However, it is evident that our idea is *plug-and-play*: any classifier can be used to pseudo-label friends and any semi-supervised clustering can be applied after.

4.3 Analysis

In this section we provide some insights on the capabilities of tr-Kmeans. As it is typical in the clustering literature (see [Ben15] for a nice overview), we start by defining a notion of clusterability over the optimal clusters of any dataset: the *bounded-scattering*. In particular,

we consider

Definition 4.3.1. *[λ-scattered set] For $\lambda \in \mathbb{R}^+$, a cluster S is λ-scattered iff for any $x \in S$ and the centroid $s = \mu(S)$ it holds that*

$$d^2(x,s) \leq \lambda\rho \text{ where } \rho = \frac{\sum\limits_{x,y \in S} d^2(x,y)}{|S|^2}$$

In the above definition, ρ is the average pairwise Kmeans distance between the points of the cluster S and intuitively the more scattered S is, the bigger the value of ρ will get. The choice of the squared euclidean distance reflects that this notion is in alignment with the cost function of the problem under investigation. We can extend the above definition by observing that for an integer $K > 1$ and a set X that admits an optimal clustering $C_{OPT} = \{S_1, \dots, S_K\}$, it holds that $\exists \Lambda \in \mathbb{R}^+$ s.t. every optimal cluster S_i is Λ-scattered (simply $\Lambda = \max\{\lambda_i : S_i \text{ is } \lambda_i\text{-scattered}, \forall i = 1, \dots, K\}$). Therefore we will define X as a Λ-scattered set.

This definition indicates that an optimal clustering can be described by its most scattered cluster. The more homogeneous the optimal clusters are, the smaller their scattering will be and the easier it will be to discover them.

Having defined our well-clusterability notion, let us assume that X admits an optimal clustering $C_{OPT} = \{S_1, \dots, S_K\}$ for an integer $K > 1$, where none of the clusters is empty (non-degenerate case). To simplify the analysis, we consider the idealized case where the seeding clustering is given without noise. To formalize this ideal scenario, we consider the existence of an oracle $\mathcal{O}$, which given a set X and an integer K, the oracle knows an optimal Kmeans clustering for X and reveals a part of it. That is, given X and K, $\mathcal{O}$ returns a seeding clustering i.e. $\mathcal{O}(X,K) = \mathcal{A}$ where $\mathcal{A} = \{A_1, \dots, A_K\}$ and for all $i = 1, \dots, K$ it holds that $A_i \subseteq S_i$ and $|A_i| \geq 1$. The lower bound on the seeding clusters is a reasonable restriction, which guarantees that every A_i is a non-empty set, that is $\mathcal{O}$ reveals a non-trivial part of the Kmeans solution. Under these assumptions, we prove the following lemma:

Lemma 4.3.1. *Given a Λ-scattered set X and a seeding clustering $\mathcal{A}$, let $\phi_{\mathcal{A}}(X)$ denote the idealized tr-Kmeans potential on this set. Then it holds that $\exists \Lambda \in \mathbb{R}^+$ such that $\phi_{\mathcal{A}}(X) \leq (1 + 2\Lambda)\phi_{OPT}(X)$.*

To prove this lemma, let us first consider the case of only one optimal cluster $S \in C_{OPT}$ where $S \subseteq X$. To choose an initial center for this cluster, the idealized version of tr-Kmeans requests from an oracle $\mathcal{O}$ to reveal a seeding clustering for X, that is we get $\mathcal{A} = \mathcal{O}(X,K)$ and we pick as center the center of mass of the set A i.e. $a = c(A) = \frac{\sum_{x \in A} x}{|A|}$ where $A \in \mathcal{A}$ and $A \subseteq S$, resulting in a potential $\phi_A(S)$ for this cluster, i.e. $\phi_A(S) = \sum_{x \in S} d^2(x, c(A))$. The following lemma establishes a relation between the optimal Kmeans potential and the idealized tr-Kmeans cost.

Lemma 4.3.2. *Let $S \in C_{OPT}$ be an arbitrary optimal λ-scattered cluster for some $\lambda \in \mathbb{R}^+$ and $\phi_A(S)$ the idealized tr-Kmeans potential on this cluster. Then it holds that $\phi_A(S) \leq (1 + 2\lambda)\phi_{OPT}(S)$.*

Proof. In the idealized version of tr-Kmeans, the seeding oracle $\mathcal{O}$ reveals a part A of the cluster S. Since, during the first iteration, the center picked by the algorithm for this cluster is $a = c(A)$ and $s = c(S)$ is the optimal center for this cluster, then from the centroid property 4.2.1 it holds that

$$\phi_A(S) = \phi(S, a) = \phi_{OPT}(S) + |S|d^2(a, s) \tag{4.1}$$

Let $o = \operatorname{argmax}_{x \in S} d(x, s)$ denote the most extreme point of the set. We can bound the distance $d(o, s)$ using the following two observations about the bounded scattering of the cluster S:

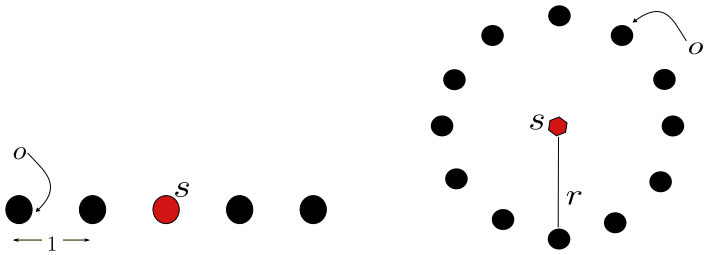
$$\sum_{i=1}^{|S|}\sum_{j=1}^{|S|} d^2(x_i, x_j) = \sum_{i=1}^{|S|} d^2(x_i, S) \overset{(1)}{=} \sum_{i=1}^{|S|} \left(d^2(s, S) + |S|d^2(x_i, s)\right) = 2|S|\phi_{OPT}(S) \tag{4.2}$$

$$\rho = \frac{\sum_{i=1}^{|S|}\sum_{j=1}^{|S|} d^2(x_i, x_j)}{|S|^2} \overset{4.2}{=} \frac{2 \cdot \phi_{OPT}(S)}{|S|} \tag{4.3}$$

Since the cluster S is λ-scattered then for all $x \in S$ it holds that $d^2(x, s) \leq \lambda \cdot \rho$. Therefore using 4.3 we can conclude that for all $x \in S$ it holds that $d^2(x, s) \leq 2\lambda|S|^{-1} \cdot \phi_{OPT}(S)$. The existence of the oracle $\mathcal{O}$ guarantees that a lies inside the convex hull of S therefore $d(a, s) \leq d(o, s)$. Therefore from 4.1 it holds that $\phi_A(S) \leq (1 + 2\lambda)\phi_{OPT}(S)$ which concludes our proof.

∎

To motivate this result, we present two tight cases of the above approximation result as depicted in Figure 4.3.1. For the first example, we consider S as a chain of five equidistant points, i.e. $S = \{(0, 0), (1, 0), (2, 0), (3, 0), (4, 0)\}$ as shown in Figure 4.3.1a. We can observe that $\phi_{OPT}(S) = 10$ and $\rho = 4$ according to equation 4.3. The worst selection is to choose any of the corner points. Since $d^2(o, s) = 4$, then $\lambda = 1$ and the result in lemma 4.3.2 states that the cost of tr-Kmeans would be at most 3 times worst than the optimal. In fact, for $A = \{o\}$ where o a corner point of the chain, we can easily see that $\phi_A(S) = 30 = 3\phi_{OPT}(S)$. Another case is to consider S as a circle of equidistant points with radius r as depicted in Figure 4.3.1b. We can observe that $\phi_{OPT}(S) = |S|r^2$ and $\rho = 2 \cdot r^2$ according to equation 4.3. The worst selection is to choose only one point from the circle, since for a bigger number of selected points the tr-Kmeans center will lie inside the convex hull of the circle, therefore it will be closer to the optimal center s. Since $d^2(o, s) = r^2$, then $\lambda = 0.5$ and the result in lemma 4.3.2 states that the cost of tr-Kmeans would be at most 2 times worst than the optimal. In fact, for $A = \{o\}$ where o is any $x \in S$ we can easily see that $\phi_A(S) = \sum_{x \in S} d^2(o, x) = |S|^{-1} \sum_{y \in S} \sum_{x \in S} d^2(y, x) = 2\phi_{OPT}(S)$, where the last equality holds from equation 4.2.



(a) Chain example (b) Circle example

Figure 4.3.1: Approximation Examples - in every figure, the points of the cluster S are depicted in black and the red point is the optimal center s. A polygone shape indicates a center that does not belong to the cluster. With o we denote the most extreme point of the set.

So far we showed what happens if the center of S was the first center produced by the idealized version of tr-Kmeans. Our next step is to show what happens to S if the center set B selected by idealized tr-Kmeans method is already non-empty, that is $|B| > 0$.

Lemma 4.3.3. *Let C be the clustering produced so far from the idealized tr-Kmeans method and let $S \in C_{OPT}$ be an arbitrary optimal λ-scattered cluster for some $\lambda \in \mathbb{R}^+$ that has not been visited yet and $\phi_A(S)$ the idealized tr-Kmeans potential on this cluster. If we add to C a center chosen according to tr-Kmeans then $\phi_A(S) \leq (1+2\lambda)\phi_{OPT}(S)$.*

Proof. As before, in the first iteration of the idealized version of tr-Kmeans, the seeding oracle $\mathcal{O}$ reveals a part A of the cluster S. Let $a = c(A)$ denote the center picked by the algorithm for this cluster and $s = c(S)$ denote the optimal center for this cluster. If we denote as $D(x)$ the shortest distance from a data point $x \in X$ to the closest center so far, then

$$\phi_A(S) = \sum_{x \in S} \min(D(x), d(x,a))^2 \leq \sum_{x \in S} d^2(x,a) \leq (1+2\lambda)\phi_{OPT}(S)$$

where the last inequality follows from Lemma 4.3.2. ∎

Therefore, with lemmata 4.3.2 and 4.3.3 we guarantee that in the first iteration of the idealized tr-Kmeans, the cost for a single optimal λ-scattered cluster S will be at most $(1+2\lambda)$ times worse than its optimal cost. Therefore, under the bounded-scattering notion for the set X, it is straightforward to see that

$$\phi_{\mathcal{A}}(X) = \sum_{k=1}^{K} \phi_{A_k}(S_k) \leq \sum_{k=1}^{K} (1+2\lambda_k)\phi_{OPT}(S_k) \leq (1+2\Lambda)\phi_{OPT}(X) \qquad (4.4)$$

where $\Lambda = \max\{\lambda_1, \ldots, \lambda_K\}$ and which concludes the proof of lemma 4.3.1. So far, it has been assumed that the seeding clustering is noise free. However, in the actual tr-Kmeans the seeding clustering is provided by a learner and that may introduce errors. To find the probability of an error, let us review how the seeding clustering is actually derived. A learner is given a set of labelled examples from a different yet similar source context C_s and picks a hypothesis h that best approximates the underlying target function in that context. As described

in definition 3.1.1, this hypothesis is selected because it minimizes a specified loss function L (that is, L is part of the learning model). Therefore the expected error of the learner is defined as $R_s(h) = E[L(h(x), y)]$ where the expected value is computed over the joint distribution $P_s(x, y)$ that governs C_s. By choosing the loss function L to be the indicator function we can see that the source expected error of the learner is $R_s(h) = Pr[h(x) \neq y] = \epsilon_s$. That would be the probability of an error if h was applied on new instances generated from the marginal distribution $P_s(x)$. However, in our case h is applied on instances generated from a different target distribution $P_t(x)$, therefore we have no concrete idea of what would be the expected error of the learner in such a setting.

