Transductive Transfer Learning

for

Computer Vision

Nazli Farajidavar

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey

Centre for Vision, Speech and Signal Processing
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

July 2015

© Nazli Farajidavar 2015
Summary

Artificial intelligent and machine learning technologies have already achieved significant success in classification, regression and clustering. However, many machine learning methods work well only under a common assumption that training and test data are drawn from the same feature space and the same distribution. A real-world application is in sports footage, where an intelligent system has been designed and trained to detect score-changing events in a Tennis single match and we are interested to transfer this learning to either Tennis doubles game or even a more challenging domain such as Badminton. In such distribution changes, most statistical models need to be rebuilt, using newly collected training data. In many real world applications, it is expensive or even impossible to collect the required training data and rebuild the models. One of the ultimate goals of the open ended learning systems is to take advantage of previous experience/ knowledge in dealing with similar future problems. Two levels of learning can be identified in such scenarios. One draws on the data by capturing the pattern and regularities which enables reliable predictions on new samples. The other starts from an acquired source of knowledge and focuses on how to generalise it to a new target concept; this is also known as transfer learning which is going to be the main focus of this thesis. This work is devoted to a second level of learning by focusing on how to transfer information from previous learnings, exploiting it on a new learning problem with not supervisory information available for new target data. We propose several solutions to such tasks by leveraging over prior models or features.

In the first part of the thesis we show how to estimate reliable transformations from the source domain to the target domain with the aim of reducing the dissimilarities between the source class-conditional distribution and a new unlabelled target distribution. We then later present a fully automated transfer learning framework which approaches the problem by combining four types of adaptation: a projection to lower dimensional space that is shared between the two domains, a set of local transformations to further increase the domain similarity, a classifier parameter adaptation method which modifies the learner for the new domain and a set of class-
conditional transformations aiming to increase the similarity between the posterior probability of samples in the source and target sets.

We conduct experiments on a wide range of image and video classification tasks. We test our proposed methods and show that, in all cases, leveraging knowledge from a related domain can improve performance when there are no labels available for direct training on the new target data.

**Key words:** Machine Learning, Classification, Transfer Learning, Domain Adaptation.

Email: N.farajidavar@surrey.ac.uk
WWW: http://www.surrey.ac.uk/cvssp
Acknowledgements

Foremost, I would like to express my sincere gratitude to my advisors Dr. Teofilo deCampos and Prof. Josef Kittler for the continuous support of my Ph.D study and research, for their patience, motivation, enthusiasm, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having better advisors and mentors for my Ph.D study.

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Timothy Hospedales, Dr. Janko Calic, Dr. Walterio Mayol-Cuevas and Dr. Terry Windeatt for their encouragement, insightful comments, and helpful questions.

My sincere thanks also goes to Dr. John Collomosse for offering me a research opportunity in his group and guiding my work on his project.

I thank my fellow labmates in the Center for Vision, Speech and Signal Processing Group: Dr. Chi Ho Chan, Dr. Paul Koppen, Dr. Cemre Zor and Dr. Pouria Mortazavian for stimulating discussions and for all the fun we have had in the last four years.

Last but not the least, I would like to thank my family: my partner Dr. Luca Remaggi, my parents Vida Khalilian and Dr. Mansour Farajidavar and my beloved brother Dr. Armin Farajidavar for supporting me spiritually throughout my life.
# Contents

1 Introduction  
1.1 Problem: Transfer Learning for Computer Vision .......................... 3  
1.2 Contributions ................................................................. 6  
1.3 Outline .............................................................................. 8  
1.4 List of Publications ............................................................... 8  

2 Background  
2.1 Notation ............................................................................. 13  
2.2 Feature Extraction Methods for Image and Video in Computer Vision ................................................................. 14  
  2.2.1 Histograms of Oriented Gradients in 3D Space-time Blocks (HOG3D)  . 15  
  2.2.2 Bag of Visual Words ......................................................... 17  
  2.2.3 Deep Convolutional Activation Feature (DeCAF) .................... 17  
2.3 Feature Selection ................................................................. 20  
  2.3.1 Principal Component Analysis .......................................... 21  
  2.3.2 Linear Discriminant Analysis .......................................... 22  
2.4 Classification ................................................................. 24
## Contents

2.4.1 Nearest Neighbor .................................................. 24  
2.4.2 Logistic Regression ................................................. 25  
2.4.3 Support Vector Machines (SVM) ................................. 26  
2.4.4 Gaussian Classifier ................................................ 29  
2.4.5 Kernel Discriminant Analysis ................................. 30  
2.5 Transfer Learning ...................................................... 31  
2.5.1 Inductive Transfer Learning ................................. 33  
2.5.2 Transductive Transfer Learning ................................. 37  
2.5.3 Unsupervised Transfer Learning ................................. 43  
2.6 Summary .............................................................. 43  

### 3 Databases

3.1 ACASVA: Player Actions Database .................................. 46  
3.1.1 ACASVA System ................................................. 46  
3.1.2 Player Action Dataset and Ground-truth Annotation .......... 52  
3.1.3 Tennis and Badminton Ground Truth Annotation System .... 56  
3.1.4 Feature Space Visualisations ................................... 57  
3.2 Chars74K Digit Dataset ............................................. 61  
3.3 MNIST-USPS Database ........................................... 62  
3.4 Caltech+Office Database .......................................... 64  
3.5 COIL20 Dataset .................................................... 65  
3.6 Summary .............................................................. 69
4 Posterior Adaptation via Feature Space Transformation 71

4.1 Introduction ......................................................... 72
4.2 Related Work ....................................................... 73
4.3 Feature Space Class-conditional Transformations ................. 75
  4.3.1 Arnold et al.’s Method and Proposed Modifications (Reweight) . . . . . 76
  4.3.2 Translating and Scaling Features (TST) .......................... 79
4.4 Experiments and Results ........................................... 81
  4.4.1 Synthetic 3-Class Dataset Experiment .......................... 81
  4.4.2 Transfer from Tennis Single to Double Matches ................. 81
  4.4.3 Badminton Dataset Experiment ............................... 85
  4.4.4 Transductive Transfer with Additional Target Adaptation Set .......... 86
4.5 Summary .......................................................... 90

5 Transductive Transfer Machine ........................................ 95

5.1 Related work ........................................................ 97
5.2 Marginal and conditional distribution adaptation .................. 97
  5.2.1 Shared space detection with MMD ............................. 100
  5.2.2 Sample-based adaptation with TransGrad ...................... 102
  5.2.3 Conditional distribution adaptation with TST ................. 104
  5.2.4 Iterative refinement of the conditional distribution ............ 106
  5.2.5 Stopping criterion ............................................. 106
  5.2.6 Classifier Selection and Model Adaptation .................... 107
Contents

5.2.7 The TTM algorithm and its computational complexity . . . . . . . . . 109
5.3 Experimental Evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 110
  5.3.1 Experiments and Results . . . . . . . . . . . . . . . . . . . . . . . . . . 111
5.4 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 120

6 Conclusions and Future work . . . . . . . . . . . . . . . . . . . . . . . . . . . . 123
  6.1 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 124
  6.2 Future Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 126

A Transductive Random Forest . . . . . . . . . . . . . . . . . . . . . . . . . . . . 129
  A.1 Related Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 130
  A.2 Decision Trees . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 131
    A.2.1 Tree training (off-line) . . . . . . . . . . . . . . . . . . . . . . . . . . . . 132
    A.2.2 Tree testing (runtime) . . . . . . . . . . . . . . . . . . . . . . . . . . . . 135
  A.3 Random Forest . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 135
    A.3.1 The training objective function of Random Forest . . . . . . . . . . . . . 136
    A.3.2 The randomness model . . . . . . . . . . . . . . . . . . . . . . . . . . . . 136
    A.3.3 The ensemble model: RF . . . . . . . . . . . . . . . . . . . . . . . . . . . 137
  A.4 Semi-supervised Random Forest . . . . . . . . . . . . . . . . . . . . . . . . . 137
    A.4.1 The Training objective function for Semi-supervised Random Forest . . . . 138
    A.4.2 Label propagation for SSRF . . . . . . . . . . . . . . . . . . . . . . . . . 139
    A.4.3 The ensemble model: SSRF . . . . . . . . . . . . . . . . . . . . . . . . . 140
  A.5 Transductive Random Forest . . . . . . . . . . . . . . . . . . . . . . . . . . . 140
# List of Figures

1.1 Traditional Machine Learning vs. Transfer Learning ........................................... 4

1.2 TTL Applications ........................................................................................................ 7

2.1 HOG3D Diagram ........................................................................................................ 16

2.2 SURF BoW Diagram .................................................................................................. 18

2.3 CNN structure ........................................................................................................... 19

2.4 PCA ............................................................................................................................ 22

2.5 Transfer Learning Diagram ....................................................................................... 33

3.1 A detailed diagram of the tennis video analysis system ........................................... 48

3.2 A simplified diagram of the tennis video analysis system (taken from [85]). .......... 49

3.3 An illustrative example of the composition of a tennis video. The length of each shot is proportional to the width of the corresponding block in the figure (taken from [85]). ................................................................. 49

3.4 Shot types: (a) crowd (b) close-up (c) play (d) commercial. ................................. 50

3.5 The GUI of the tennis video annotation system [85]. .............................................. 51

3.6 Tennis Data: Player-Actions Samples ....................................................................... 53
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.7</td>
<td>Badminton Data: Player-Actions Samples</td>
<td>54</td>
</tr>
<tr>
<td>3.8</td>
<td>Tennis and Badminton Annotation Tool</td>
<td>56</td>
</tr>
<tr>
<td>3.9</td>
<td>Tennis Data HoG3D Correlation Matrix</td>
<td>58</td>
</tr>
<tr>
<td>3.10</td>
<td>Tennis Data Projection in LDA Space</td>
<td>59</td>
</tr>
<tr>
<td>3.11</td>
<td>Tennis Data: Source and Target Kernels</td>
<td>60</td>
</tr>
<tr>
<td>3.12</td>
<td>Chars74k Database</td>
<td>62</td>
</tr>
<tr>
<td>3.13</td>
<td>USPS Database</td>
<td>63</td>
</tr>
<tr>
<td>3.14</td>
<td>MNIST</td>
<td>64</td>
</tr>
<tr>
<td>3.15</td>
<td>Caltech+Office Database</td>
<td>66</td>
</tr>
<tr>
<td>3.16</td>
<td>COIL20 Dataset-1</td>
<td>67</td>
</tr>
<tr>
<td>3.17</td>
<td>COIL20 Dataset2</td>
<td>68</td>
</tr>
<tr>
<td>4.1</td>
<td>Linear source clusters scaling: the binary classification task where the squares and crosses are representing the samples of each class; the source clusters in the left panel are scaled to better demonstrate the target distribution of the right panel.</td>
<td>79</td>
</tr>
<tr>
<td>4.2</td>
<td>Linear source clusters translation and scaling: the binary classification task where the squares and crosses are representing the samples of each class; the source clusters in the left panel are scaled and translated to better demonstrate the target distribution of the right panel.</td>
<td>80</td>
</tr>
<tr>
<td>4.4</td>
<td>Distance Metric Evaluations for KDA Kernels</td>
<td>82</td>
</tr>
<tr>
<td>4.6</td>
<td>Tennis Data: Badminton vs Different Tennis Singles Datasets</td>
<td>85</td>
</tr>
<tr>
<td>4.7</td>
<td>Serve detection failure in transfer from Badminton to Tennis, due to their different visual appearance.</td>
<td>88</td>
</tr>
</tbody>
</table>
List of Figures

4.3 Synthetic Data: Graphical Results ............................................. 92
4.5 Tennis Data: Numerical Results vs Transfer Rate Variations .......... 93
4.8 Tennis Data: Single vs Double New Results ................................. 93
5.1 Public Data: 2D Projected Graphs of TTM Performance ................... 98
5.2 TTM Diagram ........................................................................... 110
5.3 Bar chart demonstrating the same results of Table 5.4 ...................... 114
5.4 Bar chart demonstrating the same results of Table 5.6 ...................... 118
5.5 $D^{clusters}$ vs. averaged accuracy enhancement over the PCA baseline graph for Table 5.3 transfer tasks. ........................................ 119
5.6 $D^{global}$ vs. averaged accuracy enhancement over the PCA baseline graph for Table 5.3 transfer tasks. ........................................ 120
5.7 The effect of different $\gamma''$ values and number of GMM clusters in the Trans-Grad step of our framework on the final performance of the pipeline for three cross-domain experiments. Constant lines show the baseline accuracy for each experiment. ........................................ 121
6.1 Future Work Diagram 1 ................................................................. 128
A.1 Decision Tree Flowchart ............................................................... 132
A.2 Tree Testing (runtime Phase) ......................................................... 133
A.3 Twisted Moons: Graphical Results with DT, RF and SSRF .............. 145
A.4 Translated and Scaled Moons: Data Representations .................... 148
A.5 Translated and Scaled Data: Transfer Parameter Estimation Graphs .... 149
A.6 Moons: SSRF $\alpha$ and $\beta$ Study ...................................................... 151
List of Figures

A.7 Char74 Data: Noise Insertion Images .................................................. 154
A.8 Char74: Transductive-SSRF Results ...................................................... 155
A.9 Char74: Transductive-SSRF Results (Further Experiments) ..................... 156
A.10 Char74: Fine Evaluations of the Transductive-Tree Failure Case (Graphical Representation) ................................................................. 158
A.11 Char74: Transformation Estimation with Transductive-SSRF (Failure Case-Study) ................................................................. 159
B.1 Public Data: TL with NN Classifier (Further Exp) .................................. 167
List of Tables

2.1 Notation and acronyms used most frequently in this paper. .......................... 15

3.1 Summary of tennis events from [91] ......................................................... 52
3.2 Tennis Data: Clusters Data-Distributions .................................................. 53
3.3 Comparison between Tennis and Badminton. Note that, in the table, match intensity refers to the actual time the ball/shuttle was in flight, divided by the length of the match. ................................................................. 55
3.4 Summary of Badminton events ................................................................. 57
3.5 USPS data distribution [31]. ................................................................. 63
4.1 Tennis Data: semi-Supervised Domain Adaptation Exp.1 ......................... 87
4.2 Tennis Data: semi-Supervised Domain Adaptation Exp.2 ......................... 89
5.1 Classifier selection and model adaptation ............................................... 108
5.2 Classifiers evaluations on individual domains: 5-fold cross validation accuracy of a nearest neighbour classifier. All the datasets are $l_2$-normalized. .......... 111
5.3 Cross-domain dissimilarities between domains ($src \rightarrow trg$), with datasets abbreviated as M: MNIST, U: USPS, C: Caltech, A: Amazon, W: Webcam, and D: DSLR. ................................................................. 112
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4</td>
<td>Recognition accuracies with datasets abbreviated as M: MNIST, U: USPS, C: Caltech, A: Amazon, W: Webcam, and D: DSLR. Comparisons start the baseline accuracy with NN and PCA followed by the results of the discussed TTL algorithms. The last two columns show the effect of the classifier selection and model adaptation techniques (5.2.6) on JDA and TTM algorithms.</td>
</tr>
<tr>
<td>5.5</td>
<td>Recognition accuracies with NN classifiers on target domains using different shared subspace projection methods.</td>
</tr>
<tr>
<td>5.6</td>
<td>Results on Caltech+office dataset using DeCAF features.</td>
</tr>
<tr>
<td>A.1</td>
<td>Twisted Moons: Numerical Results</td>
</tr>
<tr>
<td>A.2</td>
<td>Moons: DT, RF and SSRF Numerical Performance Comparisons</td>
</tr>
<tr>
<td>A.3</td>
<td>Char74: DT vs Transductive-SSRF Numerical Results</td>
</tr>
<tr>
<td>A.4</td>
<td>Char74: Effect of PCA Dimensionality Reduction before Applying the TL Algorithm</td>
</tr>
<tr>
<td>A.5</td>
<td>Char74: ERF, TRf and ATRF Performance Evaluations</td>
</tr>
<tr>
<td>A.6</td>
<td>Public Data: JDA, Transductive-SSRF and NN-TST Performance Evaluations</td>
</tr>
<tr>
<td>A.7</td>
<td>Public Data: Transductive-RF vs Transductive-SSRF and NN</td>
</tr>
<tr>
<td>B.1</td>
<td>Public Data: Three Different Iterative Schemes for TL with NN Classifier</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction
Chapter 1. Introduction

As human beings, our learning capacity develops progressively in time as we grow. At the age of six, we recognise around 104 object categories and we keep learning more throughout our life [13]. Moreover we tend to semantically organise all our knowledge into meaningful taxonomies: concepts and categories are grouped on the basis of the common properties acquired through our five senses [3, 32, 55]. This intrinsically means that any new concept is not learned in isolation, but considering connections to what is already known, which makes analogical reasoning (the skill of building analogies) one of the cores of human intelligence [71]. This results in practical advantages: it might be easier to learn French if one already knows Italian and English and it might be easier to learn playing badminton if one already knows tennis.

In the context of computer vision, an image in the most basic representation is defined through a matrix of its pixels intensity values and the semantic organisation of an image database is known as classification where an ideal image classifier should be able to exploit complex high dimensional feature representations even when only a few labelled training samples are available. In most classification scenarios, it is expensive to acquire vast amounts of labelled training samples in order to provide classifiers with a good coverage of the feature space. One possible way of dealing with this problem is to synthesise images of training objects using computer graphics techniques (e.g. [147]), but their appearance may not be realistic and it is not possible to model all possible backgrounds. Practitioners often resort to crowd sourcing [19], but the annotations obtained are either costly or unreliable. Ideally, an image classifier should be initially capable of detecting similarities between data distributions and subsequently facilitates the exploitation of the required knowledge from all the previously trained reliable models, just as human can exploit previous experience when learning some similar concepts.

For example, when a child learns to recognise a new letter of the alphabet he will use examples provided by people with different hand-writing styles, using pens of different colors and thicknesses. Without any prior knowledge a child would need to consider a large set of features as potentially relevant for learning the new concept, so we would expect the child to need a large number of examples. But if the child has previously learnt to recognise other letters, he can probably discern the relevant attributes (e.g. number of lines, line curvatures) from the
irrelevant ones (e.g. the colour of the lines) and learn the new concept with fewer examples. This observation suggests that a sample efficient image classification algorithm might need to exploit knowledge gained from related tasks.

In the absence of such prior knowledge, a typical image classification task would require the exploration of a large and complex feature space. In such high dimensional feature spaces some features will be discriminative while most probably a large number of them are irrelevant. Additionally, out of this set of discriminative features some might help in detecting the relations in between similar tasks and hence be beneficial for a new related concept classification. While we do not know a-priori what features are irrelevant, related tasks might share these features and we can use training data from these tasks to discover them.

An initial step to this task and feature relevancy discovery is to first be able to tell which tasks can be classified as similar. While many recent works [53, 52, 138] studied this as zero-shot learning, in this thesis we assumed that our system is already aware of the task similarities between the two training (source) data an a given new unlabelled test (target) data.

The goal of this thesis is to develop efficient transfer learning algorithms for image classification that can exploit rich feature representations.

1.1 Problem: Transfer Learning for Computer Vision

Recently, there has been a significant growth in multimedia data production. This data exists in various different forms such as broadcast content (including television, DVDs, BluRays and Internet) and personal content (e.g. uploads from hand-held devices); it can also be recorded interviews or meetings, or footage from surveillance cameras. Most of this data is intended for general viewing and hence basic labelling (Date, Time and Title etc.) is attached to it which is often inaccurate. However in many cases it would be useful to add additional labels to retrieve information in a more flexible and systematic fashion (e.g. a Tennis sports video can potentially be labelled with match-events description). Such meta-data will assist in finding
material within the multimedia footage via browsing, querying or searching.

For easy retrieval of information from a very large quantity of archived images of videos, it would be very useful to have them annotated automatically i.e. to create a system that could understand the visual content (manual annotation being too unwieldy). The annotation task can become even more challenging when the new test data does not follow the same distribution as of the initial training data. Unlike traditional machine learning methods, transfer learning methods do not assume that training and test data are drawn from the same distribution [123], hence fitting well to this annotation problem.

Traditionally, machine learning algorithms were training models separately even for similar training domains. Transfer Learning algorithms try to alleviate the training by benefiting from the relations between similar domains. Figure 1.1 illustrates the difference between the learning process of traditional machine learning and transfer learning. As we can see, traditional machine learning techniques try to learn each task from scratch, while transfer learning techniques try to transfer the knowledge from some previous task(s) to a target task when the latter has fewer high-quality or labelled training data [123].

**Traditional ML vs. TL**

![Figure 1.1: Traditional Machine Learning vs. Transfer Learning](image-url)
1.1. Problem: Transfer Learning for Computer Vision

The field of transfer learning includes a range of problems in which there is a change of domain or task between source and target sets. Transfer learning techniques are becoming more popular in Computer Vision, particularly after Torralba and Efros [151] discovered significant biases in object classification datasets.

Based on Pan and Yang’s survey [123], Transfer Learning (TL) can be categorised into three types of approaches; Inductive Transfer Learning (ITL) where some labelled data in the target domain are required to induce an objective predictive model for use in the target domain, Transductive Transfer Learning (TTL) where no labelled data in the target domain are available at the training time, and finally Unsupervised Transfer Learning (UTL) where no labelled data is available in source or target domains. While much of the work focuses on inductive transfer learning problems, which assume that labelled samples are available both in the training/source and test/target domains, in this thesis we focus on the case in which only unlabelled samples are available in the test/target domain. This is a transductive transfer learning (TTL) problem, i.e., the joint probability distribution of samples and classes in the source domain, \( P(X_{\text{src}}, Y_{\text{src}}) \), is assumed to be different, but related to that of a target domain joint distribution, \( P(X_{\text{trg}}, Y_{\text{trg}}) \), but labels \( Y_{\text{trg}} \) are not available in the target set. We follow a similar notation to that of [123] (see Table 2.1).

Transductive transfer learning methods can potentially improve a very wide range of classification tasks, as it is often the case that a domain change happens between training and application of algorithms, and it is also very common that unlabelled samples are available in the target domain.

The main focus of this thesis is then in looking for a common latent representation and data transformation techniques which reduce the dissimilarities of the data configuration in the source (training) and the target (test) domains after transformation.

Recent advances in computer vision and machine learning as well as the exponential growth in the processing capacity and memory of computer technology have created the conditions where it becomes timely to investigate the possibility of designing such systems.
One of the applications is in sports footage which provides a useful test-ground given its fixed, rule-governed content. To date, there have been several annotation systems developed for different sports such as football [164], Formula 1 [128], snooker [40] and Tennis [39]. As part of the work on video data, we focus specifically on court-based games, such as Tennis and Badminton. We build upon the outcome of a previous project ACASVA (Adaptive Cognition for Automated Sports Video Annotation) to perform automatic adaptation of the trained system so that it can be applied to a different game. For instance if the main system has been trained to interpret a Tennis single match, we aim to transfer this learning to either Tennis doubles game or even a different domain such as Badminton.

Another example is in image classification where the training set may come from high quality images (e.g. from DSLR cameras) and the target test set may come from mobile devices. Figure 1.2 demonstrates instances from three different classes obtained from these two relevant domains. As one can note the image quality, illumination level and background varies from one domain to other. TTL methods can potentially generalise classification methods for a wide range of domains and make them scalable for big data problems.

In this thesis, we employ transformative approaches for solving the transductive transfer learning problem in which feature transformations are used to adapt the source domain feature distributions to the target domain. In essence, linear and non-linear joint adaptation of marginal and conditional distributions of the source and target data are introduced.