The good news is that we do not apply h to any kind of target instances, but we select the ones that are very similar to P_s, the friends. Reviewing the definition 4.2.1, as similarity measure we chose the ratio P_t/P_s and friends F are the target instances whose ratio is within a user-defined radius ϵ_F around 1. The reason why this similarity was selected is because it allows us to assess the expected error of the learner in the target context as follows:

$$\begin{aligned} R_t(h) &= \underset{(x,y)\leftarrow P_t(x,y)}{E}[L(h(x), y)] \\ &= \sum_{(x,y)} L(h(x), y) P_t(x, y) \\ &= \sum_{(x,y)} L(h(x), y) \frac{P_s(x, y)}{P_s(x, y)} P_t(x, y) \\ &= \sum_{(x,y)} L(h(x), y) \frac{P_t(x) \cdot P_t(y|x)}{P_s(x) \cdot P_s(y|x)} P_s(x, y) \\ &= \sum_{(x,y)} \frac{P_t(x)}{P_s(x)} L(h(x), y)\, P_s(x, y) \end{aligned} \tag{4.5}$$

Therefore we can observe that

$$1 - \epsilon_F \leq \frac{P_t(x)}{P_s(x)} \leq 1 + \epsilon_F \overset{4.5}{\Longrightarrow} (1 - \epsilon_F) R_s(h) \leq R_t^F(h) \leq (1 + \epsilon_F) R_s(h) \tag{4.6}$$

In fact, based on Theorem 3.1.1 we can even measure this target error from the empirical source error provided the capacity of the learner (for example, binary linear classifiers have VC-dimension equal to $m + 1$ where m is the dimension of the input space). Therefore, using again the indicator as a loss function, we can conclude that the probability of the learner making a wrong prediction in the target domain is $R_t(h) \leq (1 + \epsilon_F)\epsilon_s$. At this stage, we can observe two potential sources of noise being introduced in the seeding clustering: the first source is the intrinsic expected error of the learner ϵ_s and the smaller it gets, the more accurate our seeding clustering will be. The second source of noise is the difference ϵ_F between the source and target contexts as measured by the ratio. Since this is a user-defined parameter, it can be as small as it gets up until the point where it still produces a reasonably sized seeding clustering ($|F|$ must be at least K, the number of clusters). In the case of negative transfer, namely when the source and target domain are unrelated, this pseudo-labeling of

friends might give very noisy results, but for tr-Kmeans the initialization step will just seem as any random initialization, falling back to the traditional Kmeans heuristic. This is expected, since in the absence of any prior information, unsupervised learning is the only paradigm to follow.

Going back to the lemma 4.3.1, we could argue that the probability that the result holds (i.e. the probability that the learner gives a noise-free seeding clustering) is the union bound over the misclassified friends, that is $|F|(1+\epsilon_F)\epsilon_s$. However, such a statement looks meaningless for a probability measure and in fact it is rather pessimistic since it has been experimentally observed (see chapter 5) that friends are located in high-density areas and not on the margins between clusters. Therefore it is highly likely that the seeding clustering will be very noisy and we leave the investigation on this matter for further research.

As a final note, the time complexity of tr-Kmeans cannot be asserted in a rigid way, since the supervised learner in the initialization phase of the clustering, as well as the estimation of the density ratios, are defined by the user. Given that the input size is $O(m \cdot (N_s + N_t))$ where m is the dimension of the input and N_s, N_t the size of the source and target data respectively, we observed experimentally that in the presence of a linear supervised learner (such as the Naive Bayes classifier), the computation was dominated by the standard Kmeans which is known to have worst case time complexity exponential to the input size. Interestingly enough, it has been shown in [RW16] that Lloyd's method is actually trying to solve much harder problems, i.e. PSPACE-complete problems, providing an explanation for its worst-case running time. In practice, assuming that t are the iteration steps until Lloyd's method converges to a local minimum, the time complexity of this heuristic is $O(t \cdot m \cdot N_t)$. In the experiments performed in Chapter 5, we observed that $t < 20$, making the practical performance of tr-Kmeans linear with respect to the input.

5 Experimental Results

In order to assess the quality of the proposed tr-Kmeans method, in this chapter we carry out experiments on several datasets. In Section 5.2 we consider a set of two-dimensional toy problems on domain adaptation with different topologies. In Section 5.3 we consider a real domain adaptation problem in the framework of topic classification on documents. Before presenting the results, in section 5.1 we mention the tools we will use to conduct the experiments.

5.1 Useful Tools

5.1.1 Evaluation Strategy

For all the experiments, the shared label space O between the domains is binary, so it can take only two values: 1 (Positive) and 0 (Negative). True labels are available for both the source and the target-domain instances but the target labels are used only for a quantitative assessment of the effectiveness of the proposed method and they are not taken into account in the training phase. But how shall we define the term *effectiveness*? As described in chapter 3, a domain adaptation algorithm will try to predict the target labels. Given the set of predictions, what can we say about the quality of the algorithm? Effectiveness or performance measure or evaluation strategy is precisely the way that we assess the learning efficiency. In our scenario, we could for example consider as a performance measure the misclassification rate, i.e. the ratio of the labels that were wrongly predicted. Recall that the expected error of choosing a hypothesis $h_a \in H_A$ was defined as

$$R(a) = \underset{(x,y)\leftarrow P(x,y)}{E} [L(h_a(x), y)]$$

where L is a loss function that intuitively quantifies how much an error can cost. Since the joint distribution $P(x, y)$ is unknown, we also defined the empirical error as

$$\hat{R}(a) = \frac{1}{n} \sum_{i=1}^{n} L(h_a(x_i), y_i)$$

Let us define the loss function L as the *0-1 loss* function, i.e. $L(h_a(x), y) = I(h_a(x) \neq y)$ so that $L(h_a(x), y) = 1$ iff $h_a(x) \neq y$ otherwise $L(h_a(x), y) = 0$. We can observe that

the misclassification rate corresponds to the empirical error with the 0-1 loss function and intuitively measures how often the algorithm was wrong. Alternatively, we can check how often the algorithm is right, which is another performance measure often called *accuracy*. It is straightforward to see that accuracy $= 1 -$ misclassification rate. Although accuracy is a widely used performance measure, there is a small trap when it is practically used.

Considering a binary label space as in our case, what happens if the true target classes are imbalanced? For instance, most of the data belong to the Positive class (therefore their true label is 1) and very few belong to the Negative class (therefore their true label is 0). A simple classifier that just assigns everything to the Positive class, will actually exhibit high learning efficiency based on the above definitions. To solve this, many different performance measures have been proposed (see [FK15] for a nice overview). However, these measures are label-dependent, meaning that if we interchange the labels on the data the learning algorithm will exhibit different learning efficiency. In order to avoid the label-dependency, we chose to use accuracy as our performance measure. For the results to be meaningful, we make sure that our experiments contain balanced classes.

5.1.2 Density Ratio Estimators

As presented in Chapter 4, our algorithm is plug-and-play, meaning that it can adjust to the needs of every dataset. So for every experiment, there are three choices to be made: a density ratio estimator, a classifier and a semi-supervised clustering method. The selection of the density ratio estimator will prepare the way to identify the friends in the target domain. From the discussion in Section 3.2, we will showcase the results of the uLSIF and RuLSIF estimators available in [Lab]. The parameters for these estimators are set to the default suggestions provided in the code. Although in [YSK+11] they comment that the higher the relative parameter a gets the better is the estimation quality of the RuLSIF, the parameter a should be carefully tuned so as to smooth out but not to reduce significantly the complexity of the true density-ratio function. To balance the tradeoff between these observations, it was experimentally chosen to set $a = 0.5$, estimating in effect the 0.5-relative density ratio

$$w_{0.5}(x) = \frac{2 \cdot p_t(x)}{p_t(x) + p_s(x)}$$

where we recall that $p_t(x)$, $p_s(x)$ are the unknown marginal distributions that generated the target and source instances respectively. We will denote the ratio estimation as $\widehat{w}_{0.5}(x)$.

5.1.3 Friends Selection

Given the density ratio estimations, we have to select the parameter ϵ_F that characterizes the friends set, as described in Definition 4.2.1. An important aspect for choosing this parameter is the cardinality of the resulted friends set $|F|$: it should not be too small, because we want to initialize the cluster centroids to high-density regions of the target domain, but it should also not be too big, because we want to avoid the source domain bias. With this line of thought, we decided to fix the friends cardinality $|F| = \beta \cdot N_t$ where N_t is the cardinality of

the total target set X_t and β was manually set to 0.2. Now we can compute the parameter ϵ_F that provides us with this careful abundance of friends. Therefore our goal is

$$\text{find } \epsilon_F \text{ s.t. } |\{x \in X_t : |\widehat{w}(x) - 1| \leq \epsilon_F\}| = \beta \cdot N_t$$

In practice, it has been observed that the ratio estimators do not always return values in the $[0, 1]$ range. After investigation, we observed that this fact is related with the data representation as well as the method itself, i.e. different estimators produce different value range. Moreover, let us assume that for some $x \in X_t$ an estimator E returns $\widehat{w}(x) = 23$. Since the estimated ratio has such a large value, can we conclude that x has a large similarity with the source domain or that it is very dissimilar precisely because it is far away from the value 1?

In order to bypass this issue, we created the following heuristic: given an instance x, if the estimator E can distinguish well between the two source classes, then this instance is a friend. To explain this heuristic in more detail, recall that so far the input to the selected estimator is the target set X_t and the source set X_s : $E0 = E(X_s, X_t)$. Since the source labels can be used during training, we deploy two versions of E: the first contains only the source data that belong to the Positive class $E1 = E(X_{s1}, X_t)$ and the second contains only the source data that belong to the Negative class $E2 = E(X_{s2}, X_t)$. Intuitively, the estimator $E1$ estimates the target similarity with the source Positive class and $E2$ estimates the target similarity with the source Negative class. We argue that given an instance $x \in X_t$, if $\widehat{w}_1(x) \simeq \widehat{w}_2(x)$ then the estimator E cannot distinguish between the source classes so the instance x is not similar with the source domain. So we alter the previous goal as follows

$$\text{find } \epsilon_F \text{ s.t. } |\{x \in X_t : |\widehat{w}_1(x) - \widehat{w}_2(x)| \geq \epsilon_F\}| = \beta \cdot N_t$$

This goal is very easily implemented since we only need to sort in a descending order the differences among the ratio estimations. Furthermore, before calculating the differences we normalized separately the estimations $\widehat{w}_1, \widehat{w}_2$ in the range $[0, 1]$ to smooth out the effect of range-value imbalances. For example, let us assume that $\widehat{w}_1(x) = 0.7$ and $\widehat{w}_2(x) = 1.3$. If we consider a radius 0.3 around the value 1, these estimations translate to the same case: the ratio for the instance x is 0.3 units away from 1, so the estimator could not distinguish between the source classes. The difference calculation however will not reflect this intuition. By normalizing appropriately[1] the values that are greater than 1, we expect to derive a more meaningful friends' selection.

5.1.4 Classifiers

In order to initialize tr-Kmeans, we train a classifier on the source domain and apply it to the friends of the target domain. After that, the initial centers will be the centroids of this pseudo-labeling. It is therefore crucial to select a classifier appropriate for the data under investigation. For instance, in the case of documents it is widely argued that the simple Naive Bayes classifier (see [Ng] for a nice overview) performs remarkably well.

[1] The further they are from the value 1, the closer they will be to the value 0 after normalization.

Despite the big variety, for our experiments we only required two simple classifiers: the Naive Bayes and the K-Nearest Neighbour classifier (see [Sut12] for a nice overview). Naive Bayes is a simple yet core technique for building algorithms for classification, quite popular in the text retrieval community since the '60s. The main idea behind this technique is the use of Bayes' theorem in the classifier's decision rule

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Intuitively, the Naive Bayes classifier tries to learn the joint distribution that generated the labeled train instances (in our case, this is the source set X_s along with their labels). Although in [Ng] a more detailed overview is available, we mention one important aspect of this classifier: it makes the assumption that all the features that represent our data are independent from each other. Such an assumption might seem restrictive, but it works remarkably well in practice on specific data such as documents and reduces significantly the time complexity of learning. In the case some feature has value 0, it is typical to introduce a small value indicating the lowest value a feature can get. This is often called Laplacian smoothing and through the experiments we will set it to be 1.

The K-Nearest Neighbour classifier is a lazy classifier which given a labelled train and an unlabeled test set, assigns for every instance in the test set the majority vote of its K closest neighbours from the train set. The definition of closeness depends on the nature of the data and we will specify it along with the value K in every experiment.