### 1.2 Contributions

The main contribution of this thesis can be listed as follows:

- **Unsupervised adaptation of the conditional distributions**: We propose unsupervised class-specific transformations for domain adaptation. The technique is based on estimating linear transformations to adapt the source domain features in order to maximise the similarity between posterior probability distribution functions (PDF) for each class in
1.2. Contributions

Figure 1.2: Sample images with varying image acquisition quality. Right: sample images captured by a Webcam camera, Left: images obtained from Amazon webpage.

the source domain and the expected posterior PDF for each class in the target domain (Chapter 4).

- **Joint adaptation of the marginal and conditional distributions:**

  We propose Transductive Transfer Machine (TTM) algorithms which combine methods that adapt the marginal and the conditional distribution of the samples, so that source and target datasets become more similar, facilitating classification (Chapter 5).

- **Two unsupervised dissimilarity measures for automating the parameter setting of the proposed data processing pipeline for a new transfer task:**

  Two dissimilarity measures are devised which enable the proposed pipeline to automatically readjust its parameters for a new transductive transfer task where the source and target distributions are from different environments (Chapter 5).
• A classifier selection and model parameter adaptation algorithm:

We proposed Adaptive Transductive Transfer Machine (ATTM) which uses the aforementioned dissimilarity measures to select the right classifier and to optimise its parameters for a new target domain. We show that our method obtains state-of-the-art results in cross-domain vision datasets using naïve features, with a significant gain in computational efficiency in comparison to related methods (Chapter 5).

• Assessment of semi-supervised random forest for transductive transfer learning:

We have also constructed a transductive transfer version of random forest and study its performance on unsupervised/semi-supervised domain adaptation problems (Appendix A).

1.3 Outline

The structure of the thesis is as follows: Chapter 2 provides the necessary background on feature extraction methods and classification algorithms. Chapter 3 describes the databases with their common protocols used in our experiments. In Chapter 4 two transfer learning methods are proposed and some modifications to the existing methods are suggested (published in [47, 49]). A novel transductive transfer learning pipeline is introduced in Chapter 5, published in [48, 46]. Finally, in Chapter 6 we draw conclusions and discuss future lines of research. In addition, in Appendix A we study the semi-supervised random forest and propose a transductive transfer random forest approach.

1.4 List of Publications

Parts of this thesis appeared in conference proceedings. The list of publications is as follows:

• N. FarajiDavar and T. deCampos and J. Kittler and F.Yang Transductive Transfer Learning for Action Recognition in Tennis Games (Poster) In 3rd International Workshop on
1.4. List of Publications

Video Event Categorization, Tagging and Retrieval for Real-World Applications (VEC-TaR), in conjunction with 13th International Conference on Computer Vision (ICCV), Barcelona, Spain 2011 [47]

• N. FarajiDavar and T. deCampos and D. Windridge and J. Kittler and W. Christmas Domain Adaptation in the Context of Sport Video Action Recognition (Poster) In Domain Adaptation Workshop, in conjunction with NIPS, Sierra Nevada, Spain 2011 [49]

• N. Farajidavar, T. de Campos, D. Windridge, J. Kittler, W. Christmas. Domain Adaptation in the Context of Sport Video Action Recognition (Poster), 4th UK Computer Vision Student Workshop (BMVW), in conjunction with the British Machine Vision Conference (BMVC), Guildford, September 2012 [50]


Chapter 2

Background
In this chapter, we review the theoretical concepts, technical ideas and background knowledge on feature extraction, classification and transfer learning. We also give a brief review of some of the apparatus used in the rest of this thesis.

The goal of classification is to take an input vector \( x \in \mathcal{X} \) and to assign to it a discrete class label \( y \in \mathcal{Y} \), where the number of classes is \( C = |\mathcal{Y}| \). In the most common scenario, the classes are taken to be exclusive, so that each input is assigned to one and only one class. The input space \( \mathcal{X} \) is thereby divided into decision regions whose boundaries are called decision boundaries or decision surfaces.

The first step is to describe the data in an appropriate manner. In the case of video, a space-time descriptor is used for that, generating a vectorial representation \( x \). This may be followed by a feature extraction and feature selection method to reduce the dimensionality and improve the discrimination power.

Data sets and parameters can easily get outdated or the training data may not always facilitate a direct generalisation to the test data. Also, one may need to use knowledge from one domain for application in another domain (e.g., detection of Spam learnt in English applied to French emails). Such issues are addressed by transfer learning techniques which are motivated by the fact that people can intelligently apply knowledge learned previously to solve new problems faster or with better solutions in different (but related) domains.

The focus of this work is on transfer learning, which is reviewed in Chapter 4. But before that, in order to provide background for our experiments, we briefly review the methods used for feature extraction and classification in Sections 2.2 and 2.4, respectively. In Section 2.5 we define the Transfer Learning in mathematical terms and briefly review the literature of its taxonomy and variations.
2.1 Notation

This section is a reference for the remaining of this thesis, where we introduce some notation and definitions. In image and action classification, as in most classification problems, systems are trained with samples from one set up and often expected to be applied to another set up. Given a sample \( x \) and a class label \( y \), the standard statistical classification task is to assign a probability, \( p(y|x) \), to the event of \( x \) belonging to class \( y \). In the binary classification case the labels are \( y \in \{0, 1\} \). Typically, each example \( x^i \) is represented as a vector of features \((x_1^i, \cdots, x_F^i)\) where \( F \) is the number of features or dimensions of the feature space (e.g., \( \mathcal{X} = \mathbb{R}^F \)). The data consists of two disjoint subsets; the training set \((\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}) = \{(x_1, y_1), \cdots, (x_{n_{\text{train}}}, y_{n_{\text{train}}})\}, \) (e.g. sample feature vectors obtained from a set of people in a set of environments, illumination conditions, camera configurations/types and with certain types of background) available to the model for its training and the test set \( \mathbf{X}_{\text{test}} = \{x_1, \ldots, x_{n_{\text{test}}}\} \), upon which we want to use our trained classifier to make predictions. A pattern recognition system is expected to perform well if the test samples \( \mathbf{X}_{\text{test}} \) are obtained in the same conditions as \( \mathbf{X}_{\text{train}} \), e.g. if they are new videos of the same people performing the same actions in the same environments as before.

However, in most application scenarios there is a change of scene, video/image quality, performing actors etc., so although the domain representation and the label set are the same, i.e. \( \mathcal{X}_{\text{train}} = \mathcal{X}_{\text{test}} \) and \( \mathcal{Y}_{\text{train}} = \mathcal{Y}_{\text{test}} \), the data distributions are different, i.e. \( p(\mathbf{X}_{\text{train}}) \neq p(\mathbf{X}_{\text{test}}) \).

The two most important definitions in the transfer learning paradigm are those of Domain and Task. A domain is composed of two components: a feature space \( \mathcal{X} \) and a marginal probability distribution \( P(x) \), where \( x = \{x_1, \cdots, x_F\} \in \mathcal{X} \). Given a specific domain, \( \mathcal{D} = \{\mathcal{X}, P(x)\} \), a task \( \mathcal{T} \) consists of two components: a label space \( \mathcal{Y} \) and an objective predictive function \( P(y|x) \). Generally in transfer learning the knowledge obtained from one or more source domains can be used for classification in a new target domain. In the simplest mode we assume that there exist two domains and two tasks, consisting of source domain, \( \mathcal{D}_{\text{src}} \) and target do-
main, $D_{trg}$ and similarly source and target tasks, $T_{src}$ and $T_{trg}$ respectively. In such circumstances transfer learning is proposed as a solution.

For ease of notation, $x^j_i$ is the feature $j$ of sample $x_i$ and $E^{src}[x_j, y]$ is used to represent $E^{src} [x_j, y]$ which is the expected value of the feature $x_j$ with label equal to $y$, $\forall x \in X^{src}$ and $n_{\text{train}}^{src}$ is the number of labelled training data in the source domain.

The usual traditional machine learning approach is to assume $D^{train} \approx D^{test}$, and treat this as a classical generalisation problem in machine learning by means of classifier regularisation [14].

In our problem $D^{train} \neq D^{test}$. It characterises a special case of transductive transfer learning which has not been covered by Pan and Yang [123] transfer learning survey where both domain and task are differing from source to target and we do not have access to any labels in the new target domain: we then aim to improve the learning of the target predictive function $P(Y^{trg}|X^{trg})$ in $D^{trg}$ using the knowledge in $D^{src}$ and $T^{src}$, where $D^{src} \neq D^{trg}$.

The solution to such a task will be either transforming the domains in a way that both domains become more similar or introducing some adaptive classifier which can be generative enough to fit well the new target domain. For that, we evaluate two methods that transform the features of $X^{src}$ so that they become more similar to $X^{trg}$ and the classifier is re-trained using the transformed samples.

The most frequently used notation of this work is summarised in Table 2.1.

### 2.2 Feature Extraction Methods for Image and Video in Computer Vision

As in any machine learning architecture, the first step in learning is to efficiently represent interesting parts of an image (or video) as a compact feature vector which is known as feature extraction [155]. The goal is to come up with a vectorial representation that is discriminative enough for a later classification purpose.
2.2. Feature Extraction Methods for Image and Video in Computer Vision

Table 2.1 Notation and acronyms used most frequently in this paper.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbf{X} )</td>
<td>Input data matrix (or alternatively a set) with ( n ) samples (members) of ( f ) features</td>
</tr>
<tr>
<td>( \mathbf{x}_i )</td>
<td>Feature vectors</td>
</tr>
<tr>
<td>( \mathbf{Y} )</td>
<td>Array of class labels associated to ( \mathbf{X} )</td>
</tr>
<tr>
<td>( \mathcal{Y} )</td>
<td>Set of classes</td>
</tr>
<tr>
<td>( \mathbf{X}_{src} )</td>
<td>Source and target data matrices</td>
</tr>
<tr>
<td>( \mathbf{X}_{trg} )</td>
<td>Source and target data matrices</td>
</tr>
<tr>
<td>( \mathcal{D} )</td>
<td>Transfer Domain</td>
</tr>
<tr>
<td>( \mathcal{T} )</td>
<td>Transfer Task</td>
</tr>
<tr>
<td>( \Lambda_{src} )</td>
<td>Classification model trained with ( \mathbf{X}_{src} )</td>
</tr>
<tr>
<td>( G(\mathbf{X}) )</td>
<td>Transformation function</td>
</tr>
<tr>
<td>( W )</td>
<td>Matrix of weights for linear projection methods</td>
</tr>
<tr>
<td>( \theta )</td>
<td>Transfer rate parameter</td>
</tr>
<tr>
<td>( T )</td>
<td>Number of iterations</td>
</tr>
<tr>
<td>( \lambda )</td>
<td>Gaussian Mixture Model (GMM) parameters with ( K ) components</td>
</tr>
<tr>
<td>( E^{src}[x_j, y_i], E^{trg}[x_j, y_i] )</td>
<td>Joint expected value of feature ( j ) and label ( y_i )</td>
</tr>
<tr>
<td>( D(p, q) )</td>
<td>Dissimilarity between two distributions</td>
</tr>
<tr>
<td>( \nabla_{b_x} \mathcal{L}(\lambda_{trg}</td>
<td>\mathbf{x}_{src}) )</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>TransGrad translation regulator</td>
</tr>
<tr>
<td>( \epsilon )</td>
<td>Transfer Learning, Inductive TL, Transductive TL</td>
</tr>
<tr>
<td>( \text{MMD} )</td>
<td>Maximum Mean Discrepancy</td>
</tr>
<tr>
<td>( \text{TransGrad} )</td>
<td>Sample-based transformation using gradients</td>
</tr>
<tr>
<td>( \text{TST} )</td>
<td>Class-based Translation and Scaling Transform</td>
</tr>
</tbody>
</table>

In this section we will provide a brief background on the feature extraction methods which has been utilised in this thesis. In summary, for Player Action (video) datasets we used HoG3D which is a three dimensional generalisation of SIFT or local histograms of oriented gradients (HOG). As for the image datasets, we utilised two types of feature extractions: Bag-of-SURF and Convolutional Neural Network features.

2.2.1 Histograms of Oriented Gradients in 3D Space-time Blocks (HOG3D)

In order to extract a feature vector \( \mathbf{x} \) to describe a video segment a range of methods are available in literature. We used HoG3D [89] descriptors for our task because it was among the top performing method according to Wang et al.’s survey [153].
HOG3D is a three dimensional generalisation of SIFT [108] or local histograms of oriented gradients (HOG) [35]. It uses polyhedral structures for the quantisation of the 3-D spatio-temporal edge orientations to avoid the singularities in the use of polar coordinate systems (as advocated in [145]). Another advantage of HOG3D [89] is its computational efficiency due to the use of three-dimensional integral images.

Given a spatio-temporal block, HOG3D splits it into $M \times M \times N$ sub-regions ($M$ for spatial and $N$ for temporal splits). Within each region, this method counts how many 3D gradients are roughly aligned with each of the directions of a polyhedral structure. Figure 2.1 illustrates the spatio-temporal divisions of the HOG3D descriptors. In [89] the authors claim that the best performance in the validation set of the KTH [95] dataset was obtained with an icosahedron (i.e., 20 orientations), $M = 4$ and $N = 3$, giving a total of 960 dimensions. The temporal and spatial support of such descriptors were also optimised in [89], using the validation set of KTH.

In this work HOG3D is used as a descriptor for STS-based action matching (STS: Spatio-Temporal Shapes). But, in contrast to STS introduced in [65], HOG3D extracts information from within the foreground gray-scale patches, rather than only describing the outline of binary blobs. Since it describes 3D bounding boxes, it is also less affected by fragmentation problems in segmentation.
In the STS-based experiments of de Campos et al. [37], a single HOG3D descriptor is extracted for each detected actor at the time instance in which the action is classified. The extracted 960D vector is then passed directly to a classifier instead of using bags-of-visual-words. For problems in which the aim is to classify the activity in a video sequence rather than an instantaneous action, they use HOG3D at a number of temporal windows within a video sequence. The classification results are then combined using a voting scheme.

### 2.2.2 Bag of Visual Words

The bag-of-visual-words uses a simple assumption from natural language processing and information retrieval, and has been widely applied in computer vision. In general, there are three main steps for the model: (i) initially the local SURF [11] descriptors are obtained to describe the interest points; (ii) these descriptors are then quantised by K-means clustering to form a codebook; (iii) images are eventually represented as the K-dimensional histograms of the visual words [76]. The procedure includes two parts: learning and recognition.

For each image, interest points are sampled and the local descriptors are extracted. While any existing interest point detectors and descriptors can be used for this step, a common approach proposed in [11] is to use the SURF descriptor.

Inspired by SIFT [109], the SURF descriptors are claimed to be faster and more robust against different image transformations. It is based on an integer approximation of the determinant of the Hessian blob detector which can be computed extremely quickly with an integral image.

Secondly, these descriptors are quantised by K-means clustering to form a codebook. Finally, images can be represented as the K-dimensional histograms of the visual words.

### 2.2.3 Deep Convolutional Activation Feature (DeCAF)

In order to better evaluate our proposed approaches, we have also utilised the state-of-the-art CCN [41, 93, 27] features for representing our image data.
Figure 2.2: A typical processing pipeline for a bag-of-word category recognition system reproduced from [125]. Features are first extracted at key-points and then quantised to get a distribution (histogram) over the learned visual words (feature cluster centers). The feature distribution histogram is used to learn a decision surface using a classification algorithm such as support vector machines [150].

Natural images have the property of being stationary, meaning that the statistics of one part of the image are the same as any other part. This suggests that the features that we learn at one part of the image can also be applied to other parts of the image, and we can use the same type of feature at all locations.

More precisely, having learned features over small (i.e. 8 × 8) patches sampled randomly from larger image, we can apply this learned 8 × 8 feature detector anywhere in the image. Specifically, we can take the learned 8 × 8 features and convolve them with the larger image, thus obtaining a different feature activation value at each location in the image.

A typical Convolutional Neural Network (CNN) architecture is shown in Figure 2.3. In a fully-connected network, the output of each hidden activation unit $h_i$ is computed by multiplying the entire input $x$ by weights $W$ in that layer. However, in a CNN, each hidden activation output is computed by multiplying a small local input (i.e. $[x_1, x_2, x_3]$) against the weights $W$. The weights $W$ are then shared across the entire input space. After computing the output of the hidden units, a max-pooling layer helps to remove variability in location, scale and orientation of image content. Specifically, each max-pooling unit receives activations from a bank of convolutional bands, and outputs the maximum of the activations from these bands.

The input and output of each step in a CNN architecture are sets of arrays called feature maps.
2.2. Feature Extraction Methods for Image and Video in Computer Vision

Figure 2.3: Convolutional Neural network structure (©Torch7 documentation [27]): A CNN is a trainable architecture composed of multiple convolution and pooling stages.

For example, if the input is a colour image, each feature map would be a 2D array containing a colour channel of the input image (i.e. for a video or volumetric image, it would be a 3D array). At the output, each feature map represents a particular feature extracted at all locations on the input. Each layer in the network is composed of three steps: a filter bank step, a non-linearity step, and a feature pooling step. A typical CNN is composed of one, two or more such 3-step layers, followed by a classification module. The steps are further described for the case of image recognition:

- **Filter bank step**: the input is a 2D array feature map denoted as \( x_i \). The output is also a 2D array feature map, \( y_j \). A trainable filter (kernel) \( k_{ij} \) in the filter bank connects input feature map \( x_i \) to output feature map \( y_j \). The module computes \( y_{ij} = b_j + W_{k_{ij}} \ast x_i \) where \( \ast \) is the 2D discrete convolution operator and \( b_j \) is a trainable bias parameter. Each filter detects a particular feature at every location on the input. Hence spatially translating the input of a feature detection layer will translate the output but leave it otherwise unchanged.

- **Non-linearity step**: In traditional CNN this simply consists in a pointwise hyperbolic tangent (\( tanh() \)) sigmoid function applied to each site \( (x_{ijk}) \). However, recent implementations have used more sophisticated non-linearities. A useful one for natural image
recognition is the rectified sigmoid Rabs: $|g_i \cdot \tanh()|$ where $g_i$ is a trainable gain parameter. The rectified sigmoid is sometimes followed by a subtractive and divisive local normalisation, which enforces local competition between adjacent features in a feature map, and between features at the same spatial location.

- **Feature pooling step:** This step treats each feature map separately. In its simplest instance, it computes the average values over a neighborhood in each feature map. Recent work [154, 29] has shown that more selective poolings, based on the LP-norm, tend to work best with a $2 \times 2$ subsampling or alternatively by max pooling instead of averaging. The later approach forces the network to capture the most useful local features that are produced by the convolutional layers. The neighborhoods are stepped by a stride larger than one (but smaller than or equal the pooling neighborhood). This results in a reduced-resolution output feature map which is robust to small variations in the location of features in the previous layer.

The output of this 3-step image alterations is known as the feature map of the $l^{th}$ layer in the network. According to the granularity level of interest in terms of the substructures within an image, the output of different stages of the network can be selected as the image descriptors. In our experiments we confined ourself to using the output of the $6^{th}$ network layer.

### 2.3 Feature Selection

Feature selection methods evaluate an optimality criterion for combinations of $f$ variables and select that combination for which this criterion is maximised. The goal is to disregard those variables that do not contribute to class separability to speed up the final training process. In our experiments, we have used a range of dimensionality reduction algorithms to enhance our final training. In this section, we will briefly describe these techniques.
2.3. Feature Selection

2.3.1 Principal Component Analysis

Principal component analysis (PCA) tries to derive new variables that is a linear combination of the original ones and are uncorrelated. The most common definition of PCA, by Hotelling [74], is that, for a given set of data vectors $x_i, i \in 1, ..., n$ the $k$ principal axes are those orthonormal axes onto which the variance retained under projection is maximal. Geometrically, PCA can be thought of as a rotation of the axes of the original coordinate system to this new set of orthogonal axes that are ordered in terms of the amount of variation of the original data they account for [155].

In order to capture as much of the variability as possible, let us choose the first principal component, denoted by $U_1$ [60]. Suppose that all centred observations are stacked into the columns of an $f \times n$ matrix $X^\top$, where each column corresponds to an $f$-dimensional observation and there are $n$ observations. Under this notation principal components can be described as linear projections of $X$ defined by weight vector $w_i$.

We choose $W$ to maximise $W^\top \Sigma W$ while constraining $W$ to have unit length where $\Sigma$ is the $f \times f$ sample covariance matrix of $X$ observations.

$$\max W^\top \Sigma W \quad \text{subject to } W^\top W = 1 \quad (2.1)$$

The solution can be expressed as the singular value decomposition (SVD) of the covariance matrix, $\Sigma$:

$$\Sigma = USV^\top \quad (2.2)$$

where matrix $S$ contains the eigenvalues of the covariance matrix of the observations on its orthogonal and the columns of $U$ in the SVD contain the eigenvectors of $\Sigma$. The training data can be reconstructed as:

$$\hat{X} = UU^\top X \quad (2.3)$$

By taking the first $f_1 : f_j$ eigenvectors of $U$ the data can be projected into a lower dimensional
space where the data variance is maximised. A distribution of points drawn from a bivariate Gaussian and centered on the origin of x-axis and y-axis is illustrated in Figure 2.4.

![Figure 2.4: PCA projection defines a rotation such that the new axes (x’ and y’ corresponding to $U_1$ and $U_2$ eigenvectors respectively) are aligned along the directions of maximal variance (the principal components) with zero covariance. This is equivalent to minimizing the square of the perpendicular distances between the points and the principal components.](image)

Note that PCA does not take class labels into account. In the next section we will introduce the Linear Discriminant Analysis (LDA) method which does it.

### 2.3.2 Linear Discriminant Analysis

Linear discriminant analysis (LDA), also known as Fisher’s linear discriminant, tries to find a linear combination of features which separates two or more classes of objects or events.

From geometrical point of view the Fisher strategy is a projection of a high dimensional data
2.3. Feature Selection

X into a low dimensional space $\mathbb{R}^{C-1}$. In the case of $C$-class problems the criterion is:

$$J_F(W) = \frac{W^\top S_B W}{W^\top S_w W}$$

(2.4)

where $S_B$ is the between classes scatter (covariance) matrix and $S_w$ is the within class covariance matrix which are given by:

$$S_B = \sum_{c=1}^{C} \frac{n_i}{n} (\mu_i - \mu)(\mu_i - \mu)^\top,$$

(2.5)

$$S_w = \sum_{i=1}^{C} \frac{n_i}{n} \Sigma_i$$

(2.6)

where $c = 1, \ldots, C$ and, $\mu_i$ and $\Sigma_i$ are the sample means and covariance matrices of each class and $\mu$ is the mean of all samples ignoring the class labels. The transformation matrix $W$ can be found by solving the eigenvalues problem:

$$S_w^{-1} S_B W = \Lambda W$$

(2.7)

This $W$ is a matrix of eigenvectors of $S_w^{-1} S_B$ and $\Lambda$ is a diagonal matrix of eigenvalues. This means that when $W_i$ is an eigenvector of $S_w^{-1} S_B$, the separability will be equal to the corresponding eigenvalue. Since the rank of $S_B$ is at most $C - 1$, then the eigenvectors associated with non-zero eigenvalues identify the vector subspace containing the data.