5.1.5 Variations of Kmeans

The traditional version of Kmeans heuristic uses the squared Euclidean distance as the cluster homogeneity's criterion. Depending on the nature of the data, this criterion might need to be modified. In the following experiments, we either use the traditional criterion or another really popular distance measure: the cosine similarity. Given two vectors $v, u \in \mathbb{R}^m$, the cosine similarity between them is defined as

$$cos(v, u) = \frac{v \cdot u}{\|v\|\|u\|} = \frac{\sum_{i=1}^{m} v_i \cdot u_i}{\|v\|\|u\|}$$

where as usual the notation $\|\cdot\|$ denotes the Euclidean norm. This is a measure that actually calculates the cosine of the angle between the vectors so it intuitively expresses how much the two vectors are pointing in the same direction. If we make sure that the vectors are normalized such that $\|v\| = \|u\| = 1$ then we can define the cosine distance to be $1 - cos(v, u) = 1 - v \cdot u$. If we use this distance modification in Kmeans then we result to the often called *spherical-Kmeans* and in every experiment we will explicitly state which variation is being used and we will try to justify our choice.

5.2 Synthetic Data

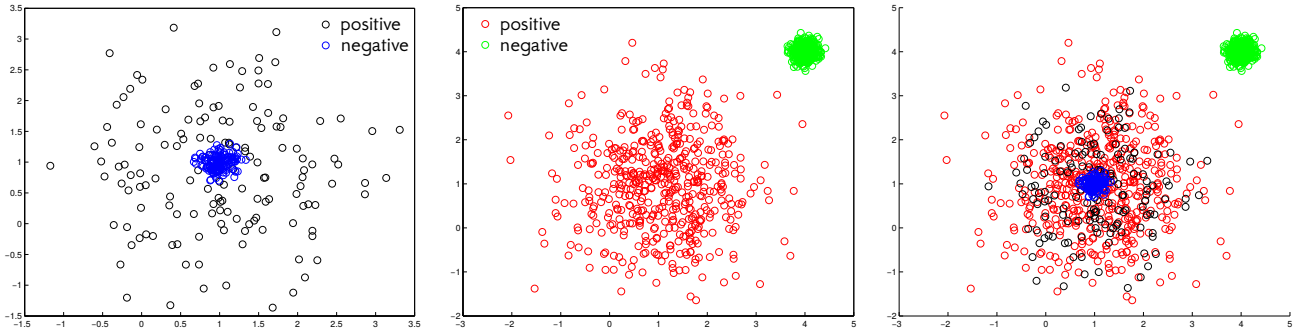
In this Section we present 3 different synthetic experiments: S1, S2, S3. Each of these experiments has been constructed in order to study the effectiveness of our method in every stage.

5.2.1 Experiment S1

For the experiment S1 we generate two-dimensional data from similar multivariate normal distributions, a generalization of the normal distribution

$$\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

to higher dimensions. We consider the covariance matrix Σ to be $\Sigma = \sigma^2 I$ where I is the identity matrix. Since Σ is diagonal, the instances generated bellow will be simply a collection of independent Gaussian random two-dimensional variables with mean μ and variance σ^2 respectively. In particular, the source instances X_s consist of 150 instances randomly drawn from $\mathcal{N}((1,1),1)$ for the Positive class (black color) and 150 instances randomly drawn from $\mathcal{N}((1,1),\frac{1}{8})$ for the Negative class (blue color). The target instances X_t consist of 500 instances randomly drawn from $\mathcal{N}((1,1),1)$ for the Positive class (red color) and 500 instances randomly drawn from $\mathcal{N}((4,4),\frac{1}{8})$ for the Negative class (green color).

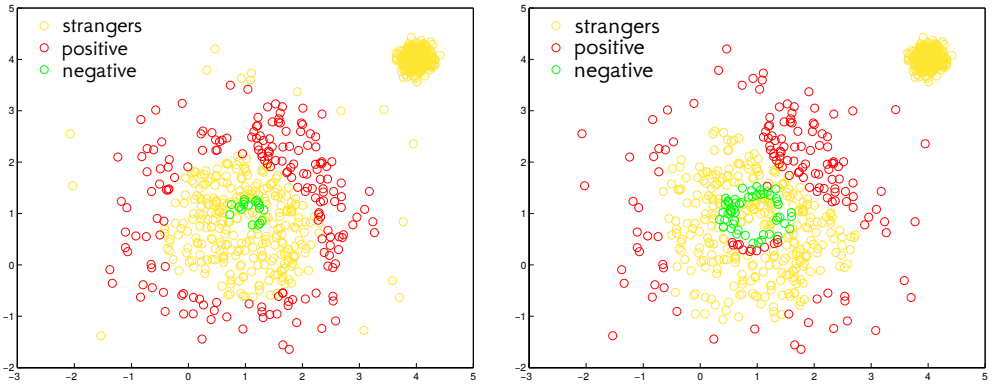


(a) source (b) target (c) combination

Figure 5.2.1: Synthetic experiment S1

As we can observe in Figure 5.2.1 Positive class in both domains is drawn from the same distribution. The target Negative class is related (mean-value modification) but not the same with the source Negative class. So during phase 1 (separation phase), we expect that the density ratio estimator will identify many friends located in the target Positive class. As discussed in the previous Section, we use the uLSIF and RuLSIF estimators for friends identification.

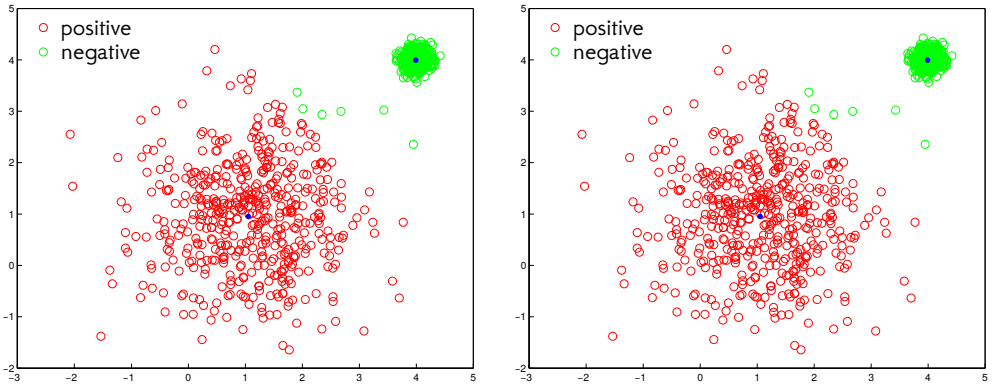
Both estimators, as we can observe in Figure 5.2.2 (for the moment let us ignore the color assignment), located friends mostly at the boundaries of the Positive target class. This was due to the ratio normalization that we discussed in Section 5.1, because they produced many values out of the $[0,1]$ range. One interesting difference is that the RuLSIF estimator not only located friends at the boundaries of the Positive target class but also to its center. Based on



(a) uLSIF (b) RuLSIF

Figure 5.2.2: Friends Pseudolabeling for S1

the topology of the source domain, this is a clear indication that this estimator made a harder work to detect friends, not only from the Positive class which is the same for both domains, but also for the source Negative class, which although different yet it is closely related as a slightly modified Gaussian distribution. This result is strongly linked with the fact that the RuLSIF estimated the a-relative ratio where a was tuned to 0.5: the more the a increases, the more is smoothed out the true density ratio. The next step is to train a classifier on the source domain and directly apply it on the selected friends. For this experiment, we utilized the K-Nearest Neighbour with $K = \sqrt{N_t}$ as it is typically used and we applied the euclidean distance to define the closeness between the data, mostly due to the low dimensionality and the linear separability of the target classes. Giving more focus now to the color assignment in Figure 5.2.2, the overlapping of the source classes challenges not only the density ratio estimators but also the classifier. It is of no surprise that the distance-based classifier we used did not succeed to classify all friends correctly. A correct classification would label all friends solely to the Positive class, therefore an initial centroid for the Negative class would have to be chosen at random. However we can avoid randomness either by selecting the instance that is furthest from the already specified centroid or by selecting an instance with the seeding technique proposed in [AV07]. We keep the noisy classifier to showcase that such an error is acceptable from our method.

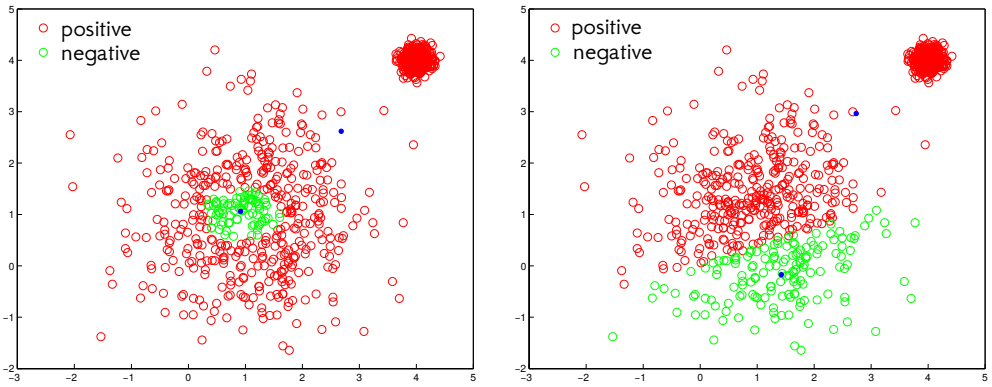


(a) uLSIF (b) RuLSIF

Figure 5.2.3: Final result of tr-Kmeans for S1

The final result of tr-Kmeans succeeded to classify the target documents with high accuracy.

It is interesting to observe in Figure 5.2.3 that both estimators led to the same result. In every clustering iteration we preserved a class correspondence between the source and the target domain by checking the closeness of each target centroid with the source centroids induced by the labelled source data. Here as well, closeness was defined in terms of euclidean distance. Despite the fact that source centroids are really close to each other due to the homocentricity of the source domain, the class correspondence has been successful for this experiment due to the friends pseudolabeling result. A naive clustering with no class correspondence between the domains, although it would succeed to identify the "well-separable" clusters, it might revert the target predictions since no information is available of what is negative and what is positive. Furthermore, a naive K-Nearest Neighbour classifier would suffer from the overlapping of the source classes. In Figure 5.2.4 we present the result obtained from naive versions of a supervised and an unsupervised approach. As a general practice, we selected the components that were employed by the tr-Kmeans in order to investigate whereas our method outperforms its building blocks and the Figure 5.2.13 shows in more detail that this is the case. In particular, we employ K-Nearest Neighbour with $K = \sqrt{N_t}$ and euclidean distance for the supervised case and standard Kmeans with random centroid selection from the target data with euclidean distance as the homogeneity criterion for the unsupervised case. In order to overcome a bad random initialization, we repeated Kmeans 30 times. As we can see, each naive method suffers from luck of additional information. In Domain Adaptation, additional related information is available and the way our method intertwines these two paradigms indicates the bi-directional gain mentioned in Chapter 4: clustering helps the classification and classification helps the clustering.