In summary, the LDA dimensionality reduction can be achieved by:

- Compute the $f$-dimensional mean vectors for the different classes from the dataset.
- Compute the scatter matrices (between-class and within-class scatter matrix).
- Compute the eigenvectors and corresponding eigenvalues for the scatter matrices.
- Sort the eigenvectors by decreasing eigenvalues and choose $k$ eigenvectors with the largest eigenvalues to form a $f \times k$-dimensional matrix $W$ (where every column rep-
represents an eigenvector).

- Use this $f \times k$ eigenvector matrix to transform the samples onto the new subspace. This can be summarised by the mathematical equation: $y = W^T x$ (where $x$ is a $f \times 1$-dimensional vector representing one sample, and $y$ is the transformed $f' \times 1$-dimensional sample in the new subspace).

2.4 Classification

The final step in supervised machine learning is training a classifier. Given a sample $x$ and a class label $y$, the standard statistical classification task is to assign a probability, $p(y|x)$ or alternatively a label, $\hat{y}$ to the event of $x$ where $y$ denotes class membership. In the binary classification case the labels are $Y \in \{0, 1\}$. Typically, each example $x_i$ is represented as an $f$-dimensional vector of features. Below, we will brief the classification algorithms which we utilised in this thesis.

2.4.1 Nearest Neighbor

Among the various methods of supervised statistical pattern recognition, the Nearest Neighbour classifier achieves consistently high performance, without requiring any a priori assumptions about the distributions from which the training examples are drawn. Given a training set of both positive and negative samples, a new test sample, $x$, is classified by searching for the nearest training point and assigning it the same label.

The k-NN classifier extends this idea by taking the $k$ nearest points and assigning the label of the majority. It is common to select $k$ small and odd to break ties (typically 1, 3 or 5). Larger $k$ values help to reduce the effect of outliers within the training data set, and the choice of $k$ is often performed through cross-validation.

As the size of training data set approaches infinity, the one nearest neighbour classifier guarantees an error rate of no worse than twice the Bayes error rate (the minimum achievable error
2.4. Classification

rate given the distribution of the data) [141].

2.4.2 Logistic Regression

Logistic regression is a probabilistic, linear classifier. It is parametrised by a weight vector $w$, and a bias scaler $b$.

Given a data set $\{y_i, x^i_1, \ldots, x^i_f\}_{i=1}^n$ of $n$ statistical samples, a linear regression model assumes that the relationship between the dependent variable $y_i$ and the $f$-dimensional vector of regressors $x_i$ is linear. This relationship is modelled through a disturbance term or error variable $b$ an unobserved random variable that adds noise to the linear relationship between the dependent variable and regressors. Thus the model takes the form

$$y_i = w_1 x^i_1 + \cdots + w_f x^i_f + b = (x_i)^\top w + b, \quad i = 1, \ldots, n, \quad (2.8)$$

A large number of solutions have been developed for estimating the weight parameters out of which the Ordinary Least Squares (OLS) is the simplest and thus the most common estimator. It minimises the sum of squared residuals, and leads to a closed-form expression for the estimated value of the unknown parameter $w$:

$$\hat{w} = (X^\top X)^{-1} X^\top y = \left(\frac{1}{n} \sum_n x_i x_i^\top\right)^{-1} \left(\frac{1}{n} \sum_n x_i y_i\right) \quad (2.9)$$

The estimator is unbiased and consistent if the errors have finite variance and are uncorrelated with the regressors [98]. Alternatively maximum-likelihood estimation can be performed when the distribution of the error terms is known to belong to a certain parametric family of probability distributions [124]. When this distribution is a normal distribution with zero mean and variance $\sigma$, the resulting estimate is identical to the OLS estimate.

As for the logistic counterpart, the outputs of this linear layer are then fed to a softmax layer, which produces a properly normalised probability distribution. Mathematically for a binary
classification task, it can be defined as:

\[ P(y_i = 1|x_i, w, b) = \frac{1}{1 + e^{-(x_i^T w + b)}} \]  

(2.10)

The final prediction is typically done by choosing the label which maximises this distribution:

\[ y_i = \arg \max_c P(y_i = c|x_i, w, b) \]  

(2.11)

### 2.4.3 Support Vector Machines (SVM)

Support Vector Machines (SVM) algorithm invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis are supervised learning models with associated learning algorithms that analyse data and recognise patterns. Given a set of \( n \) training points of the form:

\[ \{(x_i, y_i) | x_i \in \mathbb{R}^f, y_i \in \{-1, 1\}\} \]  

(2.12)

where the \( y_i \) is either 1 or \(-1\), indicating the class to which the point \( x_i \) belongs. Each \( x_i \) is an \( f \)-dimensional real vector and the SVM classifier aims to find the maximum-margin hyperplane which separates the points having \( y_i = 1 \) from those of class \( y_i = -1 \). Any hyperplane can be written as the set of points \( x \) satisfying

\[ w \cdot x - b = 0, \]  

(2.13)

where \( \cdot \) denotes the dot product and \( w \) the (not necessarily normalised) normal vector to the hyperplane. The parameter \( \frac{b}{\|w\|} \) determines the offset of the hyperplane from the origin along the normal vector \( w \).

If the training data are linearly separable, we can select two hyperplanes in a way that they separate the data and there are no points between them, and then try to maximise their distance.
2.4. Classification

The region bounded by them is called the margin. These hyperplanes can be described by the equations

\[ w \cdot x - b = 1 \quad \text{for } y_i = 1 \]
\[ w \cdot x - b = -1 \quad \text{for } y_i = -1, \]

By using geometry, we find the distance between these two hyperplanes as \( \frac{2}{\|w\|} \). In order to maximise this distance we need to minimise \( \|w\| \). As we also have to prevent data points from falling into the margin, we add the following constraint: for each \( i \) either

\[ w \cdot x_i - b \geq 1 \quad \text{for } y_i = 1 \]
\[ w \cdot x_i - b \leq -1 \quad \text{for } y_i = -1, \]

This can be rewritten as:

\[ y_i(w \cdot x_i - b) \geq 1, \quad \forall i, 1 \leq i \leq n \]  \hspace{1cm} (2.14)

We can put this together to get the optimisation problem:

\[ \min_{w,b} \|w\|^2, \quad \text{subject to } y_i(w \cdot x_i - b) \geq 1. \]  \hspace{1cm} (2.15)

This is a quadratic optimisation problem which assumes that the data is perfectly separable. To deal with the non-separable case one can rewrite the optimisation problem as:

\[ \min_{w,b} \|w\|^2 + C \sum_{i=1}^{n} \xi_i, \quad \text{subject to } y_i(w \cdot x_i - b) \geq 1 - \xi_i. \]  \hspace{1cm} (2.16)

where \( \xi_i \geq 0 \) and \( C \) is known as the margin parameter. The second term in eq. 2.16 can be formulated as a training error function known as the Hinge Loss:

\[ \xi_i = H_1[y_i, (w \cdot x_i - b)] \]  \hspace{1cm} (2.17)

where \( H_1(y, \hat{y}) = \max(0, 1 - y\hat{y}) \). The optimisation still is quadratic and solvable with
Quadratic Programming (QP) techniques.

While linear separation models are not always complex enough to describe the data, the non-linear SVM maps data to a richer feature space including non-linear features, then constructs a hyperplane in that space. Formally, we can pre-process the data by mapping it via a non-linear function $\Phi$:

$$\mathbf{x} \mapsto \Phi(\mathbf{x})$$

(2.18)

and then learn the map from $\Phi(\mathbf{x})$ to $y$ subject to:

$$y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) - b) \geq 1 - \xi_i, \quad \forall i, 1 \leq i \leq n$$

(2.19)

The dimensionality of $\Phi(\mathbf{x})$ can be very large, making $\mathbf{w}$ hard to represent explicitly in memory, and hard for the QP to solve. The representer theorem [87] shows that for SVM as a special case:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x})$$

(2.20)

for some variables $\alpha$. Instead of optimising $\mathbf{w}$ directly we can thus optimise $\alpha$. The decision rule is now:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \Phi^T(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) - b$$

(2.21)

where $K(\mathbf{x}_i, \mathbf{x}) = \Phi^T(\mathbf{x}_i) \cdot \Phi(\mathbf{x})$ is known as the kernel function and the whole solution is coined as kernel trick.

The effectiveness of SVM depends on the selection of kernel type, the kernel parameters, and soft margin parameter $C$. A common choice is a Gaussian kernel also known by Radial Basis function (RBF), which has a single parameter $\gamma$. The Gaussian kernel for $f$-dimensional feature vector $\mathbf{x}$ can be computed by:

$$G(\mathbf{x}, \gamma) = \frac{1}{(\sqrt{2\pi}\gamma)^f} e^{-\frac{|\mathbf{x}|^2}{2\gamma^2}}$$

(2.22)
The best combination of $C$ and $\gamma$ is often selected by a grid search using a validation set. Typically, each combination of parameter choices is checked using cross validation, and the parameters with best cross-validation accuracy are picked. The final model, which is used for testing and for classifying new data, is then trained on the whole training set using the selected parameters [75].

### 2.4.4 Gaussian Classifier

Suppose $f_c(x)$ is the class conditional density of $X$ in class $c$, and let $p(y = c)$ be the prior probability of class $c$, with $\sum_{c=1}^{C} p(y = c) = 1$. A simple application of Bayes’ theorem gives us:

$$P(y = c|x) = \frac{f_c(x)p(y = c)}{\sum_{c=1}^{C} f_c(x)p(y = c)}$$

(2.23)

where the $P(y = c|x)$ is the posterior probability of class $c$ given $x$. Notice that having quantities $f_c(x)$ is almost equivalent to having the $p(y = c|x)$ provided by Bayes’ rule. Assuming that the data are normally distributed, with equal covariance matrices in each class where $\Sigma_c = \Sigma$ and denoting class mean vectors by $\mu_c$, one can model each class conditional density as a multivariate Gaussian:

$$f_c(x) = \frac{1}{(2\pi)^{l/2}|\Sigma_c|^{1/2}} \exp\{-\frac{1}{2}(x - \mu_c)^{T}\Sigma_c^{-1}(x - \mu_c)\}$$

(2.24)

The discrimination rule then can be stated as assigning test sample $x$ to class $c$ if $P(y = c_i|x) > P(y = c_j|x)$. Taking the log, the class label prediction can be based on:

$$\hat{y}_i = \arg \max_c \log(P(y = c|x_i)) = \arg \max_c \left[-\log((2\pi)^{l/2}|\Sigma_c|^{-1/2}) - \frac{1}{2}(x - \mu_c)^{T}\Sigma_c^{-1}(x - \mu_c) + \log(p(y = c))\right]$$

(2.25)
Assuming $\Sigma_c = \Sigma, \forall c$ this can be rewritten as:

$$
\hat{y}_i = \arg \max_c [\log(p(y = c)) - \frac{1}{2} \mu_c^T \Sigma^{-1} \mu_c + x^T \Sigma^{-1} \mu_c]
$$

The second term on the right side of eq. 2.25 is the Mahalanobis distance, which measures the dissimilarity between groups, assuming Gaussian distribution.

### 2.4.5 Kernel Discriminant Analysis

Linear discriminant analysis (LDA) of Section 2.3.2 is a traditional statistical method which has proved successful on classification problems [54]. The projection vectors are commonly obtained by maximising the between class variance and simultaneously minimising the within class variance [22] as previously detailed in Section 2.3.2. Since the classical LDA is a linear method and fails for non-linear problems, the non-linear extension of LDA through kernel trick has been proposed. As explained in 2.4.3, the main idea of the kernel-based methods is to map the input data into a feature space through a non-linear mapping, where the inner products in the feature space can be computed by a kernel function without knowing the non-linear mapping exactly [143]. In order to deal with non-linear decision boundaries, a kernelised version of LDA (KDA) has been proposed in [115]. Following the work of de Campos et al. [37], we adopted a spectral regression based implementation of KDA [22], which they claim to be much more efficient than both standard KDA implementations and SVM [14]. In Section 4.4.2 we discuss the kernel functions used in our experiments.

Let $X = X_1 \cup X_2 = \{x_1, \cdots, x_{n_1}, x_{n_1+1}, \cdots, x_{n_1+n_2}\}$ be a set of training vectors of two classes, where $n_1$ and $n_2$ denote the number of samples in the two classes $X_1$ and $X_2$. Suppose that the input space $\mathcal{X}$ is transformed into a Hilbert space $H$ by a non-linear mapping function $\Phi: \mathcal{X} \rightarrow \Phi(\mathcal{X})$. Hence $S_B$ and $S_w$ in the $H$ space can be defined by:

$$
S_B = H_B H_B^T,
$$

(2.27)
\[ H_B = [\sqrt{m_1}(E(\Phi_1) - E(\Phi)), \sqrt{m_2}(E(\Phi_2) - E(\Phi))] \]

\[ S_w = H_wH_w^T, \quad (2.28) \]

\[ H_W = [\phi(x_1) - E(\Phi_1), \ldots, \phi(x_{n_1}) - E(\Phi_1), \phi(x_{n_1+1}) - E(\Phi_2), \ldots, \phi(x_{n_1+n_2}) - E(\Phi_2)] \]

where \( E(\phi_1) \), \( E(\phi_2) \) and \( E(\phi) \) denote the expected value of class 1, the mean of class 2 and the mean of entire data, respectively.

\( W \) can be obtained by solving a general eigenvalue problem

\[ S_B W = S_w \lambda W \quad (2.29) \]

The eigenvectors are linear combinations of \( H \) elements, and all solutions \( W \) with non-zero eigenvalues lie in the span of \( \{\phi(x_1), \ldots, \phi(x_{n_1+n_2})\} \).

## 2.5 Transfer Learning

Data mining and machine learning technologies have already achieved significant success in many knowledge engineering areas including classification, regression and clustering. However, many machine learning methods work well only under a common assumption that training and test data are drawn from the same feature space and the same distribution. When the distribution changes, most statistical models need to be rebuilt, using newly collected training data. In many real world applications, it is expensive or impossible to collect the needed training data and rebuild the models. It would be nice to reduce the need and effort to collect the training data. In such cases, knowledge transfer or transfer learning between task domains would be desirable [123]. The need for transfer learning may arise when the data can be easily outdated. In this case, the labelled data obtained in one time period may not follow the same distribution in a later time period.
In the paradigm of inductive learning, \((X_{\text{train}}, Y_{\text{train}})\) are known, while both \(X_{\text{test}}\) and \(Y_{\text{test}}\) are completely hidden during training time. In the case of semi-supervised inductive learning [6], the learner is also provided with auxiliary unlabelled data \(X_{\text{auxiliary}}\), that is not part of the test set. It has been noted that such auxiliary data typically helps to boost the performance of the classifier significantly. Another setting that is closely related to semi-supervised learning is transductive learning [6], in which \(X_{\text{test}}\) (but, importantly, not \(Y_{\text{test}}\), is known at training time. That is, the learning algorithm knows exactly on which examples it will be evaluated after training. This can be a great asset to the algorithm, allowing it to shape its decision function to match and exploit the properties seen in \(X_{\text{test}}\). One can think of transductive learning as a special case of semi-supervised learning in which \(X_{\text{auxiliary}} = X_{\text{test}}\) [6]. In the setting of transfer learning, however, we would like to apply our trained classifier to examples drawn from a distribution different from the one upon which it was trained. We therefore assume there are two different distributions, \(D_{\text{src}}\) and \(D_{\text{trg}}\), from which data may be drawn. Given this notation we can then precisely state the transfer learning problem as trying to assign labels \(Y_{\text{trg}}\) to test data \(X_{\text{trg}}\) drawn from \(D_{\text{trg}}\), given training data \((X_{\text{src}}^{\text{train}}, Y_{\text{src}}^{\text{train}})\) drawn from \(D_{\text{src}}\).

From the conceptual point of view, three main research issues can be identified in transfer learning: 1) what to transfer, 2) how to transfer, and 3) when to transfer.

"What to transfer" asks which part of the knowledge can be transferred across domains/tasks. Some knowledge is specific to individual domains/tasks, and some knowledge may be common between different domains/tasks such that they may help improve performance for the target domain/task [123]. Discovering what knowledge can be transferred, should affect the design of a learning algorithm which corresponds to "how to transfer". "When to transfer" considers under what conditions transfer should be done. In some situations, when the source and target domains are not related to each other, brute-force transfer may be unsuccessful. It may even degrade performance in the target domain, a situation which is often referred to as negative transfer.

Based on the taxonomy of transfer learning in [123], we summarise the relationship between the traditional machine learning and various transfer learning settings in Figure 2.5, where
transfer learning is categorised under three sub-settings, inductive transfer learning (ITL), transductive transfer learning (TTL), and unsupervised transfer learning (UTL), based on different situations regarding the source and target domains and tasks.

![Taxonomy of transfer learning methods from [123], showing the relation with traditional machine learning problems. Here we are highlighting the ones that we are more interested in this thesis.](image)

**2.5.1 Inductive Transfer Learning**

In the inductive transfer learning setting, the target task is different from the source task, no matter whether the source and target domains are the same or not. In this case, some labelled data in the target domain is required to induce an objective predictive model, \( p(y|x) \), to be used in the target domain. In addition, according to different situations of labelled and unlabelled data in the source domain, we can further categorise the inductive transfer learning setting into two cases: (a) A lot of labelled data in the source domain is available. In this case, the inductive transfer learning setting is similar to the multi-task learning setting. However, the inductive transfer learning setting only aims at achieving high performance in the target task.
by transferring knowledge from the source task while multi-task learning tries to learn the target and source task simultaneously. (b) No labelled data in the source domain is available. In this case, the inductive transfer learning setting is similar to the self-taught learning setting [123]. In the self-taught learning setting, the label spaces between the source and target domains may be different, which implies the side information of the source domain cannot be used directly. Thus, it is similar to the inductive transfer learning setting where the labelled data in the source domain is unavailable.

Based on what needs to be transferred through the transfer learning method, inductive transfer learning can be sub-categorised into four cases described in the next section.

**Transferring the Knowledge of Instances**

Based on available partial similarity between the source and target domain samples, although the entire source domain data cannot be reused for target domain, there are specific parts of it which are suitable for training in the target domain. TraAdaBoost suggested by Dai et al. [34] assumes that the source and target feature spaces are the same but their marginal distributions are different. It further assumes that due to this intrinsic differences some of the source data may be helpful in learning the target domain task but some of them may not and could even be harmful. The technique attempts to iteratively re-weight the source domain data to reduce the effect of harmful samples while encouraging the "good" source samples. TraAdaBoost trains the base classifier on the weighted source and target data while the error is only calculated on the target data in each iteration.

Jiang and Zhai [80] proposed a heuristic method to remove "misleading" training samples from the source domain based on the difference between the conditional probabilities $p(y^{\text{trg}}|x^{\text{trg}})$ and $p(y^{\text{src}}|x^{\text{src}})$. Liao et al. [106] proposed a new active learning method to select the unlabelled data in a target domain to be labelled with the help of the source domain data. An integrated source domain data and SVM framework for improving the classification performance was proposed by Wu and Dietterich [159] as an alternative.
2.5. Transfer Learning

In [97], the authors assume a set of source domains, for which they learned person re-identification models. To leverage the learned experience of these domains in a new target domain, a multi-kernel learning [8] strategy is used. Each source domain is providing a score indicating its confidence that a given source-target pair is a matching pair under the model of that domain.

**Transferring Knowledge of Feature Representation**

The feature representation transfer aims at finding an appropriate feature representation to minimise the divergence of the domains and hence reduce the classification or regression error. Based on the amount of available labelled data as source samples, supervised or unsupervised methods can be used to construct a feature representation.

In the presence of a large number of labelled data in the source domain, a supervised construction based on learning a low dimensional representation, shared across related tasks can be used. Supervised feature learning methods are similar to those used in multitask learning. The basic idea is to learn a low-dimensional representation that is shared across related tasks [123]. Rodner et al. [135] proposed to use random decision forests for transferring feature relevance. In their inductive transfer approach, in order to estimate the underlying feature relevance with a few labelled samples available in the target space, they first train the decision forest using all the labelled examples of both source and target sets. The next step is to count the times a specific feature is used for optimisation of a split node of each of trees in the forest (for more details on Random Forest refer to Section A.3). The idea is that the feature with the high occurrence is likely to be relevant for a given transfer task.

Lee et al. [102] proposed a convex optimisation algorithm for simultaneously learning meta-priors and feature weights from an ensemble of related prediction tasks where the meta-priors are transferable among different tasks. Jebara [79] proposed an SVM based feature selection method for multitask learning. Rückert and Kramer [139] designed a kernel-based approach to inductive transfer which aims at finding a suitable kernel for the target data.

In the absence of any labelled data, unsupervised learning methods are proposed. In [132],
Raina et al. proposed to apply sparse coding [101], which is an unsupervised feature construction method, for learning higher level features for transfer learning. The basic idea consists of two steps. In the first step, higher level basis vectors are learned on the source domain data by solving an optimisation problem which aims at minimising the cost of feature construction. In the second step, an optimisation algorithm is applied on target domain data to learn higher level features based on the source basis vectors. Finally, discriminative algorithms can be applied to train classification models for use in the target domain. One drawback of this model is that the so-called higher level basis vectors learned on the source data might not be suitable for use in the target domain. Recently, manifold learning methods have been adapted for transfer learning.

**Transferring the Knowledge of Parameters**

In this approach the assumption is that individual models for related tasks should share some parameters or prior distribution of hyper-parameters. So the aim is to adapt the previously trained model parameters in order to better fit the new domain. Lawrence and Platt [96] proposed an efficient algorithm known as MT-IVM which is based on Gaussian Process (GP) and tries to learn parameters of a GP over multiple tasks which share the same GP prior. Similarly, Bonilla et al. [17] proposed a multitask learning scheme which use a free-form covariance matrix over tasks to model intertask learning dependencies, where a GP prior is used to induce correlations between tasks.

Besides transferring the prior of GP models, some research proposed regularisation-based transfer of the SVM parameters. Evgeniou and Pontil [45] borrowed the idea of hierarchical Baysian framework [144] to SVMs for multitask learning. The proposed methods assumed that the SVM weight parameter can be separated into two terms: one being a common term over tasks, \( w_c \) and the others being task-specific terms, \( w^{src} \), \( w^{trg} \).
2.5. Transfer Learning

Transferring Relational Knowledge

Unlike other three contexts, relational-knowledge transfer deals with relational domains where data are non-i.i.d. (independent and identically distributed) [123] and can be represented by multiple relations, such as network and social network data. Statistical relational learning techniques are proposed for solving these types of problems where the most common algorithms are based on the Markov Logic Networks representations [113, 114, 36, 134].

More details on the available algorithms using this criteria can be found in [123], [92].

2.5.2 Transductive Transfer Learning

In the transductive transfer learning setting, the source and target tasks are the same, while the source and target domains are different. In this situation, no labelled data in the target domain is available while we have labelled data in the source domain [123]. In addition, according to different situations between the source and target domains, we can further categorise the transductive transfer learning setting into two cases. (a) The feature spaces between the source and target domains are different, $\mathcal{X}^{src} \neq \mathcal{X}^{trg}$. (b) The feature spaces between domains are the same, $\mathcal{X}^{src} = \mathcal{X}^{trg}$, but the marginal probability distributions of the input data are different, $P(x^{src}) \neq P(x^{trg})$. The latter case of the transductive transfer learning setting is related to the problem of domain adaptation for knowledge transfer in text classification [123] and sample selection bias [68] or covariate shift [165], whose assumptions are similar. Transductive transfer can be undertaken via the same four approaches as discussed for the Inductive transfer. In this thesis, we focus on transductive transfer learning. Different solutions have been proposed in this area.