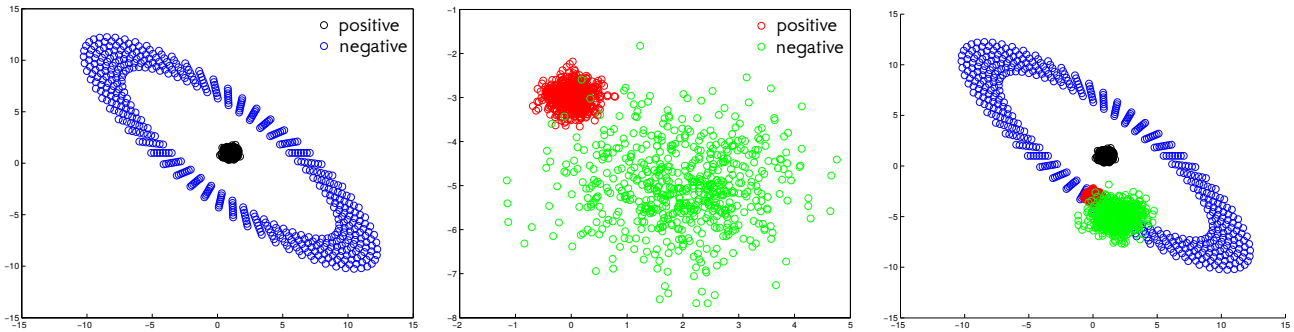
(a) Knn (b) Kmeans

Figure 5.2.4: Final result of naive supervised and unsupervised methods for S1

5.2.2 Experiment S2

In the previous experiment, all data were generated from four similar Gaussian Distributions. In the following S2 experiment we make our setup a little bit more difficult and alter one distribution to something unrelated to a Gaussian. In particular, the target instances X_t, similar as before, consist of 150 instances randomly drawn from $\mathcal{N}((0, -3), \frac{1}{4})$ for the Positive class (red color) and 150 instances randomly drawn from $\mathcal{N}((2, -1), 1)$ for the Negative class (green color). The source instances X_s consist of 500 instances randomly drawn from

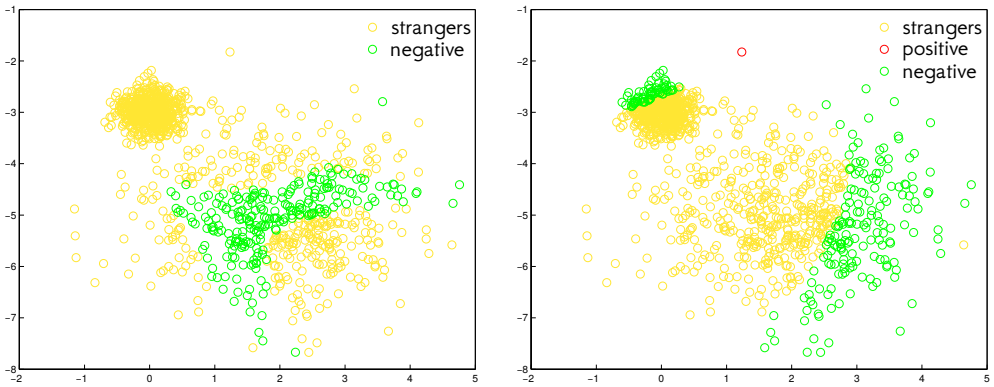
$\mathcal{N}((1,1),\frac{1}{4})$ for the Positive class (black color) and 500 instances for the Negative class (blue color) were randomly and uniformly sampled from an ellipse as depicted in Figure 5.2.5.



(a) source (b) target (c) combination

Figure 5.2.5: Synthetic experiment S2

Due to the unrelatedness between the source and target Negative classes, we expect that the density ratio estimator will identify most of the friends located in the target Positive class, since the data in this class are generated from the most related distributions. Surprisingly enough, as we can observe in Figure 5.2.6, the uLSIF estimator failed our intuition and located most of the friends in the target Negative class. This may occurred because of the high variance in the data: as we briefly mentioned in Section 3.2, the uLSIF method randomly chooses centers for the kernel functions whose combination approximates the real density ratio. Since the spread of the target Negative data is very high, most of the centers will be chosen in this class. On the opposite, the RuLSIF estimated a more smoothed version of the density ratio (as discussed in the previous experiment), therefore it successfully identified many friends also in the target Positive class as expected.

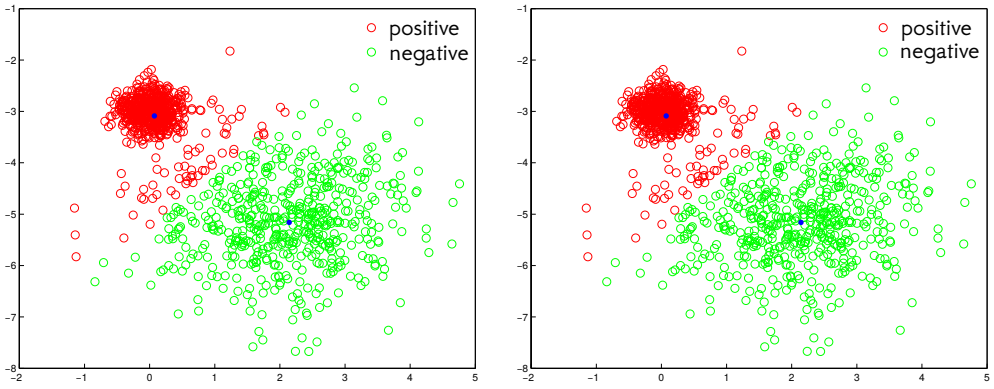


(a) uLSIF (b) RuLSIF

Figure 5.2.6: Friends Pseudolabeling for S2

Again, we utilized the K-Nearest Neighbour with $K = \sqrt{N_t}$ and we applied the euclidean distance to define the closeness between the data, mostly due to the low dimensionality and the linear separability of the target classes. Since there is no overlapping of the source classes for the part that concerns the target data (observe that the closest part of the source data to the target ones resides in the bottom left corner of the source domain as depicted in the combination section of Figure 5.2.5), the classifier was not confused, however since it is a distance-based learner, it was misguided for the friends of the Positive class as we can observe

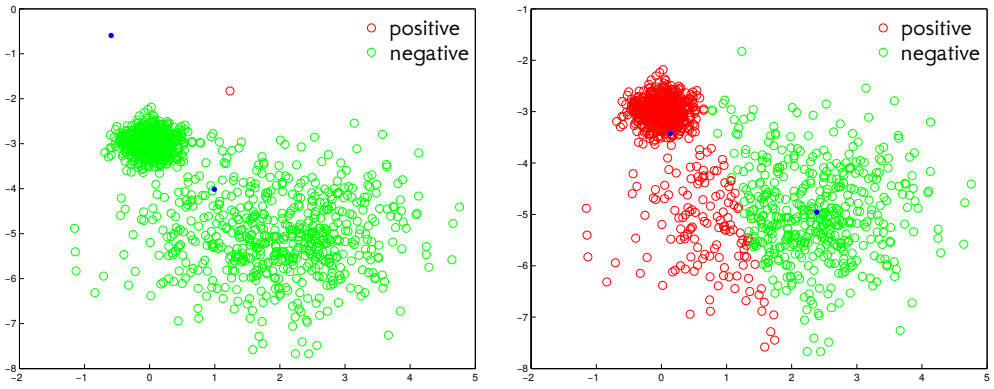
from Figure 5.2.6. In particular for the result of the uLSIF estimator, where no friends were located in the target Positive class, there is no initialization for the Positive centroid. As discussed before, we applied the simple heuristic to choose the target instance that is the furthest one (in terms of euclidean distance) from the already chosen Negative centroid.



(a) uLSIF (b) RuLSIF

Figure 5.2.7: Final result of tr-Kmeans for S2

Given our initialization and the easiness of class correspondence, the final result of tr-Kmeans powered with euclidean distance, succeeded to classify the target documents with high accuracy. As we can observe in Figure 5.2.7 both estimators led to the same result. To investigate the gains of our method, in Figure 5.2.8 we present the result obtained from naive versions of a supervised and an unsupervised approach. In particular, we employ K-Nearest Neighbour with $K = \sqrt{N_t}$ and euclidean distance for the supervised case and standard Kmeans with random centroid initialization and euclidean distance for the unsupervised case. In order to overcome a bad random initialization, we repeated Kmeans 30 times. As we can see, the unsupervised naive method converged to a bad local minimum for our needs and even if it could identify the clusters properly, there is no guarantee of class correspondence with the source domain. Also, the supervised method was distant-misguided by the topology of the source domain and gave poor results. Again the Figure 5.2.13 shows in more detail the comparison of our method with its building components.

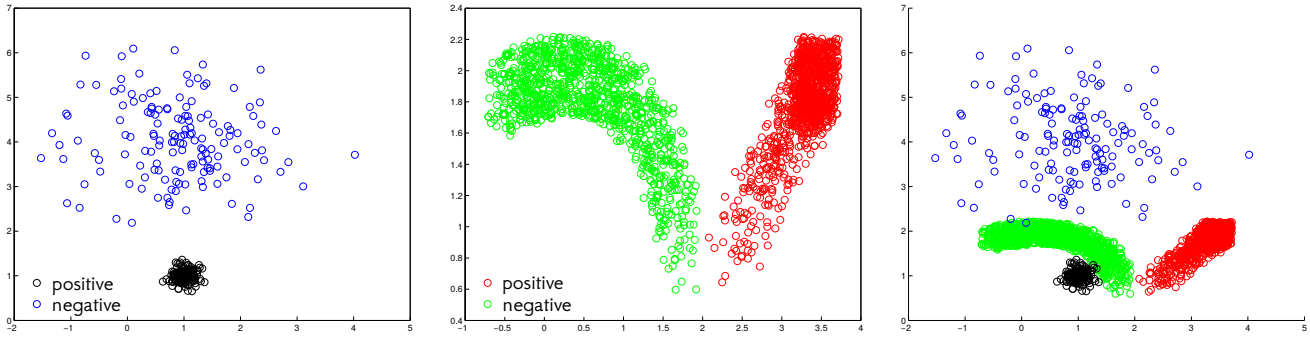


(a) Knn (b) Kmeans

Figure 5.2.8: Final result of naive supervised and unsupervised methods for S2

5.2.3 Experiment S3

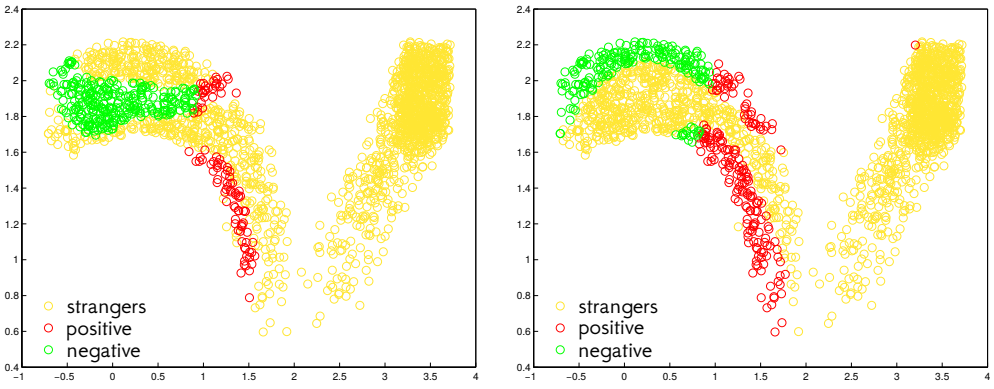
In the following S3 experiment we make our setup again a little bit more difficult and alter the whole target domain to something unrelated to a Gaussian distribution. In particular, the source instances X_s consist of 150 instances randomly drawn from $\mathcal{N}((1,1),\frac{1}{8})$ for the Positive class (black color) and 150 instances randomly drawn from $\mathcal{N}((1,4),1)$ for the Negative class (blue color). The target instances X_t, consist of 500 instances randomly drawn from a noisy sinusoidal function for the Positive class (red color) and 500 instances randomly drawn from a noisy cosinusoidal function for the Negative class (green color), as depicted in Figure 5.2.9.



(a) source (b) target (c) combination

Figure 5.2.9: Synthetic experiment S3

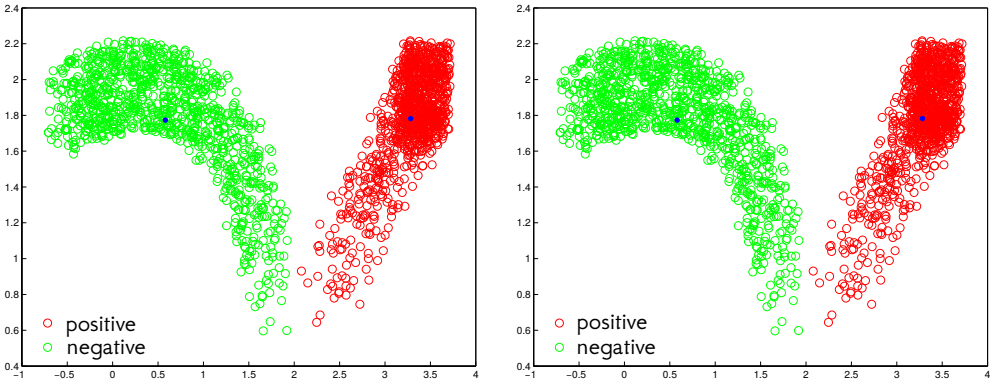
It is difficult to make any assumptions of what to expect from a density ratio estimator but Figure 5.2.10 shows again that the RuLSIF estimator smoothed out the true density ratio and as a result it located equally many friends to target areas that were closer to both source classes, as opposed to the uLSIF estimator that made a more unintuitive friends identification, yet located in high density areas of the target domain. In this experiment, we employ a different classifier since the euclidean version of the K Nearest Neighbour will be again misguided and assign most of the target data to the Positive class due to scale limitations. We therefore choose the Naive Bayes classifier with Laplacian smoothing set to 1.



(a) uLSIF (b) RuLSIF

Figure 5.2.10: Friends Pseudolabeling for S3

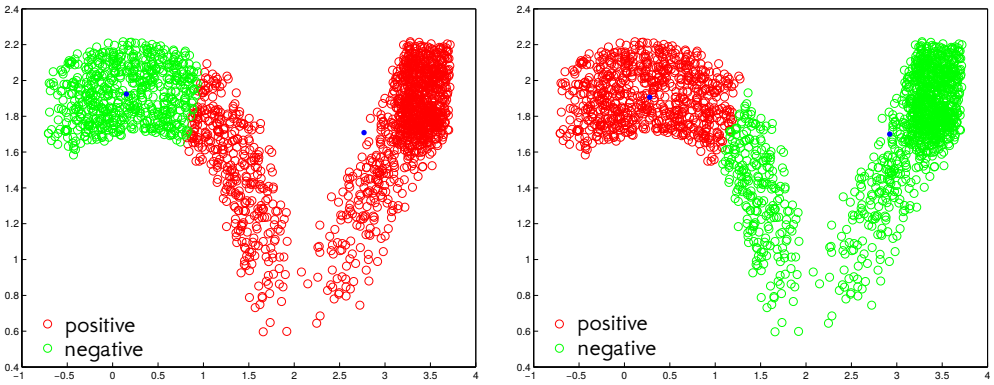
Despite the low quality of the classifier's result, the final result of tr-Kmeans powered with euclidean distance, classified the target documents with optimal accuracy in both cases, as we can observe in Figure 5.2.11.



(a) uLSIF (b) RuLSIF

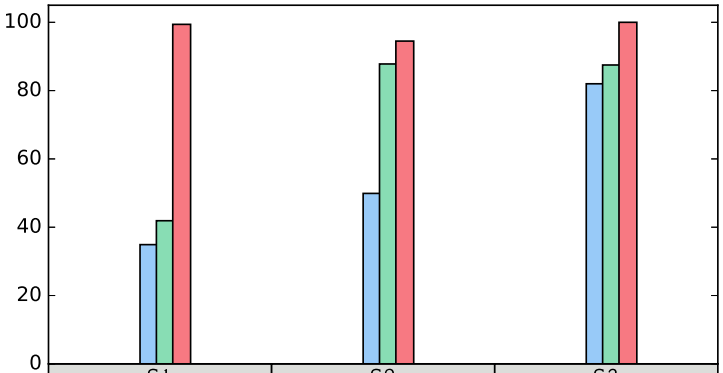
Figure 5.2.11: Final result of tr-Kmeans for S3

To investigate the gains of our method, in Figure 5.2.12 we present the result obtained from naive versions of a supervised and an unsupervised approach. In particular, we used Naive Bayes for the supervised case and standard Kmeans with random centroid selection from the target data with euclidean distance for the unsupervised case. Again to avoid bad random initialization, we repeated Kmeans 30 times. As we can see, the unsupervised naive method is sensitive to random initialization and had no information to make the class correspondence with the source domain. Also the supervised method, did not succeed to transfer well the learned generative model of the source domain. Again the Figure 5.2.13 shows in more detail the comparison of our method with its building components. For the naive Kmeans, we included the accuracy as if the class correspondence has been correct.



(a) NB (b) Kmeans

Figure 5.2.12: Final result of naive supervised and unsupervised methods for S3



	S1	S2	S3
classifier	34.9	49.9	82
Kmeans	41.9	87.8	87.5
tr-Kmeans	99.4	94.5	100

Figure 5.2.13: Accuracy Comparison on Synthetic Experiments

5.3 Real Data

In this Section we present a real domain adaptation experiment on the framework of documents. We will see that the result of tr-Kmeans is comparable with state-of-the-art domain adaptation methods, yet remaining simple and efficient.

For this experiment, we chose the widely used **20 Newsgroup** dataset, which is a collection of UseNet posts from 1993. It was originally collected by Ken Lang and contained 20.017 articles divided almost evenly among 20 different discussion topics. Articles are related by topic and date but also, some of the topics are close to each other so they are further categorized into six super-topics: recreation (rec), computers (comp), science (sci), politics (talk.politics), religion (talk.religion) and for-sale (misc.forsale). We used Jason Rennie's "bydate" version from [Ren] where documents are sorted by date, duplicates are removed and no topic-identifying headers are included. This version contains 18.846 documents in total.

Following the usual transfer learning bibliography, we created six different experiments with different source and target distributions by mixing several topics as described in Table 5.1. As in the synthetic case, for each experiment we construct a source domain (blue colour) and a target domain (red colour), each one having two classes: Positive and Negative. Similar classes in both domains are generated by the same super-topic but from different sub-topics within it. This effectively creates the difference but also the relatedness between the domains: e.g. a document from the source Positive class and a document from the target Positive class belong to different sub-topics of the same super-topic. We restrict our experiments to problems in which each document belongs to exactly one class.

Before we apply any learning system, we need to transform these human generated documents into something more manageable as discussed in Section 3.3. For every experiment, since documents from both domains are available during training time, we remove stopwords and rare words (i.e. words that appear less then 4 times in the whole corpus) and replace words by their respective word stem. Based on this pre-process, a vocabulary is being created containing a list of all the possible words/features for this experiment. Each document is therefore

DataSet	Train/Test data	Positive	Negative	Number of Samples
ds1	train	rec.{autos, motorcycles}	talk.politics{guns, misc}	3660
	test	rec.sport.{baseball, hockey}	talk.{politics.mideast, religion.misc}	3554
ds2	train	rec.{autos, sport.baseball}	sci.{med, space}	3949
	test	rec.{motorcycles, sport.hockey}	sci.{crypt, electronics}	3961
ds3	train	comp.{graphics, sys.mac.hardware, windows.x}	talk.{politics.mideast, religion.misc}	4475
	test	comp.{os.ms-windows.misc, sys.ibm.pc.hardware}	talk.politics.{guns, misc}	3623
ds4	train	comp.{graphics, os.ms-windows.misc}	sci.{crypt, electronics}	3906
	test	comp.{ sys.ibm.pc.hardware, sys.mac.hardware, windows.x}	sci.{med, space}	4888
ds5	train	comp.{graphics, sys.ibm.pc.hardware, sys.mac.hardware}	rec.{motorcycles, sport.hockey}	4894
	test	comp.{ os.ms-windows.misc, windows.x}	rec.{autos, sport.baseball}	3924
ds6	train	sci.{electronics, med}	talk.{politics.misc, religion.misc}	3369
	test	sci.{crypt, space}	talk.politics.{guns, mideast}	3821

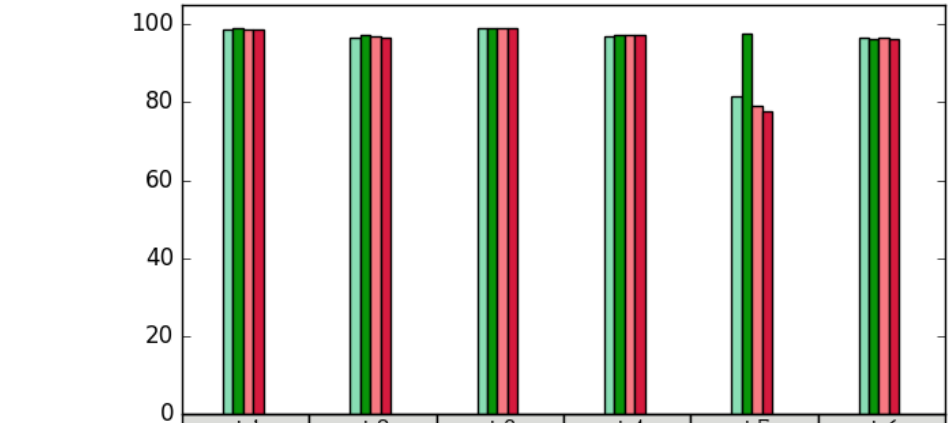
Table 5.1: 20Newsgroup - Data Sets Composition

transformed into a feature vector $x = \{x_1, \dots, x_m\}$ where $x_i = w_{x,i}$ is the importance of word i for this document in terms of tf-idf weighting.

As argued in Section 3.3, we can either choose local or global tf-idf, i.e. the document frequency is computed for each domain separately or on both domains. In Figure 5.3.1 we present the accuracy obtained by tr-Kmeans for every experiment with these two different weighting schemes. In particular, we experimented with the two proposed density ratio estimators uLSIF and RuLSIF with the relative factor a set to 0.5. To initialize the centroids, we used the Naive Bayes classifier on the identified friends. It is widely accepted that the Naive Bayes classifier performs remarkably well on documents, despite the feature independence assumption. For documents, due to the high dimensionality of the feature space, it is preferred to use the spherical Kmeans where the homogeneity criterion is based on the cosine distance of normalized vectors.

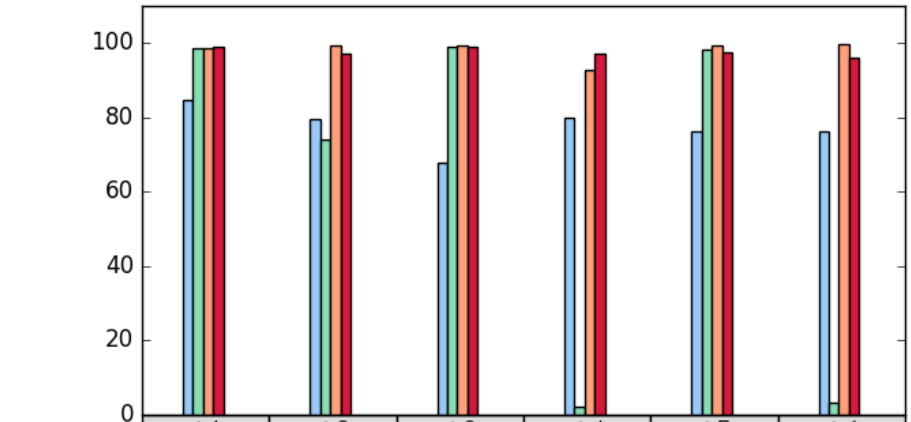
It can be observed in Figure 5.3.1 that in most experiments both weighting schemes were comparable, except in the dataset 5 where clearly the local tf-idf was better. For this reason, we can argue that the use of local tf-idf is a promising alternative weighting scheme for domain adaptation problems on documents, not only for accuracy reasons but also in terms of memory and time optimization: each domain is pre-processed independently on its own. It is important to note that the density ratio estimators were highly influenced by the data representation: a different weighting scheme may result for the same estimator to identify a different set of friends. Furthermore, we observe that both estimators gave similar results for most of the experiments but RuLSIF was slightly better, especially for dataset 5. As we saw with the synthetic experiments, since RuLSIF is more biased towards the source domain than the uLSIF method, we can argue that the dataset 5 exhibits bigger domain difference than the rest of the datasets. So despite the fact that the uLSIF method is more fast for high-dimensional data, we prefer the use of the RuLSIF estimator to better capture in a balanced way the domain divergence. Since the density ratio estimation can be included in the pre-processing step, we can refrain from taking it into account when calculating the time complexity of tr-Kmeans.