The idea of translated learning across different feature spaces was introduced by Dai et al. in [33], which tries to translate all the training data into a target feature space where learning can be performed within a single feature space. Assuming the set of features in source and target domains are represented by $X^{src}$ and $X^{trg}$ respectively, the aim is to link the two feature
spaces with the construction of a feature translator $p(X_{\text{trg}}|X_{\text{src}})$. Although this approach is not precisely related to our main problem, since two different feature extraction algorithms are defined for our source data set, this translation concept can be used to improve the result of classification in the target domain.

The dimensionality reduction method proposed by Pan et al. [120] which finds a latent space that minimises the distance between the distributions of the data in different domains is somehow similar to the Weinberger et al.’s [156] method but has been formulated for the dissimilarity between the marginal distributions of source and target data. The dimensionality reduction introduced in [156] is called Maximum Variance Unfolding (MVU) which is motivated by designing kernels for Kernel Principal Component Analysis (KPCA) from the data itself. This method [156] extracts a low dimensional representation of the data by maximising the variances of the embedding while preserving the local distances between neighbouring observations.

As for estimating the distance between different distributions a non-parametric method called Maximum Mean Discrepancy (MMD) has been defined which assumes that the distance between the distribution of two samples is equivalent to the distance between the means of the two samples mapped into a reproduced kernel Hilbert space. The later assumption in this work does not seem to accurately generalise to all types of feature spaces. Later work published by the same authors [122] is also based on the same assumption while trying to only define a more efficient model for kernel learning.

A method of structural correspondence learning (SCL) was proposed in [16] to be used for part of speech tagging. It attempts to identify some pivotal features between the source and target domains where these pivotal features are the ones that occur more frequently in the two domains and behave similarly in both. After defining a set of pivotal features for unlabelled data in both domains, they are used to learn a mapping from the original feature spaces of both domains to a shared low-dimensional real-valued feature space, where a high inner product in this new space indicates a high degree of correspondence. The motivation is that if a good mapping is found, the classifier learned on the source domain will also be effective on the target domain. The feature vectors which are used for experiments in [16] are defined to be binary
features which is not the case for the feature spaces used in this thesis.

Knowledge transfer via a multiple model local structure mapping proposed in [57] is another solution for the transductive transfer learning problem. The rational concept behind this method is that classifiers can be built using different learning algorithms on the same source domain and hence due to the inductive bias of the specific learning technique or the distributional differences among the available different training domains, they can achieve different knowledge resulting in different advantages in learning. Thus, the idea is to combine the knowledge from these classification based models. For this combination of the base models, rather than model averaging by assigning a global weight to all the existing models, Gao et al. [57] use the principle of the risk minimisation to find a solution to assign per model and per example weights to maximise their combined accuracy on the new target domain. This method proposes a graph-based approach to approximate the optimal model weights where each local weight is computed by the first mapping and then to measure the similarity between the model and the test domain local structure around the test example. This similarity is measured by comparing neighbourhood graphs, and quantified in the weight assignment equation. Intuitively, it favours classifiers whose mapped local structure is similar to the local structure around the test example and if none of the mapped local structures is similar to the original local structure in the target domain, the predicted label will be obtained by voting among its neighbours inside the same local structure of the test set. The introduced model has showed a significant improvement in performance when tested on three real tasks: Spam filtering, text categorisation and network intrusion detection.

An Audio signal processing application of domain adaptation is the acoustic environment adaptation in speech recognition. In speech recognition and audio processing, the problem of adaptation to new acoustic environments relate to adaptation of \( p(X) \) [158]. Hence, the requirement of a small amount of data from the new environment to adapt a pre-existing model is a matter of importance. The method introduced in [56] combines estimates of convolutional and additive noise to compensate an HMM (Hidden Markov Model) model, trained on clean speech and a maximum likelihood linear regression (MLLR) [103, 104] which estimates a set of linear
Chapter 2. Background

transformations for the Gaussian mean and covariances parameters. More detailed explanation can be found in [56]. Maximum a Posteriori (MAP) adaptation used for play field segmentation [10] is another approach to re-estimate the GMM parameters taking advantage of prior knowledge about the model parameter distribution. One obvious drawback to MAP adaptation compared to MLLR is its requirement of more adaptation data to be effective. In fact the two adaptation processes can be combined to improve performance still further, by using the MLLR transformed means as the priors for MAP adaptation. Methods for domain adaptation can be formulated to apply to our problem.

Kalinli et al. [83] also proposed to use the maximum likelihood linear regression (MLLR) to estimate a set of linear transformations for the Gaussian parameters of the HMMs [103]. In [83], a vector Taylor series approach for HMM adaptation was introduced for decoding noisy utterances at the test time. An intermediate step of noise adaptive model training is used that results in pseudo-clean model parameters.

In [136], Rodriguez et al. proposed an adaptation scheme for semi-continuous HMMs for unsupervised writer style adaptation in handwritten word spotting. The method exploits the fact that the semi-continuous hidden Markov model separates the word model parameters into (i) a codebook of shapes and (ii) a set of word-specific parameters. In [136], the authors employ this property to derive writer-specific word models by statistically adapting an initial universal codebook to each document. The proposed process is unsupervised and does not require the appearance of the keyword(s) in the searched document. An issue raised by the discussed approaches is that they work on the assumption that the observation data follows a probabilistic generative model.

Germain et al. in [59] proposed a modified version of the PAC-baysian approach for the domain adaptation problems. The DA-PAC-Baysian algorithm is based on the minimisation of a set of predefined bounds in order to find a suitable set of weights to assign to each hypothesis for building a majority vote. The authors seek a $\rho$-weighted majority vote that takes into account a trade-off between three quantities: (a) the complexity of the majority vote (measured by a Kullback-Leibler divergance) (b) its empirical text risk (measured by the $\rho$-average errors
2.5. Transfer Learning

on the source samples) and finally (c) the capacity of the majority vote to distinguish some structural difference between the source and target samples.

Recently Gopalan et al. [64, 63] used the Grassmann manifold for modeling the domain shift problem. The key idea was to synthesise intermediate domains using intermediate subspaces along a geodesic connecting the source and target domains and representing each object by concatenating its projections on these subspaces. The idea has further been developed in [129] by replacing the step that concatenates a few intermediate subspace projections by integrating the distance between the feature projections on all the intermediate subspaces along a geodesic.

One of the recent approaches for the domain adaptation task is combining ensembles of classifiers and clusterers to generate a more consolidated classifier [1]. In this method the ensemble of classifiers is used to estimate initial class probabilities for all the test samples. In the next step these probability distributions are refined with the help of a cluster ensemble which provides supplementary constraints for classifying the test objects with the rational that similar objects are more likely to share the same class label. In order to evaluate the similarities between the objects a similarity matrix is defined, where each entry responds to relative co-occurrence of two objects in the same cluster [58]. The results of the clusterers and classifiers are later used to obtain an optimised estimate of the class label for each sample.

In activity recognition, Calatroni et al. [23] showed the feasibility of transfer learning in Body Sensor Network (BSN) and used a transferring approach to evaluate the classification performance of transferring to a newly deployed sensor in the BSN. The transfer approach was based on a Teacher-Learner paradigm where all the pairs of nodes were considered as possible teachers and learners. The direct transferring of the classifiers in the nodes was compared to a system-supervised approach where the teacher node provides labels to the learner and a new classifier was thus trained in the node. Good results were obtained using the system-supervised approach specially when the transferring is done between nodes located in similar body part. Distance-based classifiers like Nearest Centroid Classifier, k-NN and SVM were used.

Following this trend, Casale et al. [24] considered the direct transfer of classifiers between
the nodes rather than training each node individually. In order to avoid this training, they use the collaborative training strategy and ensemble methods that provide transformations of the feature space that are potentially beneficial in the classification of activities using nodes placed at different positions.

Blanke and Schiele [15] applied transfer learning methodologies for the composition of complex activities from a time ordered sequence of events. In their work, the authors transferred the activity events that were shared between similar composite activities minimising the training effort for new events. The results obtained show that their approach achieves good performance in the recognition of composite activities, also when different application domains are considered. Although the methodology does not use a direct transfer approach, the work shows that transfer learning is feasible when a level of abstraction is considered in the learning mechanism. This consideration is also reported in van Kasteren et al. [152] where a level of abstraction is used to map features with the aim of transferring activity recognition capabilities between different home scenarios.

Another popular method for Transductive learning is Transductive SVM [82] and its extended version, domain adapted SVM [21], simultaneously learn a decision boundary and maximise the margin in the presence of unlabelled patterns, without requiring density estimation.

In [26], Chu et al. proposed to search for an instance based re-weighting matrix applied to the source samples. The weights are based on the similarity between the source and target distributions using the Kernel Mean Matching algorithm. This method iteratively updates an SVM classifier using transformed source instances for training until convergence.

In [61], Gong et al. proposed the geodesic flow kernel method which models domain shift by integrating an infinite number of subspaces that characterise changes in geometric and statistical properties from source to target domain.

In [53], Fu et al. formalised a general framework for multi-label zero-shot learning. They proposed a multi-output deep regression model which projects an image into a semantic word space, which explicitly exploits the correlations in the intermediate semantic layer of word
2.6 Summary

This chapter reviewed the theoretical concepts, technical ideas and background knowledge on feature extraction, classification and transfer learning. The mathematical contents described in this chapter are referenced throughout the rest of the thesis. More detailed literature reviews are included at the related work sections of the technical chapters of this thesis where we further discuss how this literature survey motivates our proposed approaches for TTL.

vectors. Their proposed zero-shot learning algorithm for multi-label data then exploits the unique compositionality property of semantic word vector representations. They further introduce a transductive zero-shot learning strategy to enable the regression model learned from seen classes to generalise well to unseen classes. Although the deep learning method have been claimed to neglect the need for domain adaptation, in this work we will show that the domain adaptation algorithms can enhance the classification performance when are applied to deep neural network extracted features.

More detailed references on TTL are discussed in relation with the mythologies which are introduced in this thesis (Sections 4.2, 5.1 and A.1).

2.5.3 Unsupervised Transfer Learning

Finally, in the unsupervised transfer learning setting the target task is different from yet related to the source task but it focuses on solving unsupervised learning tasks in the target domain, such as clustering, dimensionality reduction, and density estimation [123]. In this case, there is no labelled data available for either source or target domains in training.

We will not discuss these methods further because our interest is on transductive transfer learning (TTL).

2.6 Summary

This chapter reviewed the theoretical concepts, technical ideas and background knowledge on feature extraction, classification and transfer learning. The mathematical contents described in this chapter are referenced throughout the rest of the thesis. More detailed literature reviews are included at the related work sections of the technical chapters of this thesis where we further discuss how this literature survey motivates our proposed approaches for TTL.
Chapter 3

Datasets
Image and video classification systems should perform well on test data representing realistic scenarios in terms of content, variation in model database size, sensor used for data acquisition, viewing conditions, illumination and background. Therefore the evaluation of a transductive transfer algorithm with a wide range of datasets of different characteristics is a necessity. This chapter introduces the databases used for evaluating, characterising and benchmarking the object and action recognition methods developed and investigated in this thesis. The chapter is organised as follows. In the next section, the player-actions database is described which was initially generated and labelled using the tennis video annotation system of [91]. In the second part of this chapter we describe a range of object and digit recognition databases: in Sections 3.2 and 3.3 three digit databases are introduced namely; Chars74k, MNIST and USPS and in Sections 3.4 and 3.5 public object recognition public databases used in our experiments are presented.

3.1 ACASVA: Player Actions Database

The player action datasets of this section were first introduced in de Campos et al. [37] for a module of the ACASVA project (Adaptive Cognition for Automated Sports Video Annotation). They used the action features obtained by a method described as space-time-shape (STS) for describing the player actions. They further described a purpose-built ground truth annotation system for the player action datasets designed for annotating sports videos (tennis, badminton) by labelling key events such as Serve, Hit and etc.

3.1.1 ACASVA System

Figure 3.1 shows a detailed block diagram for the ACASVA system of [91]. A simplified version of the system diagram is shown in Figure 3.2. Each block in it consists of several “modules” of the system providing specific functionality (which can be found in Figure 3.1, where each rectangular block represents one module of the system). It should also be noted that
the novel contribution of this thesis to the system can be summarised in the “event detection” block of Figure 3.2 for which we introduce novel methods to make the system capable of annotating players actions of other similar sports as well. The development and implementation of the algorithms in other blocks, as well as a memory architecture that enables the modules to communicate with each other, and a graphical interface for the system, are parts of a pre-existing work carried out in this context (see [137, 119, 91, 161, 89] for more details).

Pre-Processing Tennis videos from various sources used in our experiments are generally recorded with interlaced cameras, thus in the “pre-processing” block of Figure 3.2, image frames are first de-interlaced into fields. Fields are used, rather than frames, in order to alleviate the effects of temporal aliasing. This is particularly important for the ball tracker. When the tennis ball moves fast, the ball is alternately present and absent on successive frame lines, hence the need to operate on fields rather than frames. For simplicity, we will use, the word “frame” to refer to “field” in the rest of this chapter.

After de-interlacing, the geometric distortion of camera lens is corrected. The camera position on the court is assumed to be fixed, and the global transformation between frames is assumed to be a homography [69]. The homography is found by: tracking corners through the sequence; applying RANSAC to the corners to find a robust estimate of the homography, and finally applying a Levenberg-Marquardt optimiser [110] for a finer (non-linear) optimisation of the homography.

Shot Analysis A broadcast tennis video is composed of shots, such as game-play, close-up, crowd, commercial. An illustrative example of the composition of a tennis video is shown in Figure 3.3. Example frames from different types of shots can be found in Figure 3.4. In the “shot analysis” block of Figure 3.2, shot boundaries are detected using colour histogram intersection between adjacent frames; shots are then classified into appropriate types using a combination of color histogram mode and corner point continuity. For our purposes, some shots are incorrectly classified as “play”. This situation arises when “replays” are encountered.
Figure 3.1: A detailed diagram of the tennis video analysis system.
3.1. ACASVA: Player Actions Database

Figure 3.2: A simplified diagram of the tennis video analysis system (taken from [85]).

However, these detected false positives are eventually eliminated later on by the “projection” module (see Figure 3.1), which rejects the shot for which the system is unable to find the correct tennis court.

**Court Detection, Ball Tracking, and Player Tracking** For a play shot, the tennis court is detected through a combination of an edge detector and Hough transform [85]. The players are tracked using a particle filter which facilitates the player actions detection (see [137, 119] for details).

**Event Detection** By examining the tennis ball trajectories, motion discontinuity points are detected. The work in this thesis contributed to the player action classification module, which operates on cropped video sequences around the detected players, at a temporal interval around the time when a motion discontinuity was observed at the ball trajectory (due to bounces or hits). Further details about this method are described in Section 3.1.2.

Detected motion discontinuity points are combined with player positions, player actions and court lines in the “event detection” module, to generate key events such as hit, bounce and net.
Finally, the generated key events are sent to a high level module, where the tennis rules are incorporated into a Hidden Markov Model (HMM). The HMM is used as a reasoning tool to generate the annotation, i.e. outcome of play, point awarded, etc. (see [91, 85] for details). HMMs are also employed for providing contextual prior for event detection.

As can be seen in the detailed system diagram of Figure 3.1, the system is composed of 23 modules. A memory architecture enables the modules to communicate with each other [91]. This system can carry out contextual reasoning at various levels of representation with the raw video data at the lowest level and its semantic annotation of increasing abstraction at higher levels.

A graphical interface is also implemented for this system (see Figure 3.5) that shows the output of various modules providing a match related information such as high-level annotation e.g. game scores (displayed in the left panel), player positions, ball trajectories, and shot related information (showing whether the visible shot is a close-up, advert or play). Also, temporally, as the game progresses, detected low-level features and the related key events are also displayed in the main window.
3.1. ACASVA: Player Actions Database

Figure 3.5: The GUI of the tennis video annotation system [85].
Table 3.1 shows all of the output labels from this system.

**Table 3.1** Summary of tennis events from [91]

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFR</td>
<td>Serve by Far player, Right side</td>
</tr>
<tr>
<td>SFL</td>
<td>Serve by Far player, Left side</td>
</tr>
<tr>
<td>SNR</td>
<td>Serve by Near player, Right side</td>
</tr>
<tr>
<td>SNL</td>
<td>Serve by Near player, Left side</td>
</tr>
<tr>
<td>BIF</td>
<td>Bounce Inside Far player’s half court</td>
</tr>
<tr>
<td>BOF</td>
<td>Bounce Outside Far player’s half court</td>
</tr>
<tr>
<td>BIN</td>
<td>Bounce Inside Near player’s half court</td>
</tr>
<tr>
<td>BON</td>
<td>Bounce Outside Near player’s half court</td>
</tr>
<tr>
<td>HF</td>
<td>Hit by Far player</td>
</tr>
<tr>
<td>HN</td>
<td>Hit by Near player</td>
</tr>
<tr>
<td>BIFSR</td>
<td>Bounce Inside Far player’s Serve area on the Right</td>
</tr>
<tr>
<td>BIFSL</td>
<td>Bounce Inside Far player’s Serve area on the Left</td>
</tr>
<tr>
<td>BOFS</td>
<td>Bounce Out of Far player’s Serve area</td>
</tr>
<tr>
<td>BINSR</td>
<td>Bounce Inside Near player’s Serve area on the Right</td>
</tr>
<tr>
<td>BINSL</td>
<td>Bounce Inside Near player’s Serve area on the Left</td>
</tr>
<tr>
<td>BONS</td>
<td>Bounce Out of Near player’s Serve area</td>
</tr>
<tr>
<td>NET</td>
<td>Bounce on NET</td>
</tr>
</tbody>
</table>

### 3.1.2 Player Action Dataset and Ground-truth Annotation

Videos of tennis games obtained from TV broadcast in standard resolution (SD) were processed and Yan *et al.*’s ball tracker [161] was used to detect relevant instances in time, i.e., when the velocity vector of the ball suddenly changed. The frames of those relevant instances were analysed and using a method of background subtraction, the players were detected.

For each detected player, a single HOG3D feature vector [89] is extracted. The HOG3D descriptors are reported to be among the top performing methods evaluated in Wang *et al.*’s survey [153]. As described in details in Section 2.2.1, HOG3D is a three dimensional generalisation of SIFT [108] or local histograms of oriented gradients.

deCampos *et al.*’s datasets [37] (TSWA03 and TDWA09 of Table 3.2), was obtained by processing two videos of tennis. The processing and annotation of the remaining datasets of Table 3.2
3.1. ACASVA: Player Actions Database

<table>
<thead>
<tr>
<th>label</th>
<th>sport</th>
<th>gender</th>
<th>number</th>
<th>competition</th>
<th>year</th>
<th>non-hit</th>
<th>hit</th>
<th>serve</th>
</tr>
</thead>
<tbody>
<tr>
<td>TWSA03</td>
<td>Tennis</td>
<td>Women</td>
<td>Singles</td>
<td>Australian</td>
<td>2003</td>
<td>944</td>
<td>214</td>
<td>72</td>
</tr>
<tr>
<td>TMSA03</td>
<td>Tennis</td>
<td>Men</td>
<td>Singles</td>
<td>Australian</td>
<td>2003</td>
<td>1881</td>
<td>469</td>
<td>123</td>
</tr>
<tr>
<td>TWDA09</td>
<td>Tennis</td>
<td>Women</td>
<td>Doubles</td>
<td>Australian</td>
<td>2009</td>
<td>1064</td>
<td>135</td>
<td>36</td>
</tr>
<tr>
<td>TWSJ09</td>
<td>Tennis</td>
<td>Men</td>
<td>Single</td>
<td>Japan</td>
<td>2009</td>
<td>859</td>
<td>224</td>
<td>59</td>
</tr>
<tr>
<td>BMSB08</td>
<td>Badminton</td>
<td>Men</td>
<td>Singles</td>
<td>Beijing</td>
<td>2008</td>
<td>706</td>
<td>458</td>
<td>8</td>
</tr>
</tbody>
</table>

are a contribution of the work done in this thesis. Extracted player actions of each video are labelled as serve, hit and non-hit. The number of samples in each class is shown in Table 3.2.

Figure 3.6 presents sample frames of a video of singles (TWSA03, recorded in PAL) and a video of doubles tennis (TWDA09, recorded in NTSC) and some sample actions. The two videos have different players in different costumes which in practice will affect the texture extracted information additionally, in the video of tennis single, the players’ scale is in general larger than that of the doubles video. Moreover, the background, illumination conditions and the video acquisition technique varies from one video to the other.

Figure 3.6: Sample images with players performing each action, taken from the de Campos et al.’s dataset, obtained from [37] ©IEEE 2011. The number of samples for each class is also shown next to their label.
Figure 3.7: Sample images and players performing each action from the Badminton dataset.

The video sequences are divided into point-based clips, such that each sequence provides several samples of actions performed by the same players under similar conditions (e.g. the camera set-up and video coding method remain constant in each match). Within each sequence, low level descriptors thus ought to share features that relate to the style of the players, to their appearance and to the appearance of the background and illumination. However, all of these features are subject to variation from one sequence to another and often these variations jeopardise the balance between generalisation and discriminative power of classifiers. Further, domain disparity is brought about by rule changes (e.g. number of players and their spacial distribution in singles and doubles matches). There is thus a clear requirement for an appropriate adaptation in order to solve the automatic sports video annotation problem in the different domains. Actions here, as stated before, are characterised by HOG3D features [89] (see Section 2.2.1) extracted for the bounding box of each detected player, generating a problem of large intrinsic dimensionality.

Despite having a great number of similarities in their appearance, tennis and badminton games
are quite different in their technical features especially when they are automatically analysed. Table 3.3 presents a comparison between these two games in terms of game rules, visual characteristics of the footage backgrounds and their general appearance. Figure 3.7 shows a sample frame of a badminton video and some sample player actions.

Table 3.3 Comparison between Tennis and Badminton. Note that, in the table, match intensity refers to the actual time the ball/shuttle was in flight, divided by the length of the match.