To investigate the usefulness of our method, in Figure 5.3.2 we compare tr-Kmeans with the result obtained by its domain unaware components: the Naive Bayes classifier with Laplacian smoothing set to 1 for the supervised case and the spherical-Kmeans with random centroid



	ds1	ds2	ds3	ds4	ds5	ds6
local-uLSIF	98.82	96.64	98.9	97	81.7	96.47
local-RuLSIF	98.88	97.35	99	97.2	97.6	96.2
global-uLSIF	98.76	96.82	99	97.2	79.1	96.47
global-RuLSIF	98.81	96.7	99	97.2	77.6	96.2

Figure 5.3.1: Tf-idf Comparison



	ds1	ds2	ds3	ds4	ds5	ds6
nb	84.64	79.45	67.9	79.83	76.2	76.26
kmeans	98.68	74.25	98.9	2.3	98.2	3.3
MFBRDA	98.69	99.23	99.32	92.89	99.29	99.66
tr-Kmeans	98.88	97.35	99	97.16	97.6	96.18

Figure 5.3.2: Overall Comparison

initialization for the unsupervised case. Again, in order to overcome bad initialization, we repeat spherical-Kmeans 30 times. In addition, the results of tr-Kmeans are also compared with the Multistep Fuzzy Bridged Refinement Domain Adaptation (MFBRDA) algorithm [BLZP15].

This algorithm is a fuzzy variant of the bridge refinement algorithm that was mentioned in Section 2.3. Through the use of fuzzy sets[2], a soft class assignment is assumed, which informally means that in a binary class problem, an instance can belong to both classes with some probability. Such a scenario is assumed when there is data uncertainty or it is more convenient to have a more probabilistic flavour in the predictions of the learner

[2]A fuzzy set is a set that contains not only the elements but a number associated with each element indicating membership-strength, meaning how likely is it for the element to belong to this set.

(instead of binary responses). We select this method for benchmarking since it has been already compared in [BLZP15] with many advanced domain adaptation methods and exhibited comparable accuracy performance.

As we can observe from Figure 5.3.2, in all experiments the method outperformed the shift-unaware Naive Bayes classifier. It also outperformed in most cases the naive spherical-Kmeans, which sometimes failed to guess the correct class correspondence between the domains. So far there is a strong evidence that tr-Kmeans is more accurate than its building components. Furthermore, our method outperformed in half of the experiments the MFBRDA algorithm and for the rest experiments, the accuracy difference was no higher than 3.5%. Therefore despite its simplicity, we can argue that tr-Kmeans is comparable with state-of-the-art domain adaptation methods.

6 Conclusions

This work set out to investigate the combination of the supervised and unsupervised learning paradigm in the setting of Domain Adaptation. As a result, a new transfer learning algorithm tr-Kmeans is proposed. In this final chapter, we argue on the research contributions of this thesis. Analytically, in Section 6.1 we discuss on the usefulness of the proposed method. In Section 6.2 we report experimental observations for density ratio estimators (as introduced in Section 3.2) and in Section 6.3 we review some experimental findings on document pre-processing. Finally in Section 6.4 we discuss possible directions for future research.

6.1 tr-Kmeans

The main contribution of this thesis is a transfer learning variant of the popular Lloyd's method. Powered with some prior information in the form of a different but similar domain (source domain), the algorithm tries to initiate the cluster centroids in order to obtain a good and meaningful clustering result in the target domain.

Given the prior knowledge, supervised learning is used in order to transfer this information to the target domain. By estimating the distribution ratio of the two domains, we can identify the target instances where such a transfer is more valuable, namely the target instances whose distribution appears to be more alike with the source distribution (referred to as *friends*). A classifier trained on the source domain, provides initial labels (pseudo-labels) on the friends, representing in this way the transferred knowledge. By initializing the centroids of the clustering to the mass centers of this transferred information, we theoretically guarantee that with high probability the unsupervised method will converge to a near-optimal solution.

As demonstrated experimentally the gains are bi-directional: clustering helps the classification and the other way around. For the first direction, in the initialization step, the pseudo-labeling of friends might be noisy due to the inherent generalization error of the chosen classifier but also due to the domain-difference. Clustering phase will smooth out the noise induced by the domain-difference by applying its homogeneity criterion in the target domain. For the second direction, classification can be considered as a *teacher* on friends, giving a prior knowledge to an unsupervised method and transforming it to a semi-supervised task. The unsupervised method does no longer depend only on the inherent structure of the data but is powered

with extra information to bias the search of the implicitly defined hypothesis space of the target task.

By removing the randomness of Kmeans, not only we avoid the error-prone initialization but we reduce its computational cost. In addition, after training the classifier, the source dataset is no longer used, resulting in an important memory reduction. Furthermore, it is evident that our idea is *plug-and-play*: any classifier can be used to pseudo-label friends and any semi-supervised clustering can be applied after. This allows the method to adapt to data-dependent needs accordingly. In the case of negative transfer, namely when the source and target domain are unrelated, the pseudo-labeling of friends might give very noisy results, but for tr-Kmeans the initialization step will just seem as any random initialization, falling back to the traditional Kmeans heuristic. This is expected, since in the absence of any prior information, unsupervised learning is the only viable paradigm to follow.

It has been experimentally observed that tr-Kmeans outperforms its domain-unaware building components and it has been compared with the MFBRDA algorithm which has been shown to compete with state-of-the-art domain adaptation methods. For half of the experiments, our method outperformed MFBRDA and for the rest, the accuracy difference was quite small. This result not only demonstrates that our method is competing with advanced domain adaptation methods but also it showcases an interesting tradeoff: in MFBRDA, for creating the bridge between the domains, it is required that source data are available for the most part of the algorithm. This approach to transfer the label structure, might suffer from high memory demands. Observing the little accuracy improvement of their method as opposed to ours, it is worth investigating for which instances such a memory overhead is actually necessary.

6.2 Density Ratio Estimators

A crucial step in tr-Kmeans is to identify where to transfer the source knowledge, i.e. find target instances that are similar with the source domain (friends set). In the proposed method, this notion of similarity is assumed to be probabilistic: a target instance whose distribution appears to be really close with the source distribution will be included in the friends set. Since all distributions are unknown, this measure is computed with the use of density ratio estimators: algorithms that given samples from two distributions A and B, will output estimations of the ratio A/B on the samples provided from A. In our case, A and B are the target and source distributions respectively.

Although we could choose as friends the target instances whose estimated density ratio appears to be close to 1, we observed in binary-class experiments that higher accuracy can be obtained if we employ class-specific density ratio estimators, that is estimators that try to separate between the two different classes that compose the source domain. Since the samples from the source domain are categorized into positive and negative instances, we can use this information to estimate the positive and the negative density ratios on the target instances, feeding the positive (resp. negative) estimator with all the target instances but only the positive (resp. negative) source instances. A target instance that is close to the source domain will belong to one of these classes, so we expect that the class-specific estimators

will disagree (have high difference on the output values for this instance) since the instance is more similar with one class than the other. On the opposite, for a target instance that is not close to the source domain, the class-specific estimators will not be able to disagree since both source classes will look unrelated with the instance. So friends are considered the target instances with the highest value-difference between these class-specific density ratios.

In our experiments, we considered two density ratio estimators: the uLSIF and the RuLSIF. Our two candidates were chosen based on the following properties: uLSIF is arguably one of the fastest proposed methods whereas RuLSIF handles unbounded ratios smoothly and in a tunable way. Both estimators almost always resulted in similar prediction accuracy, indicating that the computationally efficiency of uLSIF is more useful than the slight accuracy gain from RuLSIF. However, for experiments with high domain divergence, the RuLSIF method gave better results, arguably due to the smoothness introduced to the density ratio through the relative parameter a. Although in [YSK+11] they comment that the higher the a gets the better the estimation quality of the RuLSIF estimator, the parameter a should be carefully tuned so as not to reduce significantly the complexity of the true density-ratio function. To balance the tradeoff between these observations, it was experimentally chosen to set $a = 0.5$. As a final observation, the data representation heavily influences the result of any density ratio estimator. Therefore, the selection of an estimator and the pre-processing phase must be seen as correlated processes.

6.3 Document Pre-processing

In any machine learning task, human generated data such as documents require a pre-processing phase in order to transform into a manageable and succinct form. In this thesis, we selected the popular Vector space model where each document is considered as a bag of words, the order of which is not taken into consideration. Documents are therefore transformed into feature vectors, where each feature captures the importance of the respective word it represents. For the notion of importance, in this work we experimented with two different weighting schemes: the local tf-idf where the document frequency is calculated in each domain separately and the global tf-idf where document frequency takes into consideration the whole input of documents from both domains. We experimentally observed that the local tf-idf did not deteriorate the accuracy of the proposed method and was particularly useful for experiments that exhibit high domain divergence. This weighting scheme also exhibited memory and time optimization: each domain is pre-processed independently on its own. For all these reasons, we consider the local tf-idf a promising alternative weighting scheme for Domain Adaptation problems in the framework of documents.

Once the documents are transformed into feature vectors, one other popular pre-processing technique is to normalize this vector in the $[0, 1]$ range. This technique however is argued to smooth out the effect of outliers, which in some cases are required to track the domain divergence. In initial experiments, both in local and global flavour (i.e. normalizing each domain separately or the whole corpus), it was observed that normalization heavily influenced the density ratio estimators and the overall accuracy of tr-Kmeans. In fact, the global version

of normalization was experimentally less accurate and introduced slower convergence to the clustering phase. This technique was therefore overruled from the pre-processing phase.

6.4 Future Work

In this final section, we will discuss some possible directions for future research of this work. To begin with, it has already been mentioned that the core idea of this thesis can be seen as a meta-algorithm: in a domain adaptation problem, we identify in the target domain the instances that are friends with the source domain, use the labeled examples of the source to train a supervised learner which will later be applied to the target friends and finally we use this pseudo-labeling to initiate a semi-supervised learner in the target domain. Any supervised learner can be plugged in for the initialization phase and such a selection depends on the nature of data for a given experiment. It would be interesting to investigate if such a selection can be automated and a suitable supervised learner can be found via data statistics.

Furthermore, in our case we modified the traditional Kmeans heuristic into a semi-supervised variant to incorporate the friends' pseudo-labeling. We followed the semi-supervised clustering direction in order to investigate the combination of classification and clustering in the domain adaptation realm. We can observe however, that once some target instances have been initially labeled, any transductive learner can be used to complete the domain adaptation task, since friends and strangers belong to the same domain and are available during the training phase. Since the source domain is no longer involved, standard machine learning techniques for transductive learning can be employed and it would be of interest to go into this direction. This might also allow to see the behaviour of the proposed method in the extreme case where there is no domain divergence, namely the source and the target domains are the same. As a final note, given a rigid formulation of the noise introduced in the pseudo-labeling of friends, it could be the case that new generalization bounds for the Domain Adaptation problem might emerge. In fact, the notion of negative transfer can smoothly integrate itself in this line of thought, observing that the higher the noise in the pseudo-labeling, the higher the occurrence of negative transfer.

Focusing more on the separation phase of our method, two density ratio estimators were used: uLSIF and RuLSIF. In particular, for RuLSIF we experimentally tuned the relative parameter a to 0.5. However further study would be required in order to select and possibly automate this tuning in order to improve friends' selection. There is an indication that such an automation might be a data-dependent task. In addition, it easy to see that density ratios are asymmetric and as an alternative we could experiment with density-difference estimators (such as [SKS+12]) which are symmetric and always bounded as long as both densities are bounded.

Finally, documents form good instances for clustering as it can be observed from the visualization presented in Appendix A. We would be further interested to investigate the gains as well as the limitations of our method not only to documents but also to real data of different modalities, such as images and audio.

A Document Visualization

In this thesis, we experimented with documents as a real case scenario for the application of a domain adaptation method such as `tr-`K`means`. We employed the Vector space model to transform them into feature vectors, yet this representation gives a high dimensional flavour to our data, making it impossible to visualize them in order to extract meaningful observations. Such a visualization task might be useful to check the inherent structure of the data and the meaning of the clustering result in an informal level. In addition, visualization might help us investigate the limitations of the pre-processing phase and perhaps dictate us the need for a feature selection so that the clusters are likely to be compact and isolated, making them an easy instance for a simple clustering algorithm. Several methods exist in literature that try to reduce the feature space to a two dimensional plane while trying to preserve as much as possible the information carried by the data (see Hendrik Strobelt's lecture [Str15] for a nice overview) and we chose the method described in [FGM05].

Informally, their method is composed by three basic building blocks: text documents are pre-processed using the Vector Space model and the tf-idf weighting. These high dimensional vectors are fed to Latent Semantic Indexing (LSI) in order to extract main concepts, followed by multidimensional scaling (MDS) to gracefully descend to two dimensions. LSI is an automatic statistical technique (data-independent), that takes the matrix of the high-dimensional vectors $X = \{x_1, \dots, x_n\}$ where $x_i \in \mathbb{R}^m$ and performs a Singular Value Decomposition $X = USV^T$, where matrices U and V are orthogonal, called *left* and *right singular vectors* respectively for X and S is a diagonal matrix which can be arranged to be no negative and in order of decreasing magnitude. The positive entries of S are called *singular values* of X. SVD arises from finding an orthogonal basis for X's row space that gets transformed into an orthogonal basis for its column space: $Xv_i = s_i u_i$. Intuitively, the factorization from SVD tells us how to choose orthogonal bases so that the linear transformation imposed by X is represented by a matrix with the simplest possible form, that is, the diagonal S. After that decomposition, LSI keeps the k largest singular values of X and then projects X to this reduced k-dimensional space. The choice of k is selected in their proposed method so that it holds

$$\frac{\Sigma_k}{\Sigma_n} \geq \epsilon$$

where $\Sigma_{ii} = S_{11}^2 + ... + S_{ii}^2$ for $i = 1, ..., n$ and ϵ was experimentally set to 0.5. It is guaranteed by the principles of linear algebra that the reduced vector matrix is a good approximation of the original one, therefore we expect that there is no big loss of information in this reduction. MDS is a family of scaling methods for discovering structures in multidimensional data in order to reduce the data in two dimensions, yet preserve as possible the relations between them. Typically points representing the data are positioned into two dimensions so they minimize some energy function. The implicit optimization problem that MDS solves is: find points in the plane so the better the distances between points on the plane approximate real similarity between the original data, the lower the value of the energy function. Many energy functions exist with the most basic being the

$$E = \sum_{i \neq j} (\delta_{ij} - d(x_i, x_j))^2$$

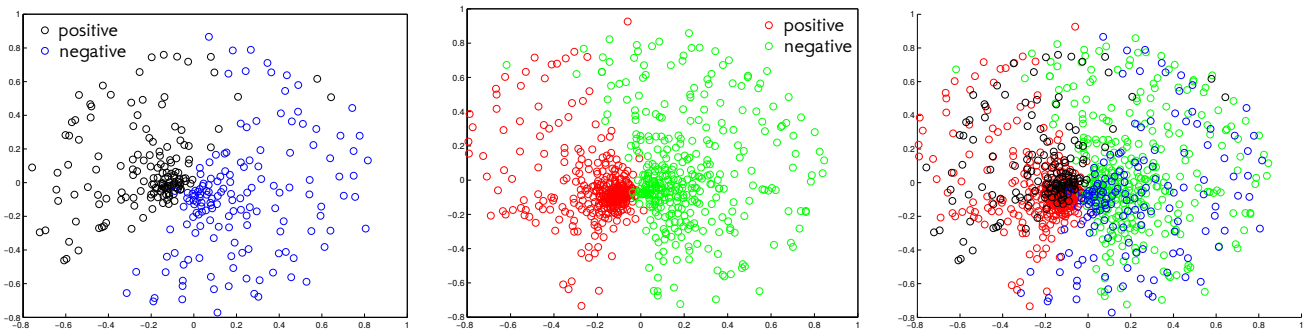
where x_i, x_j are the points in the plane, d is their euclidean distance and δ is the similarity (or the dissimilarity) between the original data.

In order to get a better insight in the nature of documents and the validity of devised transfer learning experiments, we applied this method to 3 datasets created from the 20 Newsgroup corpus:

1. RA: this dataset consists of 300 train and 700 test documents from related subcategories of Religion and Autos (alt.atheism/rec.motorcycles and soc.religion.christian/rec.autos) - Figure A.0.1

2. HP: this dataset consists of 248 train and 800 test documents from related subcategories of Hardware and Politics (comp.sys.ibm.pc.hardware/talk.politics.mideast and sci.electronics/talk.politics.misc) - Figure A.0.2

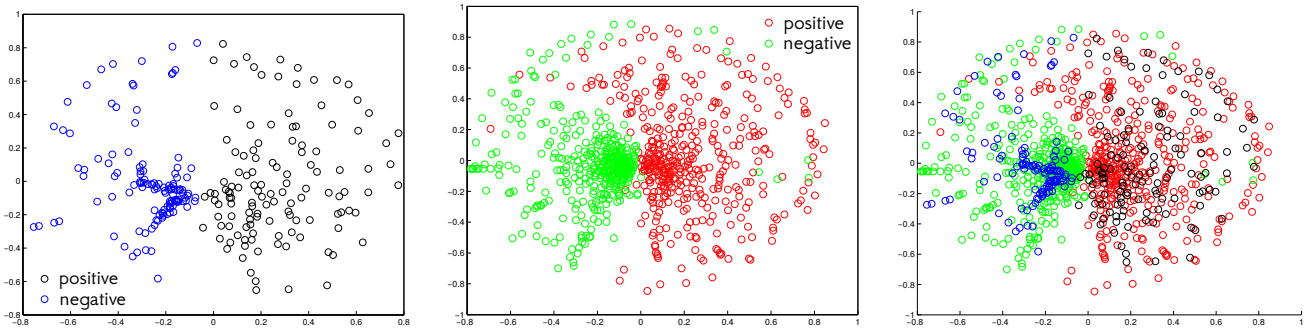
3. GS: this dataset consists of 200 train and 500 test documents from related subcategories of Graphics and Sport (comp.graphics/rec.sport.baseball and comp.windows.x/rec.sport.hockey) - Figure A.0.3



(a) source (b) target (c) combination

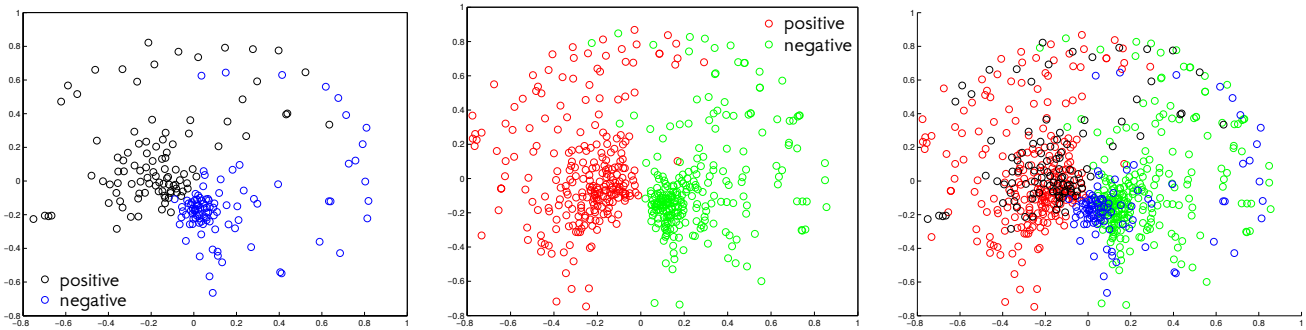
Figure A.0.1: Visualization of RA dataset

In particular, we applied the method separately for each domain, both for efficiency and in order to investigate the relation of the main concepts that LSI extracted from each domain.



(a) source (b) target (c) combination

Figure A.0.2: Visualization of HP dataset



(a) source (b) target (c) combination

Figure A.0.3: Visualization of GS dataset

As an energy function, we chose the Sammon's criterion [Sam69]

$$E = \frac{1}{\sum_{l<k} \delta_{lk}} \sum_{i<j} \frac{(d(x_i, x_j) - \delta_{ij})^2}{\delta_{ij}}$$

For similarity in the original vectors, we selected the cosine similarity as it is appropriate for documents. This weighting scheme normalizes the squared-errors in pairwise distances by using the dissimilarities δ in the original space. As a result, Sammon's criterion preserves the small δ_{ij} values, giving them more importance in this fitting procedure than the larger δ_{ij} values. In the case of documents with the tf-idf weighting, this energy function gave the most meaningful results, since we can observe in Figures A.0.1, A.0.2 and A.0.3 that related classes of both domains overlap, indicating a well-formed transfer learning experiment.

B Useful Proofs

In this Appendix, we gather proofs of Lemmata used through this thesis.

Lemma B.0.1. *[centroid property] Let $S \subseteq \mathbb{R}^m$ be a set of points with center of mass $s^* = \frac{1}{|S|}\sum_{s\in S} s$ and let $x \in \mathbb{R}^m$ be an arbitrary point. Then, for the squared euclidean distance $d(a,b) = (a-b)\cdot(a-b) = \sum_{i=1}^{m}(a_i - b_i)^2$ it holds that*

$$\sum_{s\in S} d(s,x) = \sum_{s\in S} d(s,s^*) + |S|\cdot d(s^*,x)$$

Proof. By the definition of the mass center s^*, let us observe that

$$\sum_{s\in S}(s-s^*) = \frac{1}{|S|}\sum_{s\in S}(s - \sum_{s'\in S} s') = \frac{1}{|S|}(\sum_{s\in S} s - \sum_{s'\in S} s') = 0 \tag{B.1}$$

Then:

$$\begin{aligned}
\sum_{s\in S} d(s,x) &= \sum_{s\in S}(s-x)\cdot(s-x) \\
&= \sum_{s\in S}\Big(\big((s-s^*)+(s^*-x)\big)\cdot\big((s-s^*)+(s^*-x)\big)\Big) \\
&= \sum_{s\in S}\Big(\big((s-s^*)\cdot(s-s^*)\big) + 2(s-s^*)(s^*-x) + \big((s^*-x)\cdot(s^*-x)\big)\Big) \\
&= \sum_{s\in S} d(s,s^*) + 2(s^*-x)\sum_{s\in S}(s-s^*) + |S|\cdot d(s^*,x) \\
&\overset{B.1}{=} \sum_{s\in S} d(s,s^*) + |S|\cdot d(s^*,x)
\end{aligned} \tag{B.2}$$

∎

Lemma B.0.2. *Let d denote the squared euclidean distance. Then for $a,b,c \in \mathbb{R}^m$ it holds that*

$$d(a,c) \leq 2\cdot(d(a,b) + d(b,c))$$

Proof. Using the triangle inequality of the Euclidean distance we obtain

$$\begin{aligned} d(a,c) &= \|a-c\|_2^2 = (a-c)^2 \\ &\leq ((a-b)+(b-c))^2 \\ &= d(a,b)+d(b,c)+2(a-b)(b-c) \end{aligned} \tag{B.3}$$

We can observe that

$$\begin{aligned} &((a-b)-(b-c))^2 \geq 0 \Rightarrow \\ \Rightarrow\ &(a-b)^2+(b-c)^2-2(a-b)(b-c) \geq 0 \Rightarrow \\ \Rightarrow\ &2(a-b)(b-c) \leq d(a,b)+d(b,c) \end{aligned} \tag{B.4}$$

From B.3, B.4 the result follows. ∎

Bibliography

[AGK+01] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local Search Heuristic for K-median and Facility Location Problems. *STOC*, 2001. 4.1

[ARR98] Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation Schemes for Euclidean K-medians and Related Problems. *STOC*, 1998. 4.1

[AV07] David Arthur and Sergei Vassilvitskii. K-means++: The Advantages of Careful Seeding. 2007. 4.2.1, 5.2.1

[BBM02] Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. Semi-supervised Clustering by Seeding. *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002. 4.1, 4.2.2

[BCK+08] John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. Learning Bounds for Domain Adaptation. *Advances in Neural Information Processing Systems 20*, 2008. 3.1

[BDBCP07] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. 2007. 1

[Ben15] Shai Ben-David. Computational Feasibility of Clustering under Clusterability Assumptions. *CoRR*, 2015. 4.3

[BGJV09] Indrajit Bhattacharya, Shantanu Godbole, Sachindra Joshi, and Ashish Verma. Cross-Guided Clustering: Transfer of Relevant Supervision across Domains for Improved Clustering. *ICDM 2009, The Ninth IEEE International Conference on Data Mining*, 2009. 2.3