<table>
<thead>
<tr>
<th>Concepts</th>
<th>Tennis</th>
<th>Badminton</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time [7]:</td>
<td>3 hours and 18 minutes</td>
<td>1 hour and 16 minutes</td>
</tr>
<tr>
<td>Ball/Shuttle in Play [7]:</td>
<td>18 minutes</td>
<td>37 minutes</td>
</tr>
<tr>
<td>Match Intensity [7]:</td>
<td>9%</td>
<td>48%</td>
</tr>
<tr>
<td>Rallies[7]:</td>
<td>299</td>
<td>146</td>
</tr>
<tr>
<td>Shots [7]:</td>
<td>1,004</td>
<td>1,972</td>
</tr>
<tr>
<td>Shots Per Rally [7]:</td>
<td>3.4</td>
<td>13.5</td>
</tr>
<tr>
<td>Distance Covered [7]:</td>
<td>2 miles</td>
<td>4 miles</td>
</tr>
<tr>
<td>Dominant hits:</td>
<td>Side hitting of ball [77]</td>
<td>Smash shots [163]</td>
</tr>
<tr>
<td>Body sections involved:</td>
<td>Shoulder and arm [77]</td>
<td>Wrist and forearm [163]</td>
</tr>
<tr>
<td>Ball speed:</td>
<td>150-246 km/h [77]</td>
<td>332 km/h [163]</td>
</tr>
<tr>
<td>Serve rule</td>
<td>Second chance of serving ball [77]</td>
<td>Single chance in serves [163]</td>
</tr>
<tr>
<td>Serving position</td>
<td>Serve from behind the baseline [77]</td>
<td>Serve in front of the baseline [163]</td>
</tr>
<tr>
<td>Ball/shuttle trajectory in serve:</td>
<td>Going down [77]</td>
<td>Rising up and then down [163]</td>
</tr>
<tr>
<td>Player pose in serve:</td>
<td>Hands above the head [77]</td>
<td>Hands in front of the torso [163]</td>
</tr>
<tr>
<td>Court size:</td>
<td>$36 \times 78 (m^2)$ [77]</td>
<td>$6.1 \times 13.4 (m^2)$ [163]</td>
</tr>
<tr>
<td>Net height:</td>
<td>The net is 1.07 meters (3 feet 6 inches) high at the posts and 0.914 meters (3 feet) high in the center [77]</td>
<td>net is 1.55 meters (5ft 1 inch) high at the edges and 1.524 meters (5ft) high in the center [163]</td>
</tr>
</tbody>
</table>
3.1.3 Tennis and Badminton Ground Truth Annotation System

In order to set a baseline standard for the annotation system of Section 3.1.1 and to measure the accuracy levels of the novel methodologies presented in this thesis, it is important to have a tool that can be employed to generate error-free labelling (i.e. a ground truth annotation). For this purpose, deCampos [37] built a system, capable of frame-wise and/or event-wise (i.e. every few frames when a “key event” takes place) annotation of video frames with spatial information. Figure 3.8 shows the visual appearance of this annotation toolbox.

With this tool, a particular frame can be annotated with the following meta-data:

- Ball and players locations in the image plane i.e. the input image co-ordinates.

- Ball and players locations in the court co-ordinates i.e. the top view of the tennis or badminton court.

- Event descriptors such as Serve, Hit, Bounce and Net.
• Using all of the above information, a high-level symbol is generated identifying the type and location of an event (see Tables 3.1 and 3.4)

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF</td>
<td>Serve by Far player</td>
</tr>
<tr>
<td>SN</td>
<td>Serve by Near player</td>
</tr>
<tr>
<td>BIF</td>
<td>Bounce Inside Far player’s half court</td>
</tr>
<tr>
<td>BOF</td>
<td>Bounce Outside Far player’s half court</td>
</tr>
<tr>
<td>BIN</td>
<td>Bounce Inside Near player’s half court</td>
</tr>
<tr>
<td>BON</td>
<td>Bounce Outside Near player’s half court</td>
</tr>
<tr>
<td>HF</td>
<td>Hit by Far player</td>
</tr>
<tr>
<td>HN</td>
<td>Hit by Near player</td>
</tr>
</tbody>
</table>

In addition to the above features, this annotation tool is also capable of reading the output of the system introduced in Section 3.1.1. Thus, instead of newly annotating every single frame for player and ball locations, we can simply adjust any errors made by the system (or newly annotate only the missed frames), resulting in a speedy groundtruthing process.

As a part of the contribution of this thesis, we used this tool to annotate badminton videos where the play structure is similar to tennis. Note, these annotations are only event descriptors and do not contain information related to point allocations.

The ACASVA player action dataset has been made available for download in [86, 20] (From September 2012 to January 2015 this dataset was downloaded 835 times from the Machine Learning Dataset Repository website [20]).

### 3.1.4 Feature Space Visualisations

In this section we compared two different sets of parameters which were proposed for HOG3D feature extraction. In the first attempt of feature extraction, the parameters used for HOG3D were those optimised for the KTH dataset in [89]. They generate 960D vectors \((4 \times 4 \times 3 \times 20)\) using a \(4 \times 4\) grid in space and 3 splits in time. For each sub-block, a histogram
of edge orientations is quantised using a icosahedron (20 faces regular polyhedron) where the features take values in the range $[0, 1]$ with relatively small correlation between each other (see Figure 3.9). As for the second HOG3D parameter setting, the data volume is partitioned by a $2 \times 2$ grid in space and 5 splits in time. For each sub-block, a histogram of edge orientations is obtained using polar coordinates with 5 orientations in space and 3 orientations in time. These parameters were optimised for the Hollywood dataset as claimed in [90], since in videos the variations in time are of more importance. So in total we ended up with a 300D feature vector for the second dataset.

Figure 3.9: Absolute values of the correlation matrix [42] of the HOG3D feature space (960 dimensions), obtained from the 1230 action samples of TWSA03 dataset. The emergence of some mild patterns at every 60 dimensions is probably caused by the use of 3 temporal splits and icosahedron for edge orientation quantisation (see detailed descriptions in 3.1.4).

For a quantitative evaluation of how well the representation learned from the tennis dataset can represent badminton actions, we used LDA (Linear Discriminant analysis) [130] described in Section 2.3.2. Figure 3.10 graphically shows the result of LDA projection, where the source data comes from the 300D TWSA03 (tennis match) and the target is 300D BMSB08 (badminton match) dataset containing 3 action classes. Consequently, the data in the LDA space
lies in a 2D space. As it can be seen in both far and near player projection figures 3.10(a) and 3.10(b), the two Hit and Non-Hit classes of the target data are mapped into the same location as of the source Non-Hit class, \textit{i.e.} the source domain LDA-projected representation is not appropriate to be used for separating the target domain classes.

![Near players data projection](image1)

![Far players data projection](image2)

Figure 3.10: Data projection in LDA space where Blue colour represents the data in source/tennis domain and the markers ‘o’, ‘.’ and ‘x’ stand for Non-Hit, hit and serve classes respectively.

Another visualisation aspect can be seen from the point of view of kernel data remapping. Since the classification which has been used in this work is a kernel discriminant analysis (KDA) classifier, it is worth comparing the train and test kernel matrices which are defined by an RBF kernel as explained in Section 2.4.3 where the city-block metric [78] (or $\ell_1$ distance) was selected as the distance measure. To facilitate the visualisation, the initial matrices are reordered to show entries in the order of Non-Hit, Hit and finally Serve class samples. An image of these kernels is in Figure 3.11.

One can instantly notice the higher intensity values of the train kernel compared to the test kernel. A closer look at the RBF equation (eq. 2.22) explains this finding. Due to dissimilarity of the source and target feature vectors, their estimated distance (the numerator of the exponential power at eq. 2.22) exceeds the variance in the source (denominator of the exponential power
at eq. 2.22). The absolute value of the exponential therefore increases, resulting in a higher negative exponential power and subsequently a lower test kernel intensity value. The results show that there is a clear separation between the serve class and the other two (at the bottom of the figure).
right corner of the source image) but not a clear discrimination can be detected between the Hit and Non-Hit classes, which again confirms how challenging the classification-transfer task is.

In the remaining sections of this chapter we will introduce several public image-databases which are later used for comparing our transfer algorithms with other state-of-the-art transfer approaches.

### 3.2 Chars74K Digit Dataset

Character recognition is a classic pattern recognition problem. With today’s omnipresence of cameras, the applications of automatic character recognition are broader than ever. For Latin script, this is largely considered a solved problem in constrained situations, such as images of scanned documents containing common character fonts and uniform background. However, images obtained with popular cameras and hand held devices still pose a formidable challenge for character recognition. The challenging aspects of this problem are evident in the Chars74K dataset [38]. In that dataset, Latin script (excluding accents) and Hindu-Arabic numerals are available. The dataset consists of:

- 64 classes (0-9, A-Z, a-z)
- 7705 characters obtained from natural images
- 3410 hand drawn characters using a tablet PC
- 62992 synthesised characters from computer fonts

This gives a total of over 74K images (which explains the name of the dataset). Figure 3.12 shows sample characters and digits from Chars74k database.

In our experimental evaluation we only use the synthesised characters subset of the Chars74K database which is a computer generated dataset composed of 254 English fonts synthesised in four styles; Normal, Bold, Italic and Bold Italic. Out of 64 possible classes of upper and lower
Figure 3.12: Sample characters and digits from the Chars74k database [38].

case English letters and zero to nine numbers, we evaluate our method on the 10 classes of digits. The resolution of the images is reduced to $32 \times 32$ for computational efficiency. In this dataset, the vectorised intensity values of each image are used as a feature vector.

### 3.3 MNIST-USPS Database

*USPS* dataset refers to numeric data obtained from the scanning of handwritten digits of postal codes on envelopes carried out by the U.S. Postal Service (see Figure 3.13). The original scanned digits are binary and of different sizes and orientations; the images have been translated and size normalised, resulting in $16 \times 16$ grey-scale images [31]. The *USPS* dataset consists of 7,291 training images and 2,007 test observations, distributed as introduced in Table 3.5.
3.3. MNIST-USPS Database

Table 3.5 USPS data distribution [31].

<table>
<thead>
<tr>
<th>Class</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>1194</td>
<td>1005</td>
<td>731</td>
<td>658</td>
<td>652</td>
<td>556</td>
<td>664</td>
<td>645</td>
<td>542</td>
<td>644</td>
<td>7291</td>
</tr>
<tr>
<td>Test</td>
<td>359</td>
<td>264</td>
<td>198</td>
<td>166</td>
<td>200</td>
<td>147</td>
<td>166</td>
<td>177</td>
<td></td>
<td></td>
<td>2007</td>
</tr>
</tbody>
</table>

Figure 3.13: Examples of normalised digits from USPS [31].

The MNIST dataset has a training set of 60,000 examples and a test set of 10,000 examples. It is a subset of the NIST database. The original black and white (bilevel) images from the NIST database were size normalized to fit in a 20 × 20 pixel box while preserving their aspect ratio. The resulting images contain grey-scales as a result of the anti-aliasing technique used by the normalisation algorithm. In order to obtain the MNIST data, the NIST images were centred in a 28 × 28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28 × 28 field (see Figure 3.14). The MNIST database was constructed from NIST’s Special Database 3 (SD3) and Special Database 1 (SD1) which contain binary images of handwritten digits while the SD3 subset is much cleaner and easier to recognise than SD1. The reason for this is that SD3 was collected among Census Bureau employees, while SD1 was collected among high-school students. The MNIST training set is composed of 30,000 patterns from SD3 and 30,000 patterns from SD1 and the test set was composed of 5,000 patterns from SD3 and 5,000 patterns from SD1. The 60,000 pattern
training set contains examples from approximately 250 writers. Disjoint sets of writers were engaged in constructing the training and test sets.

USPS and MNIST datasets follow very different distributions but they share 10 classes of digits. We follow the settings of [107] for USPS → MNIST using their randomly selected samples composed of 1,800 images from USPS as the source data, and 2,000 images in MNIST to form the target data and also switch source-target pairs to get another dataset MNIST → USPS. The images are uniformly rescaled to 16 × 16 pixels, and each represented by a feature vector, encoding the gray-scale pixel values. Hence the source and target data can share the same feature space.

![Examples of normalised digits from MNIST](image)

Figure 3.14: Examples of normalised digits from MNIST [100, 99].

### 3.4 Caltech+Office Database

*CALTECH+OFFICE* is composed of 10-classes of sample taken from four datasets; Amazon (images downloaded from online merchants), Webcam (low-resolution images by web camera), DSLR (high-resolution images by a digital SLR camera) and Caltech-256. The first three datasets (Amazon, Webcam and DSLR) are known as OFFICE database [140]. We
followed [61] to focus on 10 common classes: Back-pack, Touring-bike, Calculator, Headphones, Computer-keyboard, Laptop, Computer-monitor, Computer-mouse, Coffee-mug and Video-projector. Each dataset is assumed as a different domain and there are between 8 and 151 samples per category per domain, and 2533 images in total (see Figure 3.15), and adopted the experimental settings used in [61, 11]. Two types of features are extracted for this dataset; BoW on SURF descriptors (see Section 2.2.2) and DeCAF features (see Section 2.2.3).

SURF features were extracted and the images encoded with 800-bins histograms with the codebook trained from a subset of Amazon images. The histograms were then normalised and z-scored to follow a normal distribution in each dimension.

Deep Convolutional Activation features (DeCAF) [41] have shown promising performance for object recognition. They are extracted by first training a deep conventional model in a fully supervised setting using the state-of-the-art method of [93]. We use the outputs from the 6th layer as the visual features, leading to 4,096D DeCAF6 features.

### 3.5 COIL20 Dataset

COIL20 contains 20 object classes with 1,440 images. The images of each object were taken 5 degrees apart as the object is rotated on a turntable and each object has 72 images. Each image is $32 \times 32$ pixels with 256 gray levels. In experiments, we follow the settings of [107] and partition the dataset into two subsets: COIL1 and COIL2. COIL1 contains images of objects in the orientations of $[0^\circ, 85^\circ] \cup [180^\circ, 265^\circ]$ (quadrants 1 and 3, see Figure 3.16); COIL2 contains images in the orientations of $[90^\circ, 175^\circ] \cup [270^\circ, 355^\circ]$ (quadrants 2 and 4, see Figure 3.16). In this way, subsets COIL1 and COIL2 will follow different distributions. One dataset, COIL1 $\rightarrow$ COIL2, is constructed by selecting all 720 images in COIL1 to form the source data, and all 720 images in COIL2 to form the target data. Source-target pairs are switched to form another dataset COIL2 $\rightarrow$ COIL1.
Figure 3.15: Caltech+Office Database: from left to right sample images of Amazon [140], Caltech256 [67], DSLR [140] and Webcam [140]. Database composed of 10 classes of images captured by different devices and with varying image quality and acquisition conditions.
Figure 3.16: Sample objects from the Columbia University Image Library (COIL1 [118] in 0°).
Figure 3.17: Sample objects from the Columbia University Image Library (COIL2 [118] in 90°).
3.6 Summary

In this chapter four well-known cross-domain image databases were introduced for evaluating and benching of our transfer systems. Following Long et al. [107], we carried out a pre-processing $l_2$-normalisation on the raw images of MNIST, USPS, COIL1 and COIL2 datasets.

Additionally, we described our own video player-action database composed of action features extracted from tennis and badminton videos of varying acquisition characteristics.
Chapter 4

Posterior Adaptation via Feature Space Transformation
In this chapter we propose a domain adaptation method for the problem of recognizing common actions between differing court-game sport videos (in particular tennis and badminton games). Actions are characterized in terms of HOG3D features extracted at the bounding box of each detected player, and thus have large intrinsic dimensionality. The techniques evaluated here for domain adaptation are based on estimating linear transformations to adapt the source domain features in order to maximize the similarity between posterior PDFs for each class in the source domain and the expected posterior PDF for each class in the target domain. We thus demonstrate that significant performance improvement can be achieved by applying domain adaptation in this context.

4.1 Introduction

We consider the application scenario of an autonomous system that is capable of detecting when a change of domain happens. An example is that of Almajai et al. [5], who present a method that detects anomalies if a system trained to automatically annotate videos of tennis singles is presented with videos of tennis doubles. Their system uses an effective ball tracker [161] to detect sequences of events with the HMM-based method of [4]. It also uses player action classification cues, but does not use the number of detected people as a cue so the analysis is purely based on the sequence of events.

Once a change of domain is detected, a system for automatic video annotation can easily start to gather data from the new (target) domain in order to adapt the models for it. If the application considered allows off-line processing, such as video annotation for after-match analysis or for data retrieval, such a scenario is possible and we show that it leads to better performance on action classification.

This problem characterises a case of transductive transfer learning, as defined by Pan and Yang [123]: we aim to improve the learning of the target predictive function \( P(Y_{\text{trg}} | X_{\text{trg}}) \) in \( D_{\text{trg}} \) using the knowledge in \( D_{\text{src}} \) and \( T_{\text{src}} \), where \( D_{\text{src}} \neq D_{\text{trg}} \). The idea is to learn a transformation of the source data to better match the target space in order to improve the result.
of the initial predictions of target class-conditional posterior probabilities. For that, we evaluate two methods that transform the features of $\mathbf{X}^{\text{src}}$ so that they become more similar to $\mathbf{X}^{\text{trg}}$ and the classifier is re-trained using the transformed samples. Our method builds upon insights from Arnold et al. [6].

In the experiments of this chapter, we assume that labels of all the source domain samples are available, so $\text{src} = \text{train}$. We also assume that all the target samples are available but none of their labels are, and the challenge consists in classifying all the target samples, so $\text{trg} = \text{test}$. Once the transfer has been learnt, the retrained classifier should become apt for application on unseen samples in the target domain.

In the next section we give an overview of related work, follow by focusing on the work of Arnold et al. [6] in Section 4.3.1 and proposing modifications to their methodology. Next, we introduce a new transformation in Section 4.3.2. In Section 4.4 we describe the experiments and results. This chapter concludes in Section 4.5 where the contributions are highlighted.

We have published much of the work presented in this chapter in [49] and [47].

### 4.2 Related Work

In domain adaptation, a probability joint distribution $P(\mathbf{Y}, \mathbf{X}^{\text{src}})$ over pattern vectors $\mathbf{X}$ and classes $\mathbf{Y}$ in the source domain is assumed to be related to that of a target domain joint distribution $P(\mathbf{Y}, \mathbf{X}^{\text{trg}})$. We seek to obtain the conditional class probabilities $P(\mathbf{Y}|\mathbf{X}^{\text{trg}})$ using a set of labelled samples distributed according to $P(\mathbf{Y}|\mathbf{X}^{\text{src}})$.

Within the definition of domain adaptation there are many approaches possible that were discussed in Section 2.5.2. Broadly speaking, however, we can split domain adaptation into two distinct areas: the transformative and the non-transformative. In the transformative case (c.f. e.g. [12]), the idea is to find some transformation $G$ of $\mathbf{X}$, such that $P(\mathbf{Y}, G(\mathbf{X}^{\text{src}}))$ and $P(\mathbf{Y}, \mathbf{X}^{\text{trg}})$ can be assumed identical, and the problem of labellings $\mathbf{X}^{\text{trg}}$ becomes a straightforward classification problem. An example of this approach is that of Satpal and
Sarawagi [142], who use feature selection to match source and target distributions. In the non-transformative case, the alternative is to amalgamate labelled source data and unlabelled target data together and treat the problem as one of semi-supervised learning (c.f. [167]), on the assumption of reasonable similarity between $P(Y, X^{src})$ and $P(Y, X^{trg})$.

The problem described above relates to a number of other problems in Machine Learning, such as semi-supervised learning and domain adaptation [123]. Perhaps the main difference is that we assume that a set of samples from the new domain $X^{trg}$ is given all at once and the problem switches to classifying elements of this new domain, i.e., we do not necessarily require the new model to be applicable to samples in $X^{src}$. Transductive transfer learning seems to have been dealt with by a relatively small niche of researchers, despite its broad range of applications.

Dai et al. [33] try to translate $X^{src}$ into $X^{trg}$ so that learning can be done within a single feature space. Their aim is to link the two feature spaces with the construction of a feature translator $p(X^{trg}|X^{src})$. This approach is not directly related to our problem because we assume the typical values $p(X)$ change, but the feature space remains the same $X^{src} = X^{trg}$.

A more related approach is that of estimating a lower dimensional feature space $X^{new}$ to which both source $X^{src}$ and target $X^{trg}$ spaces are mapped [120, 122, 16, 18]. If a good mapping is found, the classifier learnt on the source domain will also be effective on the target domain. The downside is that these approaches may lead to loss of information and they use assumptions that may not generalise to all types of feature spaces.

Methods for domain adaptation can be formulated to apply to our problem. In speech recognition and audio processing, the problem of adaptation to new acoustic environments relate to adaptation of $p(X)$ [158]. One can use the maximum likelihood linear regression (MLLR) to estimate a set of linear transformations for the Gaussian parameters of the HMMs [103]. In [83], a vector Taylor series approach for HMM adaptation was introduced for decoding noisy utterances at test time. An intermediate step of noise adaptive model training is used and results in pseudo-clean model parameters. In [136], Rodriguez et al. proposed an adaptation scheme for semi-continuous HMMs for unsupervised writer style adaptation in handwritten
4.3 Feature Space Class-conditional Transformations

word spotting. In image segmentation, Maximum a Posteriori (MAP) adaptation was used for play field segmentation in order to re-estimate the GMM parameters [10]. All the approaches discussed in this paragraph work on the assumption that the observation data follows a probabilistic generative model.

Another approach can be based on the TSVM [82, 28] (Transductive SVM) which is a model adaptation technique for transductive learning tasks. TSVM tries to iteratively adjust the SVM hyperplanes, previously learnt on source data, until it is well fitted to target space, by minimising the misclassification errors on the test samples.

In this chapter, we employ a transformative approach in which linear transformations are used to adapt the source domain feature distributions to the target domain. The assumption of linearity is particularly advantageous in the video environment given the very large feature dimensionality of video data, both in terms of speeding-up the transformation calculation and also in reducing the danger of over-fitting (a covariance-based transformation would, in contrast, scale with the square of feature dimensionality).

4.3 Feature Space Class-conditional Transformations

In our first proposed TL approach we follow [10, 136, 158, 103, 83] and assume that source clusters follow a probabilistic generative model. Hence in order to adapt the class-conditional distribution mismatch between the corresponding clusters of the two domains, we introduce a set of linear class-specific transformations.

To achieve this, one can assume that a Gaussian Mixture Model fitted to the source classes can be adapted in a way that it matches the target classes. While the general GMM uses full covariance matrices, we follow Reynolds et al. [133] and use only diagonal covariance matrices. This way, the complexity of the estimation system becomes linear in dimensionality, \( f \). We further simplify the model by using only one Gaussian model per class.

In order to adapt the class conditional distributions one can start with an attempt to match
Chapter 4. Posterior Adaptation via Feature Space Transformation

the joint distribution of the features and labels between the corresponding clusters of the two domains. However, as explained earlier in this chapter, labelled samples are not available in the target domain. We thus propose to use posterior probability of the target instances to build class-based models in the target domain.

Let \( X = [x_1, \ldots, x_i, \ldots, x_n]^T \in \mathbb{R}^{n \times f} \) be a set of measurement feature vectors, \( x_i \). The goal is to find a transformation \( G(X) \) such that the PDF of \( P(Y, G(X^{src})) \approx P(Y, X^{trg}) \). In other words, we wish to transform the samples in the source domain so that they become more similar to those observed in the unlabelled target domain. As discussed before, several methods can be applied. We built upon the work of [6] because it is based on linear transformations applied to each individual feature. The transformation is thus estimated and applied for each feature and each class.

4.3.1 Arnold et al. ’s Method and Proposed Modifications (Reweight)

Given the nature of the feature space and the transductive transfer task, the method based on MaxEnt proposed by Arnold et al. [6] seemed to be very appropriate and it possibly is the least expensive in terms of computational cost. In this section we describe it and propose some modifications.

For ease of notation, \( x^j_i \) is the feature \( j \) of sample \( x_i \) and \( E^{src}[x_j, y] \) is used to represent the expectation of feature \( x_j \) with label equal to \( y \), \( \forall x \in X^{src} \):

\[
E^{src}[x_j, y = c] = \frac{1}{n^{src}_{train}} \sum_{i=1}^{n^{src}_{train}} x^j_i \mathbb{1}_c(y_i), \tag{4.1}
\]

\( n^{src}_{train} \) is the number of labelled training data in the source domain and

\[
\mathbb{1}_c(y_i) = \begin{cases} 
1 & \text{if } y_i = c, \\
0 & \text{otherwise}
\end{cases}
\]
The problem in transductive tasks is that the joint distribution of the features with labels differs between the source and target domains, so \( E^{\text{src}}[x_j, y] \) is not the same as \( E^{\text{trg}}[x_j, y] \), \( \forall j \in 1, \cdots, f \), where \( f \) is the dimensionality of \( x \). If the expectations in the train and test datasets are similar, then the model \( \Lambda_{\text{src}} \) learnt on the training data will better generalise to the test data.