[BHLS13] Mahsa Baktashmotlagh, Mehrtash Tafazzoli Harandi, Brian C. Lovell, and Mathieu Salzmann. Unsupervised Domain Adaptation by Domain Invariant Projection. *IEEE International Conference on Computer Vision*, 2013. 2.2.3

[BLZP15] Vahid Behbood, Jie Lu, Guangquan Zhang, and Witold Pedrycz. Multistep Fuzzy Bridged Refinement Domain Adaptation Algorithm and Its Application to Bank Failure Prediction. *IEEE Transactions on Fuzzy Systems*, 2015. 5.3

[CGH+04] Julia Chuzhoy, Sudipto Guha, Eran Halperi, Sanjeev Khanna, Guy Kortsarz, and Joseph (Seffi) Nao. Asymmetric K-center is $\log^* n$-hard to Approximate. *STOC*, 2004. 4.1

[Cha13] Pascal Chabot. The Philosophy of Simondon: Between technology and individuation. *Bloomsbury Academic*, 2013. 1

[CMM10] Corinna Cortes, Yishay Mansour, and Mehryar Mohri. Learning Bounds for Importance Weighting. *NIPS*, 2010. 3.2

[CPS13] B. J. Copeland, C. Posy, and O. Shagrir. Computability: Turing, Gödel, Church, and Beyond. *MIT Press*, 2013. 1

[Das08] Sanjoy Dasgupta. The hardness of k-means clustering. *Technical Report CS2008-0916, University of California*, 2008. 4.1

[Deu11] David Deutsch. The Beginning of Infinity: Explanations That Transform the World. 2011. 1, 1

[DH04] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. *Proceedings of the Twenty-first International Conference on Machine Learning*, 2004. 4.1

[DI07] Hal Daumé III. Frustratingly Easy Domain Adaptation. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007. 2.2.1

[DJX+09] Wenyuan Dai, Ou Jin, Gui-Rong Xue, Qiang Yang, and Yong Yu. EigenTransfer: A Unified Framework for Transfer Learning. *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009. 2

[DMM03] Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. Information-theoretic Co-clustering. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003. 2.2.3

[DrXYY07] Wenyuan Dai, Gui rong Xue, Qiang Yang, and Yong Yu. Transferring naive bayes classifiers for text classification. *In Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, 2007. 2.3

[DXYY07] Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Co-clustering Based Classification for Out-of-domain Documents. *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007. 2.3

[DYXY] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for Transfer Learning. 2.2.1

[DYXY08] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Self-taught Clustering. *Proceedings of the 25th International Conference on Machine Learning*, 2008. 2.2.3

[FG88] Tomás Feder and Daniel Greene. Optimal Algorithms for Approximate Clustering. *STOC*, 1988. 4.1

[FGM05] Blaz Fortuna, Marko Grobelnik, and Dunja Mladenic. Visualization of Text Document Corpus. *Informatica (Slovenia)*, 2005. A

[FK15] Peter A. Flach and Meelis Kull. Precision-Recall-Gain Curves: PR Analysis Done Right. 2015. 5.1.1

[Gon85] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 1985. 4.1

[HA16] Jeff Hawkins and Subutai Ahmad. Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex. *Frontiers in Neural Circuits*, 2016. 1

[Han00] L. K. Hansen. Bayesian Averaging is Well-Temperated. *Advances in Neural Information Processing Systems 1999*, 2000. 1.1

[Har75] John A. Hartigan. Clustering Algorithms. *John Wiley & Sons, Inc.*, 1975. 4.1

[HLCN+15] Nic Herndon Hongmin Li, Nicolais Guevara, Doina Caragea, Kishore Neppalli, Cornelia Caragea, Anna Squicciarini, and Andrea H. Tapia. Twitter Mining for Disaster Response: A Domain Adaptation Approach. *12th International Conference on Information Systems for Crisis Response and Management*, 2015. 2.2.2

[HMSW04] Wolfgang K. Härdle, Marlene Müller, Stefan Sperlich, and Axel Werwatz. Non-parametric and Semiparametric Models. 2004. 3.2

[HMT05] Tadayoshi Hara, Yusuke Miyao, and Jun'ichi Tsujii. Adapting a Probabilistic Disambiguation Model of an HPSG Parser to a New Domain. *Natural Language Processing - IJCNLP 2005, Second International Joint Conference*, 2005. 2.2.1

[Hoc97] Dorit S. Hochbaum. Approximation Algorithms for NP-hard Problems. *PWS Publishing Co.*, 1997. 4.1

[HPM04] Sariel Har-Peled and Soham Mazumdar. On Coresets for K-means and K-median Clustering. *STOC*, 2004. 4.1

[HPS13] Amaury Habrard, Jean-Philippe Peyrache, and Marc Sebban. Boosting for Unsupervised Domain Adaptation. *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD*, 2013. 2.2.2

[HS86] Dorit S. Hochbaum and David B. Shmoys. A Unified Approach to Approximation Algorithms for Bottleneck Problems. *Journal of the ACM*, 1986. 4.1

[IKI94] Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of Weighted Voronoi Diagrams and Randomization to Variance-based K-clustering: (Extended Abstract). *Proceedings of the Tenth Annual Symposium on Computational Geometry*, 1994. 4.1

[JMS02] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A New Greedy Approach for Facility Location Problems. *STOC*, 2002. 4.1

[KH79a] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. I: The p-centers. *SIAM Journal on Applied Mathematics*, 1979. 4.1

[KH79b] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. II: The p-medians. *SIAM Journal on Applied Mathematics*, 1979. 4.1

[KHA09] T. Kamishima, M. Hamasaki, and S. Akaho. TrBagg: A Simple Transfer Learning Method and Its Application to Personalization in Collaborative Tagging. *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, 2009. 2.2.1

[KHS09] Takafumi Kanamori, Shohei Hido, and Masashi Sugiyama. A Least-squares Approach to Direct Importance Estimation. *J. Mach. Learn. Res.*, 10:1391–1445, December 2009. http://www.jmlr.org/papers/volume10/kanamori09a/kanamori09a.pdf. 3.2

[KP13] Anastasia Krithara and Georgios Paliouras. TL-PLSA: Transfer Learning between Domains with Different Classes. *2013 IEEE 13th International Conference on Data Mining*, 2013. 2.2, 2.2.2

[KR99] Stavros G. Kolliopoulos and Satish Rao. A Nearly Linear-Time Approximation Scheme for the Euclidean k-median Problem. *ESA*, 1999. 4.1

[Lab] Sugiyama-Sato Lab. Density Ratio Software. *University of Tokyo*. 3.2, 5.1.2

[Lew98] David D. Lewis. Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval. *Proceedings of the 10th European Conference on Machine Learning*, 1998. 2.3

[Llo82] Stuart P. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 1982. 1.1

[Mit80] Tom M. Mitchell. The Need for Biases in Learning Generalizations. *Readings in Machine Learning*, 1980. 2.1

[MnSDB15] Carlos Muñoz Suárez and Felipe De Brigard. Content and Consciousness Revisited : With Replies by Daniel Dennett. *Springer International Publishing*, 2015. 1

[MNV09] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The Planar k-Means Problem is NP-Hard. *Proceedings of the 3rd International Workshop on Algorithms and Computation*, 2009. 4.1

[MS84] Nimrod Megiddo and Kenneth J. Supowit. On the Complexity of Some Common Geometric Location Problems. *SIAM Journal on Computing*, 1984. 4.1

[Ng] Andrew Ng. Generative Learning algorithms. *Stanford Lecture Notes*. 5.1.4

[ORSS13] Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, and Chaitanya Swamy. The Effectiveness of Lloyd-type Methods for the k-means Problem. *Journal of the ACM*, 2013. 4.2.1

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. *Stanford InfoLab*, 1999. 2.3

[Por80] M. F. Porter. An Algorithm for Suffix Stripping. *Program*, 1980. 2

[PY10] Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 2010. 2.1, 2

[RBL+07] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught Learning: Transfer Learning from Unlabeled Data. *Proceedings of the 24th International Conference on Machine Learning*, 2007. 2.2.1

[Ren] Jason Rennie. 20 Newsgroups. 5.3

[RM11] Yacine Rezgui and John Christopher Miles. Harvesting and managing knowledge in construction: from theoretical foundations to business applications. *Routledge*, 2011. 3.3

[RW16] Tim Roughgarden and Joshua R. Wang. The Complexity of the k-means Method. *Leibniz International Proceedings in Informatics (LIPIcs)*, 2016. 4.3

[Sam69] John W. Sammon. A Nonlinear Mapping for Data Structure Analysis. *IEEE Transactions on Computers*, 1969. A

[Sim06] Gilbert Simondon. MentalitÉ technique. *Revue Philosophique de la France et de l'Étranger*, 2006. 1

[SKS+12] Masashi Sugiyama, Takafumi Kanamori, Taiji Suzuki, Marthinus Christoffel du Plessis, Song Liu, and Ichiro Takeuchi. Density-Difference Estimation. *NIPS*, 2012. 6.4

[SNK+08] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul V. Buenau, and Motoaki Kawanabe. Direct Importance Estimation with Model Selection and Its Application to Covariate Shift Adaptation. *Advances in Neural Information Processing Systems 20*, 2008. 3.2

[SS07] Sandeepkumar Satpal and Sunita Sarawagi. Domain Adaptation of Conditional Probability Models Via Feature Subsetting. *Knowledge Discovery in Databases: PKDD 2007: 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2007. 2.2.2

[SSK04] Yogish Sabharwal, Sandeep Sen, and Amit Kumar. A Simple Linear Time (1+ ϵ) -Approximation Algorithm for k-Means Clustering in Any Dimensions. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 2004. 4.1

[SSK12] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density Ratio Estimation in Machine Learning*. 2012. 3.2

[Str15] Hendrik Strobelt. Text and Document Visualization. *Harvard School of Engineering and Applied Sciences*, 2015. A

[Sut12] Oliver Sutton. Introduction to k Nearest Neighbour Classification and Condensed Nearest Neighbour Data Reduction. 2012. 5.1.4

[SWY75] G. Salton, A. Wong, and C. S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 1975. 3.3

[TCWX09] Songbo Tan, Xueqi Cheng, Yuefen Wang, and Hongbo Xu. Adapting Naive Bayes to Domain Adaptation for Sentiment Analysis. *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, 2009. 2.3

[Thr96] Sebastian Thrun. Is Learning The n-th Thing Any Easier Than Learning The First? *Advances in Neural Information Processing Systems*, 1996. 1.1

[TKH+09] Yuta Tsuboi, Hisashi Kashima, Shohei Hido, Steffen Bickel, and Masashi Sugiyama. Direct Density Ratio Estimation for Large-scale Covariate Shift Adaptation. *JIP*, 17:138–155, 2009. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.147.4428&rep=rep1&type=pdf. 3.2

[Val84] L. G. Valiant. A Theory of the Learnable. *Communications of the ACM*, 1984. 1

[Vap98] Vladimir Naoumovitch Vapnik. *Statistical learning theory*. Wiley, New York, 1998. 3.1, 3.1

[XDXY07] Dikan Xing, Wenyuan Dai, Gui-Rong Xue, and Yong Yu. Bridged Refinement for Transfer Learning. *Knowledge Discovery in Databases: PKDD, 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2007. 2.3

[YS09] Makoto Yamada and Masashi Sugiyama. Direct Importance Estimation with Gaussian Mixture Models. *IEICE Trans. Information and Systems*, E92-D(10):2159–2162, 2009. http://www.ms.k.u-tokyo.ac.jp/2009/GMKLIEP.pdf. 3.2

[YS11] Makoto Yamada and Masashi Sugiyama. Direct Density-ratio Estimation with Dimensionality Reduction via Hetero-distributional Subspace Analysis. *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011. 3.2

[YSK+11] Makoto Yamada, Taiji Suzuki, Takafumi Kanamori, Hirotaka Hachiya, and Masashi Sugiyama. Relative Density-Ratio Estimation for Robust Distribution Comparison. *Advances in Neural Information Processing Systems 24*, 2011. 3.2, 5.1.2, 6.2