As discussed in [6], a transformation \( G(\cdot) \) of the feature space \( \mathcal{X} \) can be learnt such that the joint distributions of the source and target features with their labels are aligned:

\[
E^{\text{trg}}[G(x_j), y] = E^{\text{src}}[G(x_j), y], \forall x_j \in \mathcal{X}.
\]  

(4.2)

It can be too challenging (if not impossible) to estimate a single transformation of the feature space that would generate a space where \( E^{\text{trg}} = E^{\text{src}} \). An alternative would be to have one transformation for the source and another for the target samples. This condition can even further be relaxed by arguing that it is enough to transform only one of the domains, say the source data, so that data from both domains could be separated by a single hyper-plane:

\[
E^{\text{trg}}[x_j, y] = E^{\text{src}}[G(x_j), y], \forall x_j \in \mathcal{X}.
\]  

(4.3)

The problem with this, of course, is that in the unsupervised transductive transfer case, we do not have \( \mathbf{Y}^{\text{trg}} \) and therefore cannot estimate \( E^{\text{trg}}[x_j, y] \). Arnold et al. [6] propose to use the following assumption:

\[
E^{\text{trg}}[x_j, y] \approx E^{\text{trg}}_{\Lambda_{\text{src}}}[x_j, y] = \frac{1}{n^{\text{test}}_{\text{trg}}} \sum_{i=1}^{n^{\text{test}}_{\text{trg}}} x^i_j P_{\Lambda_{\text{src}}}(y|x_i),
\]  

(4.4)

where \( n^{\text{test}}_{\text{trg}} \) is the number of target domain (unlabelled) test examples. Note that in [6], the authors use the joint probability \( P_{\Lambda_{\text{src}}}(y, x_i) \), instead of the posterior \( P_{\Lambda_{\text{src}}}(y|x_i) \). This approximation of \( E^{\text{trg}}[x_j, y] \) may not reflect the true target expectation, but it is the best that can be performed in the unsupervised transductive setting. We tried to use the above definitions in the transfer learning method and obtained poor results.
We suggest that more accurate definitions of $E[x_j, y]$ would have denominators depending on class labels/predictions and propose the following modifications of Equations (4.1) and (4.4):

$$E_{\text{src train}}[x_j, y = c] = \frac{\sum_{i=1}^{n_{\text{src train}}} x^i_j \mathbb{1}[y = c](y_i)}{\sum_{i=1}^{n_{\text{src train}}} \mathbb{1}[y](y_i)}, \quad (4.5)$$

and

$$E_{\text{trg}}[x_j, y = c] \approx E_{\Lambda_{\text{src}}}^{\text{trg}}[x_j, y = c] = \frac{\sum_{i=1}^{n_{\text{trg}}} x^i_j P_{\Lambda_{\text{src}}}(y|x_i)}{\sum_{i=1}^{n_{\text{trg}}} P_{\Lambda_{\text{src}}}(y|x_i)}. \quad (4.6)$$

Based on these expectations the source domain transformation $G(\cdot)$ is defined as:

$$\forall i = 1: n_{\text{src train}}^{\text{trg}}, \quad G(x^i_j) = x^i_j \frac{E_{\Lambda_{\text{src}}}^{\text{trg}}[x_j, y_i]}{E_{\text{src}}^{\text{trg}}[x_j, y_i]} \quad (4.7)$$

i.e. the effect is to re-scale $x_j$. In the case of binary features (which are common in Natural Language Processing), this is equivalent to give more weight to features that occur frequently in the target but rarely in the source (in a conditional sense), and to down weight features that are common in the source but seldom seen in the target [6].

In practice, since the target expectation $E_{\Lambda_{\text{src}}}^{\text{trg}}[x_j, y]$ is only an approximation as it depends on the result of $P_{\Lambda_{\text{src}}}(y|x^{\text{trg}})$ to reduce the effect of poor estimation of the posterior, the transformed features need to be smoothed with the original ones as follows:

$$G'(x^i_j) = (1 - \theta)x^i_j + \theta G(x^i_j), \quad (4.8)$$

where $\theta$ controls the degree to which we use the target conditional estimates to alter the source conditionals. One can note that, $\theta$ equal to zero infers no adaptation and $\theta$ equal to one infers solely relying on the proposed transformation. Figure 4.1 is a synthetic representation of this linear reweighting transformations.

Once the labelled samples have been transformed by $G'(\cdot)$, it is necessary to update the model $\Lambda_{\text{src}}$ and retrain the classifier. If a kernel method is used, this also means the kernels have to be
4.3. Feature Space Class-conditional Transformations

![Figure 4.1: Linear source clusters scaling: the binary classification task where the squares and crosses are representing the samples of each class; the source clusters in the left panel are scaled to better demonstrate the target distribution of the right panel.](image)

Although the reweighting transformation is capable of capturing the changes in expectations from a source distribution to the target, but the expectation is just one aspect of a multi-class data distribution and it is not necessarily model the whole extent of the distribution. This drawback can be better visualised in Figure 4.3 where the source samples (shown with ‘+’ ) of the red class have been transformed to an area of target space where they fall far from the target red class centre (see Figure 4.3(b)). In these cases as shown in Figure 4.3(c) a more accurate transformation composed of a additional translation step will better capture the changes in between the two distributions. In the next section we will propose transformation which takes this into account.

4.3.2 Translating and Scaling Features (TST)

The feature reweighting scheme of Section 4.3.1 is probably ideal for binary feature spaces, but may be too simple for other types of features. We propose to translate and scale the samples of
Chapter 4. Posterior Adaptation via Feature Space Transformation

the training set based on the expected value and standard deviation of features for each class:

\[
\forall i = 1 : n_{\text{train}}^\text{src}, \ y = c \\
G(x_j^i) = \frac{x_j^i - E_{\text{src}}^\text{src}[x_j, y = c]}{\sigma_{j,c}^\text{src}} \sigma_{j,c}^\text{trg} + E_{\Lambda^\text{trg}}^\text{trg}[x_j, y = c],
\]

(4.9)

where \( \sigma_{j,c}^\text{src} \) is the standard deviation of feature \( x_j \) of the source samples labelled as \( y = c \):

\[
\sigma_{j,c}^\text{src} = \sqrt{\frac{\sum_{k=1}^{n_{\text{test}}^\text{src}} (x_j^k - E_{\text{src}}^\text{src}[x_j, y = c])^2}{\sum_{i=1}^{n_{\text{train}}^\text{src}} \mathbb{1}[y(c)]}}.
\]

(4.10)

and

\[
\sigma_{j,c}^\text{trg} = \sqrt{\frac{\sum_{k=1}^{n_{\text{test}}^\text{trg}} (x_j^k - E_{\Lambda^\text{trg}}^\text{trg}[x_j, y = c])^2 P_{\Lambda^\text{src}}(y = c | x_k)}{\sum_{k=1}^{n_{\text{test}}^\text{trg}} P_{\Lambda^\text{src}}(y = c | x_k)}}.
\]

(4.11)

The effect is to align the expected values of the two domains and to make sure that the standard deviation of source and target domain distributions are similar. Figure 4.2 demonstrates a synthetic representation of our proposed translation + scaling linear transformations.

![Figure 4.2: Linear source clusters translation and scaling: the binary classification task where the squares and crosses are representing the samples of each class; the source clusters in the left panel are scaled and translated to better demonstrate the target distribution of the right panel.](image)

The smoothing function is then applied as before (eq. 4.8).
4.4 Experiments and Results

We evaluate and compare the performance of both proposed transfer methods of this chapter: reweight and TST. In the next section we will first verify the proposed transfer methods by testing them on a synthetic 2D data composed of 3 distinct class distributions. We then show the performance on the player actions classification problem, transferring between tennis singles to tennis doubles and badminton to tennis singles.

4.4.1 Synthetic 3-Class Dataset Experiment

In order to have a clear visualisation of the algorithm a synthetic data set in a 2-D space is generated. This dataset is composed of 3 sets of randomly drawn samples (100 samples per class) for both source and target spaces while the marginal distribution of the data changes from source to target. The data in the source space is shown in Figure 4.3 which includes source samples transformed to a target space along with the target classes as described in its caption. We show the adaptation results obtained using both methods explained in Sections 4.3.1 and 4.3.2. The transfer rate parameter is chosen to maximise the accuracy for each case. As one can note in Figures 4.3(b), 4.3(c) both TST and reweighting lead to an acceptable level of performance boost especially with higher values of the transfer rate.

4.4.2 Transfer from Tennis Single to Double Matches

Experimental Setup

For player action data of Section 3.1, we use HOG3D feature vectors [89] extracted from the bounding box of each player, with a buffer of 24 frames around the key event (e.g. when the player hits the ball). We used the parameters optimised for the KTH dataset, which give 960 dimensional vectors, as described in [89] and Section 3.1.

For classification, we followed [37] and used KDA (Kernelised Linear discriminant analysis [22]).
Figure 4.4 shows a comparison in the performance of the KDA classifier on tennis single (TWSA03 see Table 3.2) where the three distance metrics: \( \ell_1 \), \( \ell_2 \) and \( \chi^2 \) are used for kernel computation. The main difference with [37] is that instead of using the \( \chi^2 \) measure to build the RBF kernel functions, we used the \( \ell_1 \) distance because \( \chi^2 \) is not a metric (as the triangle inequality does not hold for it) and \( \ell_1 \) seem to be better than \( \ell_2 \) to compare histogram-based feature vectors. In this experiments we followed the experimental setting of [37] and the results are presented in terms of the ROC curves, as well as mean-class accuracy.

As one can note, the \( \ell_1 \) metric exhibits a slight superiority compared with the other two measures.

Given the training kernels, KDA generates a \( |\mathcal{Y}| - 1 \) dimensional space where samples are projected (\( |\mathcal{Y}| \) is the number of classes). Due to the nature of this dataset, KDA over-fits the data, so all the samples in the training set belonging to the same class are projected to the same point, making it impossible to estimate the covariance matrix for each class.

This problem was not experienced in [37] because the authors used KDA, without a generative data model. In order to apply the transfer algorithms of Sections 4.3.1 and 4.3.2, it is necessary to estimate \( \Lambda_{src} \) of \( P_{\Lambda_{src}}(Y|X) \). For that, we used a five-fold cross validation in the training set to obtain estimates of the mean and covariance matrix for each class.
4.4. Experiments and Results

The Effect of $\theta$ on the Transfer Results

As discussed in Section 3.1, the games of singles and doubles in the dataset are quite different from each other. In [37], the authors simply took the game of singles (TWSA03) for training and the doubles (TWDA09) for testing. In this section, we evaluate the use of transductive transfer learning to improve the classifier performance.

First of all we reproduced the experiment of [37]. To obtain a consistency in results comparisons we followed [37] and additionally evaluated the results in terms of mean Area Under the ROC Curve (mAUC) and obtained 90.3%. In the same experiments (training on singles, testing on doubles, without using transfer learning), we obtained an mAUC of 91.2%. Our baseline is slightly higher than that of [37] probably because of the use of a full generative model in $P_{\lambda_{src}}(Y|X)$ and because we use $\ell_1$ instead of $\chi^2$ for the kernel radial basis function.

In a preliminary evaluation of the mAUC as a function of the transfer rate $\theta$ of (4.8), we observed that the presented methods did not lead to a significant variation in mAUC, but if the results are presented in terms of the mean accuracy (mAcc), the benefit becomes clearer. The mAcc is measured by averaging out the correct classification rate for each class, giving equal importance to all classes, so it is not affected by data skew. On the same experiment as above, we obtained an mAcc of 58.72%, which seems quite low, but this is a challenging dataset.

Figure 4.5 presents the mAcc as a function of the transfer rate. Note that the method of Section 4.3.1 (reweight) presents a steady climb in mAcc as the transfer rate grows, but the method of Section 4.3.2 (TST) reaches a higher peak of classification performance, presenting an improvement of nearly 25%. The confusion matrix obtained with $\theta = 0.4$ is shown in Figure 4.5(c). In brackets, the same figure also shows the results presented in [37], which hand-picked thresholds on the classifier output (“thresholds selected so that the true positive rate is 77.62% and the false positive rate is 22.38%”). Note that our results are better at distinguishing serves and non-hits, but hits get confused with non-hits more often, whereas in [37] non-hits are often confused with serves.

After $\theta = 0.4$, the performance of TST starts to drop possibly because its transformation is
more complex and may lead to over-fitting to unlabelled data that was not necessarily classified correctly.

Figure 4.5(b) shows the results obtained by swapping the source (train) and target (test) samples. Note that with conservative transfer rate both methods increase the performance, but for $\theta \geq 0.4$, TST leads to negative transfer. This failure can be justified by noting the fact that the distribution of the samples in doubles match (which is now considered as training data) is more diverse and the training data size is much larger compared to the previous experiment. This resulted in a better initial fitting of the source model to the target data which in fact depicts the failure of the proposed trans + scaling transformation. In scenario where the source-trained model shows an acceptable level of performance on the target, even a few faulty initial target label predictions can lead in a negative transfer. This in turn results in a drop in performance. To avoid this, it is necessary to pick a more conservative transfer rate ($\theta$) and iteratively and gradually transform the source distribution.

For an additional analysis, we also performed a five-fold cross validation experiment on each domain to estimate what would be the best expected result of using transductive transfer. The resulting mAcc was: Singles: 92.29% and Doubles: 73.34%. These results are shown as cross-val in Figures 4.5(a) and 4.5(b).

Note in Figure 4.5(a) that both methods lead to a better performance than the cross-validation result on the target domain. In other words, both were able to achieve full benefit of having the test samples as unlabelled data for transfer. The same is not true in Figure 4.5(b). We suggest that this is because the doubles dataset is more unbalanced and noisier than the singles dataset, thus the cross-validation results on doubles are not so good but on singles they were very good. This also means that when the singles game is used as the test set, the baseline mAcc is much higher and the improvement of 3.69% in mAcc (obtained with reweight and $\theta = 0.4$) actually means a reduction of 5.07% in mean error rate, which is quite significant.
4.4.3 Badminton Dataset Experiment

A similar set of experiments were carried on transforming from Badminton (BMSB08) to Tennis. The results of these experiments are shown in Figure 4.6 where the transfer task is designed from badminton domain to three different tennis matches of varying video qualities.

As can be seen the results do not present a performance boost as significant as that of Figure 4.8, but there is always a performance improvement with conservative values of $\theta$. The TST result was sometimes disappointing, but it always lead to performance improvement with low values of $\theta$.

Running the KDA classification method using the Badminton data as a training set and Tennis as the target for testing always results in a better detection accuracy in comparison with the Tennis game considered as a training set, even in the case of reducing the Tennis set to a
subset, which is more balanced regarding the Non-hit and Hit labels (with a proportionality of 285/441 for Hit versus Non-hit), showed better results for training on Badminton and testing on Tennis. The same experiment was undertaken for Tennis-doubles data versus Badminton and the accuracy level was still higher for Badminton. This can be explained by the differences in the nature of these games with Badminton containing hits of greater diversity compared to Tennis. In other words, side hits in Tennis are more dominant while in Badminton, different types of hits including side hits, smashes and hits at various angles with respect to player’s trunk can be noticed.

4.4.4 Transductive Transfer with Additional Target Adaptation Set

In this section we assumed that we have access to an adaptation target set where its samples are drawn from a distribution which is more similar to our test target distribution when compared to the source vs. test target e.g. in a badminton (BMSB08) to tennis transfer task we have access to samples from two different tennis single matches (TWSA03 and TMSA03) where one of these unlabelled target sets is uses as an adaptation set to estimate the source-to-target transformation and the second target set is used for the final model evaluation.

In Section 4.4.2 a fully automatic player detection method was used for extracting key frames, which in some instances resulted in merging and miss-labelling of bounding boxes. In this section we focus on the action classification task and allowed for some manual correction of bounding boxes and their labels for training and evaluation of results. This explains why there are less samples per class in comparison to previous section experiments, which included false positive player action detections. For this reason, we repeat the experiments of Section 4.4.2.

The Transfer Results

The results are shown in rows (a–d) of Table 4.1, where rows (a) and (c) are the baseline (without domain adaptation, DA) and rows (b) and (d) show the DA results with both methods described in Section 4.3. We used experiments (b) and (d) to evaluate a range of $\theta$ values
4.4. Experiments and Results

Table 4.1 Baseline results and results with two methods for domain adaptation (DA): reweight|TST, in %.

<table>
<thead>
<tr>
<th>source</th>
<th>target adaptation</th>
<th>test</th>
<th>accuracy per class (%)</th>
<th>macro average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>source</td>
<td></td>
<td>non-hit</td>
<td>hit</td>
</tr>
<tr>
<td>a</td>
<td>TWSA03</td>
<td>TWDA09</td>
<td>99.6</td>
<td>14.9</td>
</tr>
<tr>
<td>b</td>
<td>TWSA03</td>
<td>TWDA09</td>
<td>93.9/93.9</td>
<td>41.8/43.3</td>
</tr>
<tr>
<td>c</td>
<td>TWDA09</td>
<td>TWSA03</td>
<td>97.8</td>
<td>30.5</td>
</tr>
<tr>
<td>d</td>
<td>TWDA09</td>
<td>TWSA03</td>
<td>87.0/91.2</td>
<td>67.6/63.4</td>
</tr>
<tr>
<td>e</td>
<td>TWSA03</td>
<td>TWSS03</td>
<td>98.1</td>
<td>24.8</td>
</tr>
<tr>
<td>f</td>
<td>TWSA03</td>
<td>TWSS03</td>
<td>97.5/97.3</td>
<td>42.7/44.2</td>
</tr>
<tr>
<td>g</td>
<td>BMSB08</td>
<td>TWSS03</td>
<td>35.9</td>
<td>77.9</td>
</tr>
<tr>
<td>h</td>
<td>BMSB08</td>
<td>TWSS03</td>
<td>32.7/39.3</td>
<td>88.6/85.2</td>
</tr>
<tr>
<td>i</td>
<td>BMSB08 +TWSA03</td>
<td>TWSS03</td>
<td>94.0</td>
<td>35.7</td>
</tr>
<tr>
<td>j</td>
<td>BMSB08 +TWSA03</td>
<td>TWSS03</td>
<td>97.5/91.7</td>
<td>42.7/54.7</td>
</tr>
<tr>
<td>k</td>
<td>BMSB08</td>
<td>TWSS03</td>
<td>24.5/33.0</td>
<td>98.3/90.8</td>
</tr>
</tbody>
</table>

for eq. 4.8. Although high values of $\theta$ lead to better results the performance nearly plateaued or worsened after $\theta = 0.6$. Following the observations in Section 4.4.2, we used a more conservative value, $\theta = 0.5$ for the experiments in Table 4.1.

We further present experiments with a more challenging change of domain: from a badminton game (BMSB08) to a tennis game (TMSS03). All the results shown confirm that when DA is used, there is an improvement in performance. All of them also show that the TST method gives better results than reweight, but this is not consistent across all class labels.

In most experiments presented, we show the results using the target set to adapt the parameters and obtain $G'$ (eq. 4.8), which is then applied to the test set, i.e., $X^{trg} = X^{test}$. This is the same scenario evaluated in Section 4.4.2; it considers that all the unlabelled test set samples are available at once, which is a reasonable assumption for batch processing methods. For the experiment in line (k) of Table 4.1, we consider the scenario that samples from one domain (badminton, BMSB08) are used for training and unlabelled samples from a second domain (tennis women’s, TWSA03) are available to compute $G'$. Then one wishes to use the same adaptation parameters $G'$ for another set of test samples that come from a similar domain.
(tennis men’s, TMSA03). The result shows an improvement over the baseline which is in line (g), even though the adaptation set is different from the test set \( X_{\text{trg}} \neq X_{\text{test}} \). Even more notable is that the result of TST in (k) where \( X_{\text{trg}} \neq X_{\text{test}} \) approximates that in (h).

As expected, the results in (k) are not as good as those in (i) and (j), which used the tennis women’s game (TWSA03) and its labels to complement the training set. Line (j) shows an upper bound of performance in this scenario, as both BMSB08 and TWSA03 were used to compute their individual adaptation functions (i.e., one \( G' \) for each training set) and their combined adapted sets \( G'(X^{\text{TWSA03}}) \) and \( G'(X^{\text{BMSB08}}) \) were used to re-train the classifier. As expected, the result in (j) is significantly better than that in (i).

Note that owing both to the scarcity of serve samples and to the fact that serves in badminton are very similar to backhand hits in tennis, this class was never detected when badminton was the only game used for training. Figure 4.4.4 highlights the visual diversity between the serves in these two games.

Figure 4.7: Serve detection failure in transfer from Badminton to Tennis, due to their different visual appearance.

This observation suggests the way for future explorations with transfer learning methods to deal
4.4. Experiments and Results

Table 4.2 Results (accuracy in %) obtained by swapping TMSA03 and TWSA03, i.e., using the men’s game for validation or adaptation and the women’s game for test. We follow the same format as in Table 4.1: baseline (top) and reweight/TST (bottom).

<table>
<thead>
<tr>
<th>source</th>
<th>target adaptation</th>
<th>test</th>
<th>accuracy per class (in %)</th>
<th>macro average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>non-hit</td>
<td>hit</td>
</tr>
<tr>
<td>a TMSA03</td>
<td>– TWSA03</td>
<td>TWSA03</td>
<td>97.1</td>
<td>42.7</td>
</tr>
<tr>
<td>b TMSA03</td>
<td>test set TWSA03</td>
<td>93.1</td>
<td>91.7</td>
<td>61.0</td>
</tr>
<tr>
<td>c BMSB08</td>
<td>– TWSA03</td>
<td>TWSA03</td>
<td>39.1</td>
<td>88.3</td>
</tr>
<tr>
<td>d BMSB08</td>
<td>test set TWSA03</td>
<td>36.2</td>
<td>44.0</td>
<td>93.0</td>
</tr>
<tr>
<td>e BMSB08</td>
<td>– TWSA03</td>
<td>TWSA03</td>
<td>96.6</td>
<td>48.8</td>
</tr>
<tr>
<td>f BMSB08 +TMSA03</td>
<td>test set TWSA03</td>
<td>92.1</td>
<td>85.1</td>
<td>70.9</td>
</tr>
<tr>
<td>g BMSB08</td>
<td>TMSA03 TWSA03</td>
<td>TWSA03</td>
<td>30.0</td>
<td>36.9</td>
</tr>
</tbody>
</table>

with changes of domain in which classes may split into sub-categories (e.g. hit → {backhand, forehand}) or merge into super-categories (e.g. {hit, serve} → moving arm).

We also performed experiments swapping the validation (or adaptation) set with the test set. The results, shown in Table 4.2, present a similar pattern to that of Table 4.1, i.e., domain adaptation normally leads to performance improvement. One exception is observed in the experiment (g) which shows that only the TST method gives a better performance than the baseline (c).

Figure 4.8 presents an update of Figure 4.5 using an improved annotation of player bounding boxes and action labels, as discussed in Section 4.4.4. Note that the baseline (θ = 0) is lower in both experiments but the best results with domain adaptation are similar to the best presented in Section 4.4.2. The best results were obtained with values of θ that are higher than those shown in Section 4.4.2. Note that θ = 0.5 is a conservative transfer rate value that leads to a significant improvement in both experiments, and for this reason we chose this value for the experiments presented in Tables 4.1 and 4.2.

Another contrast to the results of Section 4.4.2 is that there is a reduction in the difference between the performance of reweight and TST for the same values of θ. This hints that a more complex transformation may not necessarily lead to better domain adaptation.
4.5 Summary

We based our work on the method introduced in [6], where the source domain features are reweighted based on the ratio of the joint expectation of features and class labels in target and source domains. We then proposed a rectification of Arnold et al. [6] method and a modification employing a more complex transformation, based on translation and scaling of each feature, for each class label.

We presented experiments on a dataset of actions in tennis games and showed that, in one scenario, the proposed method can lead to an increase of nearly 20% in mean accuracy, giving results that are better than a 5-fold cross-validation on the test data set.

We further complemented these experiments using more video sequences and introduced experiments involving different sports. We used a video of badminton for training and tennis for testing. By applying domain adaptation, we obtained an improvement in classification results, even if the video used to compute the adaptation parameters was not the same as that in the test set.

The intended application for the experiments in these datasets was to perform video annotation as part of a system that is able to detect changes of context (or anomalies, as proposed in [88]). When such change is detected, the system can start to gather new unlabelled samples. Once a set of samples is gathered, the system can apply methods of transductive transfer learning in order to adapt the models and to improve classification in this new domain.

The court-game sport video environment is an inherently interesting setting for domain adaptation because of the potentiality for cross-modal (audio and video) domain adaptation, and also the possibility of inductive rule transfer; ultimately this will lead us to consider the problem of simultaneous rule adaptation and low-level feature adaptation as shown in the preliminary results of Windridge et al. [157].

As concluded in the experimental evaluations a more comprehensive transformation composed of translation and scaling is superior to a simple scaling but on the other hand, this transfor-
mation is more sensitive to wrong initial target label predictions which can lead into a negative transfer. In our proposed model we introduced the transfer rate parameter $\theta$, which up to some extent controls this negative transfer effect but due to lack of labelled target data availability in transductive transfer tasks, it is not practical to accurately and automatically tune this parameter for a new transfer task.

One obvious solution would be to have access to a few labelled target samples which can be utilised as an adaptation set for tuning the $\theta$ parameter. This in fact will slightly change our transductive transfer setting to of a partially inductive setting which is not of interest in this thesis and can be seen as a potential objective for future work. In the next chapter we propose an alternative solution by introducing techniques and algorithms which deal with this issue. More specifically, we boost our model by focusing on enhancing the initial target label estimation step via utilising and introducing shared sub-space detection techniques. We expand our methodology and provide more experimental results using more datasets for a broader evaluation of the application of the proposed transductive transfer learning method.
Figure 4.3: (a) A synthetic 2 dimensional dataset of 3 classes (in red, blue and black) where source and target distributions are shown with ‘+’ and ‘o’ respectively. A KDA classifier (Section 2.4.5) is used as the learner. (b) source clusters are transformed via reweighting transformation algorithm. (c) source clusters are transformed via TST transformation algorithm. (d) Reweighting and TST transformations performance as a function of different transfer rate ($\theta$) values.
4.5. Summary

Figure 4.5: (a) Mean accuracy obtained as a function of the transfer rate ($\theta$ of eq. 4.8), using $X_{\text{train}}^{\text{src}}$ = singles (TWSA03) and $X_{\text{test}}^{\text{trg}}$ = doubles (TWDA09). The reweight curve refers to the method of Section 4.3.1 and Trans+scale (TST) refers to the method of Section 4.3.2. The best mAcc of those methods are 78.14% for TST and 74.64% for reweight. The baseline (no transfer) gives 58.72% and the cross-validation on the test set gives 73.34%. (b) Same as Figure 4.5(a) but using $X_{\text{train}}^{\text{src}}$ = doubles and $X_{\text{test}}^{\text{trg}}$ = singles. The peak mean accuracies are: 83.89% (TST), 84.99% (reweight), the baseline is 81.30% and cross-validation on the test set gives 92.29%. (c) Best confusion matrix was obtained with the TST method ($\theta = 0.4$) on the Tennis dataset, training with singles (TWSA03) and testing with doubles (TWDA09). The numbers in the brackets are the results from [37].

Figure 4.8: Mean accuracy obtained as a function of the transfer rate $\theta$ of eq. 4.8 for the two adaptation methods discussed in Section 4.3 on datasets with adjusted player locations.
Chapter 4. Posterior Adaptation via Feature Space Transformation
Chapter 5

Transductive Transfer Machines
In this chapter as well as the previous chapter, the focus is on the case in which only unlabelled samples are available in the target domain and we know that the number of classes is persistent priorly. This is a transductive transfer learning (TTL) problem, i.e., the joint probability distribution of samples and classes in the source domain $P(X^\text{src}, Y^\text{src})$ is assumed to be different, but related to that of a target domain joint distribution $P(X^\text{trg}, Y^\text{trg})$, but labels $Y^\text{trg}$ are not available in the target set. We follow a similar notation to that of [123] (see Table 2.1).

The main focus of this chapter will be to propose methods for enhancing the initial target label estimation step of the trans+scaling algorithm of the previous chapter. We propose Adaptive Transductive Transfer Machine (ATTM) which combines methods that adapt the marginal and the conditional distribution of the samples, so that source and target datasets become more similar, facilitating classification. The proposed ATTM approaches this problem by combining four types of adaptation: solving the task in a lower dimensional space that is shared between the two domains, a set of local transformations to further increase the domain similarity, a classifier parameter adaptation method which modifies the learner for the new domain and finally a set of class-conditional transformations (previously proposed in Chapter 4) aiming to increase the similarity between the posterior probability of samples in the source and target sets.

We further introduce two unsupervised dissimilarity measures which are the backbones of our classifier adaptation approach. ATTM uses these measures to select the best classifier and to further optimise its parameters for a new target domain. We show that our method obtains state-of-the-art results in cross-domain vision datasets, with a significant gain in computational efficiency in comparison to related methods.

We show that our pipeline leads to an improvement over the state-of-the-art in cross-domain image classification, using various datasets of different features.

In the next section, we briefly review related works and give an outline of our contribution. Section 5.2 presents the core components of our method and further discusses the relation between them and previous works. This is followed by a description of our framework and an
5.1 Related work

As discussed in Section 2.5.2, different types of methods can potentially be combined. In this work, we focus on Transferring The Knowledge of Feature Representation and approach the TTL problem by finding a set of transformations that are applied to the source domain samples $G(X_{\text{src}})$ such that the joint distribution of the transformed source samples becomes more similar to that of the target samples, i.e. $P(G(X_{\text{src}}), \hat{Y}_{\text{src}}) \approx P(X_{\text{trg}}, \hat{Y}_{\text{trg}})$, where $\hat{Y}_{\text{trg}}$ are the labels estimated for target domain samples.

Following this line of work, Long et al. [107] proposed to do Joint Distribution Adaptation (JDA) by iteratively adapting both the marginal and conditional distributions using a procedure based on a modification of the Maximum Mean Discrepancy (MMD) algorithm [18]. JDA uses the pseudo target labels to define a shared subspace between the two domains. At each iteration, this method requires the construction and eigen decomposition of an $n \times n$ matrix whose complexity can be up to $O(n^3)$ where $n = n_{\text{src}} + n_{\text{trg}}$ is the total number of samples. Similarly, Gong et al. in [61] proposed a kernel-based domain adaptation method that exploits intrinsic low-dimensional structures in the datasets.

5.2 Marginal and conditional distribution adaptation

We propose the Transductive Transfer Machine (TTM) pipeline which first searches for a global transformation such that the marginal distributions of the two domains become more similar and then with the same objective applies a set of local transformations to each transformed source domain sample. Finally in an iterative scheme, our algorithm aims to reduce the difference between the conditional distributions in source and target spaces where a class-based
Chapter 5. Transductive Transfer Machine

Figure 5.1: The effect of different steps of the pipeline on digits 1 and 2 of the MNIST→USPS datasets, visualised in 2D through PCA. The source dataset (MNIST) is indicated by stars, the target dataset (USPS) is indicated by circles, red indicates samples of digit 1 and blue indicates digit 2.

transformation is applied to each of the transformed source sample. The complexity of the latter step is linear on the number of features in the space, i.e., \(O(f)\).

The following pipeline summarises our Transductive Transfer Machines:

(a) A global linear transformation \(G^1\) is applied to \(X^{src}\) and \(X^{trg}\) such that the marginal \(P(G^1(X^{src}))\) becomes more similar to \(P(G^1(X^{trg}))\).

(b) With the same objective, a local transformation is applied to each transformed source domain sample \(G^2_i(G^1(x_i^{src})))\).

(c) Finally, aiming to reduce the difference between the conditional distributions in source and target spaces, a class-based transformation is applied to each of the transformed source samples \(G^3_{y_i}(G^2_i(G^1(x_i^{src}))))\).
5.2. Marginal and conditional distribution adaptation

Figure 5.1 illustrates the effect of the three steps of the pipeline above on a dataset composed of digits 1 and 2 from MNIST and USPS datasets. The source dataset is indicated by stars, the target dataset is indicated by circles, red indicates class 1 and blue indicates class 2. The original space projection is generated by using the first two principal components of the source data. The effect of step (a) is to bring the mean of the two distributions closer to each other while it projects the data into its principal components directions of the full data including the source and target\(^1\). We use a marginal distribution adaptation method which relates to the works of [66, 149, 107]. This uses the empirical Maximum Mean Discrepancy proposed by Pan et al. [122] (MMD) to compare different distributions and compute a lower-dimensional embedding that minimises the distance between the expected values of samples in source and target domains. Following Long et al. [107] we adopt the MMD to compare different distributions and compute a lower-dimensional embedding that minimises the distance between the expected values of samples in source and target domains.

For the second step of our pipeline (Figure 5.1(b)), we proposed a method that distorts the source probability density function towards target clusters. We employ a sample-wise transformation that uses likelihoods of source samples given a GMM that models target data. Up to our knowledge, this is the first time a sample-based transformation is proposed for transfer learning.

In the final step (Figure 5.1(c)), the source class-conditional distributions are iteratively transformed to become more similar to their corresponding target conditionals. A related approach has been followed in [107] using pseudo-labels to iteratively update a supervised version of MMD. With the same intention, we adopt a method that uses insights from Arnold et al. [6], who used the ratio of the expected class- based posterior probability of target samples and the expected value of source samples per class. This effectively re-scales the source feature space. Our method which was fully described in Chapter 4 is a more complex transformation, as each individual feature is both scaled and translated, with different parameters per class.

\(^1\)In Figure 5.1(a), the feature space is visualised with PCA projection and only two classes are shown, while the MMD computation was done on a higher dimensional space on samples from 10 classes. For these reasons it may not be easy to see that the means of source and target samples became closer after MMD.
We additionally propose to improve our TTM pipeline by introducing a classifier selector and adapter which uses two dissimilarity measures to select a proper classifier based on the intrinsic similarities of the source domain to that of target. The measures are also used to adapt the classifier’s parameters for obtaining a better fit for target. We coin this adaptive version of TTM, Adaptive Transductive Transfer Machine (ATTM).

For transferring the knowledge of parameters in ATTM, we introduce two unsupervised dissimilarity measures which are used for selecting a proper classifier and for adapting its parameters. The first measure compares the two source and target domains of a global level for selecting the classifier while the second measure is representative of a dissimilarity at the clusters level.

The proposed four steps jointly implement a complex transformation that would be difficult to determine in a single step. The order in which the three transformations are applied, global (MDD) + sample-based (TransGrad) + conditional (TST), is important because neither MMD nor TransGrad take class labels into account. TST achieves better results if it is applied to data in which the difference between source and target domains is not too large, as it uses estimates of $P(x|y)$ based on classifiers learnt on the (adapted) source domain. If the marginals were far off the desired solution, the classifier may generate poor estimates of $P(x|y)$, leading to poor transfer. Similarly, TransGrad transformation is less constrained than MMD, which is why it is important that it is applied after MMD. These three steps complement each other, as each applies transformations of a different nature.

The effect of the model selection step is examined on both state-of-the-art JDA [107] algorithm and our TTM pipeline.

The next subsections detail each of the steps above.

5.2.1 Shared space detection with MMD

The LDA and PCA dimensionality reduction methods of Chapter 2 are originally introduced for a single distribution.
5.2. Marginal and conditional distribution adaptation

In the first step of our pipeline, we look for a shared space projection that reduces dimensionality of the data whilst minimising the reconstruction error. As explained in [107], one possibility for that is to search for an orthogonal transformation matrix \( W \in \mathbb{R}^{f' \times f'} \) such that the embedded data variance is maximised:

\[
\max_{W^TW = I} \text{tr}(W^T X H X^T W), \quad (5.1)
\]

where \( X = [X^{\text{src}}; X^{\text{trg}}] \in \mathbb{R}^{f \times n_{\text{src}} + n_{\text{trg}}} \) is the input data matrix that combines source and target samples, \( \text{tr}(\cdot) \) is the trace of a matrix, \( H = I - \frac{1}{n_{\text{src}} + n_{\text{trg}}} \mathbb{1} \) is a centering matrix where \( \mathbb{1} \) is a \((n_{\text{src}} + n_{\text{trg}}) \times (n_{\text{src}} + n_{\text{trg}})\) matrix of ones and \( f' \) is the dimensionality after the projection where \( f' \leq f \).

The optimisation problem can be efficiently solved by eigen-decomposition. However, the above PCA-based representation may not reduce the difference between source and target domains, hence the need for a more appropriate transformation remains.

Following [66, 121, 149, 107] the Maximum Mean Discrepancy (MMD) measure can be used as the distance measure to compare different distributions. This algorithm searches for a projection matrix, \( W \in \mathbb{R}^{f \times f'} \) which aims to minimise the distance between the samples means of the two distributions:

\[
\left\| \frac{1}{n_{\text{src}}} \sum_{i=1}^{n_{\text{src}}} W^T x_i - \frac{1}{n_{\text{trg}}} \sum_{j=n_{\text{src}}+1}^{n_{\text{src}}+n_{\text{trg}}} W^T x_j \right\|^2 = \text{tr}(W^T M X^T W) \quad (5.2)
\]

where \( M \) is the MMD matrix and is computed as follows:

\[
M_{ij} = \begin{cases} 
\frac{1}{n_{\text{src}}}, & x_i, x_j \in X^{\text{src}} \\
\frac{1}{n_{\text{trg}}}, & x_i, x_j \in X^{\text{trg}} \\
\frac{1}{n_{\text{src}} n_{\text{trg}}}, & \text{otherwise.}
\end{cases} \quad (5.3)
\]
The optimisation problem then is to minimise (5.2) such that (5.1) is maximised, i.e. solve
the following eigen-decomposition problem: \((\mathbf{X} \mathbf{M} \mathbf{X}^\top + \epsilon \mathbf{I}) \mathbf{W} = \mathbf{X} \mathbf{H} \mathbf{X}^\top \mathbf{W} \mathbf{D})\), obtaining
the eigenvectors \(\mathbf{W}\) and the eigenvalues on the diagonal matrix \(\mathbf{D}\). The effect is to obtain
a lower dimensional shared space between the two domains. Consequently under the new
representation \(\mathbf{W}^\top \mathbf{X}\), the marginal distributions of the two domains are drawn closer to each
other.

5.2.2 Sample-based adaptation with TransGrad

In the next step of the pipeline, we propose a sample-based transformation to refine PDF adap-
tation of source domain samples. In general, target data may, but does not have to, lie in the
same observation space. However, for the sake of simplicity, we shall assume that the transfor-
mation of the source to the target domain is locally linear, i.e. a sample’s feature vector \(\mathbf{x}\) from
the source domain is mapped to the target space by

\[ G^2(\mathbf{x}) = \mathbf{x} + \gamma \mathbf{b}_\mathbf{x}, \quad (5.4) \]

where the \(f\) dimensional vector \(\mathbf{b}_\mathbf{x}\) represents a local offset in the target domain and \(\gamma\) is a
translation regulator. In order to impose as few assumptions as possible, we shall model the
unlabelled target data, \(\mathbf{X}^{trag}\) by a mixture of Gaussian probability density functions:

\[ p(\mathbf{x}|\lambda) = \sum_{k=1}^{K} w_k p(\mathbf{x}|\lambda_k) \quad (5.5) \]

whose parameters are denoted by \(\lambda = \{w_k, \mu_k, \Sigma_k, k = 1, \cdots, K\}\) where \(w_k, \mu_k\) and \(\Sigma_k\)
denote the weight, mean and covariance matrix of Gaussian component \(k\) respectively where
\(K\) denotes the number of Gaussians and \(p(\mathbf{x}|\lambda_k) = \mathcal{N}(\mu_k, \Sigma_k)\).

We formulate the problem of finding an optimal translation parameter \(\mathbf{b}_\mathbf{x}\) as one of moving
the point \(\mathbf{x}\) to a new location \(G^2(\mathbf{x}) = \mathbf{x} + \gamma \mathbf{b}_\mathbf{x}\) to increase its likelihood as measured using
\(p(\mathbf{x}|\lambda)\).
Using the Taylor expansion, in the vicinity of $x$, the likelihood of the $p(x + \gamma b_x)$ can be expressed as:

$$p(x + \gamma b_x | \lambda) = p(x | \lambda) + \gamma (\nabla_x p(x | \lambda))^T b_x$$

(5.6)

We wish to maximise the $p(x + \gamma b_x | \lambda)$ with respect to unknown parameter $b_x$. The learning problem then can be formulated as

$$\max_{b_x} (p(x | \lambda) + \gamma (\nabla_x p(x | \lambda))^T b_x)$$

s.t. $b_x^T b_x = 1$

(5.7)

The Lagrangian of eq. 5.7 is

$$p(x | \lambda) + \gamma \nabla_x p(x | \lambda)^T b_x - \gamma' (b_x^T b_x - 1)$$

(5.8)

Setting the gradient of eq. 5.8 with respect to $b_x$ to zero

$$\nabla_x p(x | \lambda)^T - \gamma'' b_x = 0$$

(5.9)

where $\gamma''$ is considered as TransGrad’s step size parameter and is equal to $\frac{2\gamma'}{\gamma}$.

Clearly, the source data-point $x$ should be moved in the direction of maximum gradient of the function $p(x | \lambda)$. Accordingly, $b_x$ is defined:

$$b_x = \nabla_x p(x | \lambda) = \sum_{k=1}^{K} w_k p(x^{src} | \lambda_k) \Sigma_k^{-1} (x - \mu_k)$$

(5.10)

In practice, equation 5.4 translates $x^{src}$ using the combination of the translations between $x^{src}$ and $\mu_k$, weighted by the likelihood of $G^2(x^{src})$ given $\lambda_k$. 

---

5.2. *Marginal and conditional distribution adaptation*

103
5.2.3 Conditional distribution adaptation with TST

In order to adapt the class-conditional distribution mismatch between the corresponding clusters of the two domains, we used the same set of linear trans + scaling class-specific transformations of the previous chapter. To achieve this, we assume that a Gaussian Mixture Model fitted to the source classes can be adapted in a way that it matches to target classes. While the general GMM uses full covariance matrices, we follow Reynolds et al. [133] and use only diagonal covariance matrices. This way, the complexity of the estimation system becomes linear in $f$. In our experiments, we further simplify the model for this step of the pipeline by using only one Gaussian model per class which is not unrealistic considering the fact that what we are eventually interested in are compact and dense classes.

In order to adapt the class conditional distributions one can start with an attempt to match the joint distribution of the features and labels between the corresponding clusters of the two domains. However, as explained previously, labelled samples are not available in the target domain. We thus use posterior probability of the target instances to build class-based models in the target domain. We restrict our class-based adaptation method to a translation and scale transformation (abbreviated as TST). This approximation makes the computational cost very attractive.

The proposed adaptation is introduced by means of a class-based transformation $G_{y^i}(X)$ which aims to adjust the mean and standard deviation of the corresponding clusters from the source domain, i.e., each feature $j$ of each sample $x^i$ is adapted as follows

$$G_{y^i}(x^i_j) = \frac{x^i_j - E_{\text{src}}[x^i_j, y^i]}{\sigma_{\text{src}}^{x^i_j, y^i}} + E_{\text{strg}}^{y^i}[x^i_j, y^i], \forall i = 1: n_{\text{src}},$$

(5.11)

where $E_{\text{src}}[x^i_j, y^i]$ is the joint expectation of the feature $x^i_j$ and labels $y^i$, and $\sigma_{\text{src}}^{x^i_j, y^i}$ is the standard deviation of feature $x^i_j$ of the source samples labelled as $y^i$, defined by

$$E_{\text{src}}^{x^i_j, y^i} = \frac{\sum_{i=1}^{n_{\text{src}}} x^i_j \mathbb{1}_{[y^i]}(y^i)}{\sum_{i=1}^{n_{\text{src}}} \mathbb{1}_{[y^i]}(y^i)}.$$  

(5.12)
5.2. Marginal and conditional distribution adaptation

Here \( \mathbb{1}_{|y|}(y') \) is an indicator function\(^1\).

An estimation of the target joint expectation is thus formulated as

\[
E_{\text{trg}}[x_j, y] \approx E_{\text{trg}^\Lambda}[x_j, y] = \frac{\sum_{n=1}^{N_{\text{trg}}} x_j^n P_{\Lambda_{\text{src}}}(y'|x_n)}{\sum_{i=1}^{N_{\text{trg}}} P_{\Lambda_{\text{src}}}(y'_i|x_i)} \quad (5.13)
\]

and we propose to estimate the standard deviation per feature and per class using

\[
\sigma_{j,y'}^{\text{trg}} = \sqrt{\frac{\sum_{n=1}^{N_{\text{trg}}} (x_j^n - E_{\text{trg}^\Lambda}[x_j, y_i])^2 P_{\Lambda_{\text{src}}}(y'_i|X_n)}{\sum_{n=1}^{N_{\text{trg}}} P_{\Lambda_{\text{src}}}(y'_i|X_n)}}. \quad (5.14)
\]

In other words, in a common TTL problem, the joint expectation of the features and labels over source distribution, \( E_{\text{src}}[x_j, y] \), is not necessarily equal to \( E_{\text{trg}}[x_j, y] \). Therefore, one can argue that if the expectations in the source and target domains are similar, then the model \( \Lambda \) learnt on the source data will generalise well to the target data. Consequently the less these distributions differ, the better the trained model will perform.

Since the target expectation \( E_{\text{trg}^\Lambda}[x_j, y_i] \) is only an approximation based on the target’s posterior probabilities, rather than the ground-truth labels (which are not available in the target set), there is a danger that samples that would be miss-classified could lead to negative transfer. To alleviate this, we follow Arnold et al.’s [6] suggestion and smooth out the transformation by applying the following:

\[
G_{y'}^\Lambda(x_j^i) = (1 - \theta)x_j^i + \theta G_{y'}(x_j^i), \quad (5.15)
\]

with \( \theta \in [0, 1] \).

As it can be inferred from the MMD and TST equations, the effect of these two transformations is that the first is trying to find a shared subspace between the two domains to reduce the distributional mismatch in a global level while the later is actually a class specific transformation aiming to reduce the class conditional mismatch among the clusters from one domain to another.

\(^1\)Equations (5.12) and (5.13) rectify equations from [6], as we discussed in [47].
5.2.4 Iterative refinement of the conditional distribution

Matching the marginal distributions does not guarantee that the conditional distribution of the target can be approximated to that of the source. To our knowledge, most of the recent works related to this issue [21, 25, 131, 166] are Inductive TL methods and they have access to some labelled data in the target domain which in practice makes the posteriors’ estimations easier.

Instead, our class-specific transformation method (TST), reduces the difference between the likelihoods $P(G_y^s(x^{src})|y=c)$ and $P(x|y=c)$ by using the target posteriors estimated from a model trained on gradually modified source domain (eq. 5.15). Hence, these likelihood approximations will not be reliable unless we iterate over the whole distribution adaptation process and retrain the classifier model using $G_y^s(x^{src})$.

Alternative schemes for iterations are discussed in Appendix B.1.

5.2.5 Stopping criterion

In order to automatically control the number of the iterations in our pipeline, we introduce a domain dissimilarity measure inspired by sample selection bias correction techniques [11, 27]. Many of those techniques are based on weighting samples $x^{src}_i$ using the ratio $w(x^{src}_i) = \frac{P(x^{trg}_i)}{P(x^{src}_i)}$. This ratio can be estimated using a classifier that is trained to distinguish between source and target domains, i.e., samples are labelled as either belonging to class $src$ or $trg$.

Based on this idea, we use this classification performance as a measure of dissimilarity between two domains, i.e., if it is easy to distinguish between source and target samples, it means they are dissimilar. We coin this measure as Global Dissimilarity, $D_{global}(X^{src}; X^{trg})$. The intuition is that if the domain dissimilarity is high, then more iterations should be needed to achieve a better match between the domains. The intuition is that if the domain dissimilarity is high, then more iterations are required for achieving a better match between the domains. We coin this dissimilarity as Global Dissimilarity $^1$. One alternative future direction would be to check the

\[ ^1 \text{As explained in Appendix B one can replace this stopping criterion by the use of the Frobenius norm between the covariances of the transformed source matrices of two consecutive iterative steps.} \]
amount of the transformation in between two consecutive iterations and stop the iterations in case that this measure is below a specific threshold.

5.2.6 Classifier Selection and Model Adaptation

We do not assume that source and target domain samples follow the same distribution, so the best performing learner for the source set may not be the best for the target set. We propose to use dissimilarity measures between source and target sets in order to select the classifier and adjust its kernel parameter. Empirical results showed that the optimisation of SVM using grid search on the parameter space with cross-validation on the training set leads to overfitting. We therefore prefer to use Kernel LDA (KDA) [22] and PCA+NN classifiers as the main learners.

To select between these classifiers and to adapt the KDA kernel lengthscale parameter, we propose to use two measures. The first is the **Global Dissimilarity** between the source and target distributions, described in Section 5.2.5. The second measure, coined **Clusters Dissimilarity** ($D_{\text{clusters}}(X^{\text{src}}, X^{\text{trg}})$), is proportional to the average dissimilarity between the source and target clusters, computed using the average of the distances between the source class centres and their nearest target cluster center. The target clusters centres are obtained using K-means on the target data, initialised using source class centres. We therefore assume that there is no shuffle in the placement of the clusters from one domain to another.

The proposed **Clusters Dissimilarity** is similar to the cross-domain “sparse-shot” similarity of Xu *et al.* [160] which is used for multi-source motion tracking. Xu *et al.* proposed to use tracking data of object motion in each domain and compare tracks across-domain using the Kullback-Leibler Divergence (KLD) between their mixture of Gaussians (GMM) representations.

Table 5.4 shows these two measures computed on all datasets.

When both dissimilarity measures indicate that the cross-domain datasets are very different, we suggest that it is better to use a non-parametric classifier, like Nearest Neighbour, so no
optimisation is employed at training. When the two domains are similar at global levels, it is sensible to use a classifier such as KDA, whose parameters optimised on the source domain have a better chance of working on the target space. For those cases, we propose to adapt the lengthscale $\sigma$ of the RBF kernel of KDA using a linear function of the cluster dissimilarity measure. Following the common practice in the vision community (e.g. [153]), we initially set

$$\sigma = \frac{1}{n_{\text{src}}^2} \sum_{i,j} |x_i - x_j|_1, \forall x_i, x_j \in X_{\text{src}}$$

(5.16)

(we used $\ell^1$ norm in the kernel function). This is then adapted using

$$\sigma' = \sigma \times \frac{\text{const}}{D_{\text{clusters}}(X_{\text{src}}, X_{\text{trg}})}$$

(5.17)

where $\text{const}$ is empirically set to be the average cluster dissimilarity obtained in a set of cross-domain comparisons. This was devised based on the fact that the credibility of a classifier is inversely proportional to the dissimilarity between training and test samples. In the case of KDA, the best way to tune its generalisation ability is via the kernel lengthscale.

Note that the clusters dissimilarity measure can only be computed if enough samples are available in both source and target sets or if they are not too unbalanced. When these conditions are not satisfied, our algorithm avoids kernel-based method and selects the nearest neighbour classifier. The explained parameter selection and model adaptation is concluded in Table 5.1:

<table>
<thead>
<tr>
<th>$D_{\text{global}}$</th>
<th>$D_{\text{clusters}}$</th>
<th>Classifier</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\uparrow$</td>
<td>$\uparrow$</td>
<td>NN</td>
<td>$-$</td>
</tr>
<tr>
<td>$\downarrow$</td>
<td>$\downarrow$</td>
<td>KDA</td>
<td>$\sigma = \sigma_{\text{src}}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KDA</td>
<td>$\sigma' = \sigma_{\text{src}} \times \frac{\text{const}}{D_{\text{clusters}}(X_{\text{src}}, X_{\text{trg}})}$</td>
</tr>
</tbody>
</table>

In conclusion, our ATTM pipeline will use a PCA+NN classifier as its main learner model if the global dissimilarity between the two domains is high or there are not enough source samples and consequently not enough cluster-wise samples to train a highly reliable learner model or
to further adjust the classifier parameters. In any other circumstances, the model will use the KDA classifier and adjusts the kernel lengthscale.

5.2.7 The TTM algorithm and its computational complexity

The proposed TTM method is illustrated in Figure 5.2 and algorithm 1. Its computational cost is as follows, where \( n \) is the size of the dataset, \( f \) is its dimensionality and \( K \) is the number of GMM components:

1. **MMD:** \( O(n^2) \) for constructing the MMD matrix, \( O(nf^2) \) for covariance computation and and \( O(f^3) \) for eigendecomposition.

2. **TransGrad:** \( O(nK) \) for Expectation step of GMM computation, \( O(nKf) \) for the computation of covariance matrices and \( O(K) \) for the Maximization step of the GMM computation. Once the GMM is built, the TransGrad transformation itself is \( O(nKf) \).

3. **TST:** \( O(Cf) \) for class specific TST transformations.

4. **NN classifier:** zero for training and \( O(n^2f) \) for reapplying the classifier.

For each of the \( T \) iterations, the classifier is re-applied and TST is computed. Therefore, the overall complexity of our training algorithm is dominated by the cost of training a GMM (which

---

**Algorithm 1** TTM: Transductive Transfer Machine

**Input:** \( X^{src}, Y^{src}, X^{trg} \)

**Output:** \( Y^{trg} \)

1. Search for the shared subspace between the two domains (Sec. 5.2.1)
2. Adjust the marginal distribution mismatch between the two domains (Sec. 5.2.2)
3. **while** \( (T < max_{iter}) \) and \( (||D(G^t(X^{src}), X^{trg}||) > threshold) \) **do**
   3. Find the feature-wise TST transformation (eqs: 5.13, 5.14, 5.11)
   4. Transform the source domain clusters (eq. 5.15)
   5. Retrain the classifier using the transformed source
4. **end while**
is low by using diagonal covariances) and iteratively applying a classifier. The core transfor-
mations proposed in this pipeline, TransGrand and TST are $O(nKf)$ and $O(Cf)$, respectively, i.e., much cheaper than most methods in the literature.

Figure 5.2: The Transductive Transfer Machine (TTM).

Adding an additional step of Classifier selection and model adaptation to TTM, we can derive the ATTM algorithm 2:

**Algorithm 2 ATTM: Adaptive Transductive Transfer Machine**

**Input:** $X^{src}$, $Y^{src}$, $X^{trg}$

**Output:** $Y^{trg}$

1. Search for the shared subspace between the two domains (Sec. 5.2.1)
2. Adjust the marginal distribution mismatch between the two domains (Sec. 5.2.2)
3. Select the appropriate classifier (Sec. 5.2.6), if it is kernel-based, tune $\sigma$ using (5.17)

\[\text{while } T < 10 \text{ and } |D^{global}(G^f(X^{src}), X^{trg})| > \text{threshold} \text{ do}\]

4. Find the feature-wise TST transformation (Sec. 5.2.3)
5. Transform the source domain clusters
6. Retrain the classifier using the transformed source

\[\text{end while}\]

The reason why we kept the classifier selection step out of the iterative loop, is simply to reduce the computation complexity as this step requires an expensive computation of the Clusters Dissimilarity of order $O(n^2)$ (K-means clustering).

### 5.3 Experimental Evaluation

USPS, MNIST, COIL20 and Caltech+office are four benchmark datasets widely adopted to evaluate computer vision and pattern recognition algorithms. These datasets are fully described
5.3. Experimental Evaluation

in Chapter 3. We used these public databases to compare our proposed algorithms with other state-of-the-art methods.

Table 5.2 Classifiers evaluations on individual domains: 5-fold cross validation accuracy of a nearest neighbour classifier. All the datasets are $l_2$-normalized.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>MNIST</th>
<th>USPS</th>
<th>COIL1</th>
<th>COIL2</th>
<th>Caltech</th>
<th>Amazon</th>
<th>Webcam</th>
<th>DSLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA+NN</td>
<td>91.97</td>
<td>93.64</td>
<td>99.02</td>
<td>98.91</td>
<td>38.80</td>
<td>60.59</td>
<td>79.58</td>
<td>76.95</td>
</tr>
<tr>
<td>LR</td>
<td>86.15</td>
<td>89.22</td>
<td>92.36</td>
<td>92.22</td>
<td>56.27</td>
<td>72.46</td>
<td>80.01</td>
<td>67.49</td>
</tr>
<tr>
<td>KDA</td>
<td>94.05</td>
<td>94.84</td>
<td>100.00</td>
<td>99.71</td>
<td>58.16</td>
<td>78.73</td>
<td>89.54</td>
<td>63.94</td>
</tr>
<tr>
<td>SVM</td>
<td>91.80</td>
<td>95.28</td>
<td>99.72</td>
<td>99.44</td>
<td>57.17</td>
<td>74.86</td>
<td>86.44</td>
<td>75.80</td>
</tr>
</tbody>
</table>

We have evaluated the performance of a set of widely used classifiers on all the datasets based on a 5-fold cross validation mean accuracy measure. In the case of applying the NN classifier we further projected our full space into its principal components (PCA), retaining 90% of the energy. The results are presented in Table 5.2. As one can note in most of the experiments KDA is the winning classifier. SVM is also a strong learner but it requires optimisation of parameters $C$ and $\sigma$, which can make it optimal for the source domain, but not necessarily for the target. It is worth noting that PCA+NN’s performance is remarkably close to that of KDA on the first two datasets and it is even superior on the DSLR dataset.

The two cross-domain dissimilarity measures from Sections 5.2.5, 5.2.6 are shown in Table 5.4, where the datasets are abbreviated as M: MNIST, U: USPS, C: Caltech, A: Amazon, W: Webcam, and D: DSLR.

5.3.1 Experiments and Results

We coin the iterative version of different combinations of our proposed algorithms as Transductive Transfer Machine (TTM). TTM0 refers to an iterative version of TST adaptations, TTM1 is the combination of the MMD and TST and finally TTM2 is TTM1 with a sample-wise marginal adaptation (TransGrad) applied before TST (see Figure 5.2). We have also carried out experiments to show that our proposed classifier selection and model adaptation techniques (ATTM) improve the performance of both TTM and JDA algorithms significantly. We com-
Chapter 5. Transductive Transfer Machine

Table 5.3 Cross-domain dissimilarities between domains (\(src \rightarrow trg\)), with datasets abbreviated as M: MNIST, U: USPS, C: Caltech, A: Amazon, W: Webcam, and D: DSLR.

<table>
<thead>
<tr>
<th>src</th>
<th>trg</th>
<th>(D_{\text{clusters}})</th>
<th>(D_{\text{global}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>U</td>
<td>0.034</td>
<td>0.984</td>
</tr>
<tr>
<td>U</td>
<td>M</td>
<td>0.032</td>
<td>0.981</td>
</tr>
<tr>
<td>COIL1</td>
<td>COIL2</td>
<td>0.026</td>
<td>0.627</td>
</tr>
<tr>
<td>COIL2</td>
<td>COIL1</td>
<td>0.025</td>
<td>0.556</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td>0.032</td>
<td>0.548</td>
</tr>
<tr>
<td>C</td>
<td>W</td>
<td>0.033</td>
<td>0.78</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>0.031</td>
<td>0.786</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>0.031</td>
<td>0.604</td>
</tr>
<tr>
<td>A</td>
<td>W</td>
<td>0.035</td>
<td>0.743</td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>0.036</td>
<td>0.85</td>
</tr>
<tr>
<td>W</td>
<td>C</td>
<td>0.037</td>
<td>0.752</td>
</tr>
<tr>
<td>W</td>
<td>A</td>
<td>0.035</td>
<td>0.717</td>
</tr>
<tr>
<td>W</td>
<td>D</td>
<td>0.037</td>
<td>0.51</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>0.035</td>
<td>0.78</td>
</tr>
<tr>
<td>D</td>
<td>A</td>
<td>0.034</td>
<td>0.790</td>
</tr>
<tr>
<td>D</td>
<td>W</td>
<td>0.033</td>
<td>0.471</td>
</tr>
</tbody>
</table>

pared our methods with four state-of-the-art approaches [61, 107, 122, 148] using the same public datasets and the same settings as those of [107, 61]. The results are in Table 5.4. Further comparisons with other TTL methods such as Sampling Geodesic Flow (SGF) using the Grassmann manifolds [62] are reported in [61].

As one can note, all the TTL methods improve the accuracy over the baseline. Furthermore, our ATTM method generally outperforms all the other methods. The main reason for that is that our method combines three different feature adaptation techniques with a further classifier parameter adaptation step.

In most of the tasks, both TTM1,2 algorithms show comparative performance with respect to the state-of-the-art approach of JDA [107]. The average performance accuracy of the TTM1 and TTM2 on 16 transfer tasks is 55.10% and 56.20% respectively where the performance improved by 0.22% and 1.32% compared to the best performing baseline method JDA [61]. Moreover in almost all datasets TTM2 wins over TTM1 due to its initial domain dissimilarity adjustments using the TransGrad. The JDA method of Long et al. [107] also benefits from
5.3. Experimental Evaluation

Table 5.4 Recognition accuracies with datasets abbreviated as M: MNIST, U: USPS, C: Caltech, A: Amazon, W: Webcam, and D: DSLR. Comparisons start the baseline accuracy with NN and PCA followed by the results of the discussed TTL algorithms. The last two columns show the effect of the classifier selection and model adaptation techniques (5.2.6) on JDA and TTM algorithms.

<table>
<thead>
<tr>
<th>TTL test</th>
<th>NN baseline</th>
<th>PCA baseline</th>
<th>TCA [122]</th>
<th>TSL [148]</th>
<th>GFK (PLS, PCA) [61]</th>
<th>JDA (INN) [107]</th>
<th>TTM0 (TST NN)</th>
<th>TTM1 (MMD + TTM0)</th>
<th>TTM2 (Trans-Grad + TTM1)</th>
<th>AJDA (Adapt. JDA)</th>
<th>AFTM (Adapt. TTM2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M→U</td>
<td>65.94</td>
<td>66.22</td>
<td>56.28</td>
<td>66.06</td>
<td>67.22</td>
<td>67.28</td>
<td>75.94</td>
<td>76.61</td>
<td>77.94</td>
<td>67.28</td>
<td>77.94</td>
</tr>
<tr>
<td>U→M</td>
<td>44.70</td>
<td>44.95</td>
<td>51.05</td>
<td>53.75</td>
<td>46.45</td>
<td>59.65</td>
<td>59.79</td>
<td>59.41</td>
<td>61.15</td>
<td>59.65</td>
<td>61.15</td>
</tr>
<tr>
<td>COIL1→2</td>
<td>83.61</td>
<td>84.72</td>
<td>88.47</td>
<td>88.06</td>
<td>72.50</td>
<td>89.31</td>
<td>88.75</td>
<td>93.19</td>
<td>72.50</td>
<td>89.31</td>
<td>93.19</td>
</tr>
<tr>
<td>COIL2→1</td>
<td>82.78</td>
<td>84.03</td>
<td>85.83</td>
<td>87.92</td>
<td>74.17</td>
<td>88.47</td>
<td>88.89</td>
<td>88.61</td>
<td>88.75</td>
<td>92.36</td>
<td>91.11</td>
</tr>
<tr>
<td>C→A</td>
<td>23.70</td>
<td>36.95</td>
<td>38.20</td>
<td>44.47</td>
<td>41.4</td>
<td>44.78</td>
<td>39.87</td>
<td>44.28</td>
<td>46.76</td>
<td>58.56</td>
<td>60.85</td>
</tr>
<tr>
<td>C→W</td>
<td>25.76</td>
<td>32.54</td>
<td>34.24</td>
<td>40.68</td>
<td>31.69</td>
<td>31.02</td>
<td>39.66</td>
<td>41.02</td>
<td>48.81</td>
<td>62.03</td>
<td>62.03</td>
</tr>
<tr>
<td>C→D</td>
<td>25.48</td>
<td>38.22</td>
<td>41.40</td>
<td>43.31</td>
<td>41.1</td>
<td>45.22</td>
<td>50.31</td>
<td>47.13</td>
<td>45.86</td>
<td>20.32</td>
<td>20.32</td>
</tr>
<tr>
<td>A→C</td>
<td>26.09</td>
<td>34.73</td>
<td>27.76</td>
<td>37.58</td>
<td>37.9</td>
<td>39.36</td>
<td>36.24</td>
<td>35.53</td>
<td>39.62</td>
<td>40.43</td>
<td>42.92</td>
</tr>
<tr>
<td>A→W</td>
<td>29.83</td>
<td>35.59</td>
<td>37.63</td>
<td>33.90</td>
<td>35.7</td>
<td>37.97</td>
<td>37.63</td>
<td>42.37</td>
<td>39.32</td>
<td>49.83</td>
<td>50.51</td>
</tr>
<tr>
<td>A→D</td>
<td>25.48</td>
<td>27.39</td>
<td>33.12</td>
<td>26.11</td>
<td>36.31</td>
<td>39.49</td>
<td>33.75</td>
<td>29.30</td>
<td>29.94</td>
<td>38.21</td>
<td>39.49</td>
</tr>
<tr>
<td>W→C</td>
<td>19.86</td>
<td>26.36</td>
<td>29.30</td>
<td>29.83</td>
<td>29.3</td>
<td>31.17</td>
<td>26.99</td>
<td>29.83</td>
<td>30.36</td>
<td>35.80</td>
<td>34.02</td>
</tr>
<tr>
<td>W→A</td>
<td>22.96</td>
<td>31.00</td>
<td>30.06</td>
<td>30.27</td>
<td>35.5</td>
<td>32.78</td>
<td>29.12</td>
<td>30.69</td>
<td>31.11</td>
<td>38.94</td>
<td>39.67</td>
</tr>
<tr>
<td>W→D</td>
<td>29.23</td>
<td>37.07</td>
<td>87.26</td>
<td>87.26</td>
<td>80.89</td>
<td>89.17</td>
<td>85.98</td>
<td>89.17</td>
<td>89.17</td>
<td>89.17</td>
<td>89.17</td>
</tr>
<tr>
<td>D→C</td>
<td>26.27</td>
<td>29.65</td>
<td>37.10</td>
<td>28.50</td>
<td>30.28</td>
<td>31.52</td>
<td>29.65</td>
<td>31.25</td>
<td>32.06</td>
<td>28.31</td>
<td>32.41</td>
</tr>
<tr>
<td>D→A</td>
<td>28.50</td>
<td>32.05</td>
<td>32.15</td>
<td>27.56</td>
<td>36.1</td>
<td>33.09</td>
<td>31.21</td>
<td>29.75</td>
<td>30.27</td>
<td>37.47</td>
<td>38.73</td>
</tr>
<tr>
<td>D→W</td>
<td>63.39</td>
<td>75.93</td>
<td>86.10</td>
<td>85.42</td>
<td>79.1</td>
<td>89.49</td>
<td>85.08</td>
<td>90.84</td>
<td>88.81</td>
<td>89.49</td>
<td>88.81</td>
</tr>
<tr>
<td>Avg</td>
<td>43.06</td>
<td>49.23</td>
<td>50.35</td>
<td>52.34</td>
<td>50.00</td>
<td>54.88</td>
<td>54.12</td>
<td>55.10</td>
<td>56.20</td>
<td>59.17</td>
<td>60.72</td>
</tr>
</tbody>
</table>

Jointly adapting the marginal and conditional distributions but their approach has the global and class specific adaptations along each other at each iteration which in practice might cancel the effect of each other, limiting the final model from being well fitted to the target clusters. While in JDA [61] the number of iterations is a predefined constant, in our algorithm we based this number on a sensible measure of domain dissimilarity described earlier. Moreover, the proposed TTM guarantees an acceptable level of performance about five times faster than the best performing state-of-the-art approach. GFK performs well on some of the Office+Caltech experiments but poorly on the others. The reason is that the subspace dimension should be small enough to ensure that different sub-spaces transit smoothly along the geodesic flow, which may not be an accurate representation of the input data. JDA and TTM perform much better by learning a more accurate shared space.

Further results showing the effect of the proposed classifier selection and model adaptation
techniques are also encompassed in Table 5.4. We have tested our proposed methodologies on both JDA and TTM algorithms as AJDA and ATTM. Their enhanced performance shows that the model adaptation drastically enhances the final classifier. One should note that in the cases where our model adaptation technique selects the NN classifier as the main learner of the algorithm, the results remain steady. The performance gains of 4.59% and 4.29% in ATTM and AJDA respectively validate the proposed dissimilarity measures for model selection and adaptation. The proposed model adaptation step of the pipeline selected the Nearest Neighbour classifier for MNIST→USPS and for DSLR→Webcam. For all other transfer problems, KDA was chosen and $\sigma$ adaptation was used.

After developing our MMD-based algorithm, we came across alternative subspace projection methods [9, 51]. In [9] the author proposes the Domain Invariant Projection (DIP) where
a Grassmann manifold latent subspace is used to project the data and the MMD measure is subsequently applied for evaluating the source and target domains dissimilarity. The goal is to find a representation of the data that is invariant across different domains. Alternatively, they propose another projection that not only minimises the distance between the distribution of the projected source and target, but also yields good classification performance (DIP-CC). The algorithm searches for a projection that encourages samples with the same labels to form a more compact cluster which is achieved by minimising the distance between the projected samples of each class and their mean.

In [51], the authors exploit the global covariance statistical structure of the two domains during the adaptation process in contrast to the manifold alignment methods that use local statistical structure of the data [16, 17, 18]. They project the source data onto the source subspace and the target data onto the target subspace in contrast to most domain adaptation methods in literature. The Subspace Alignment Domain Adaptation (SADA) method is totally unsupervised and does not require any target labels. SADA makes use of the correlated features in both domains where some of these features can be specific to one domain yet correlated to some other features in the other one allowing the method to use both shared and domain specific features.

In Table 5.5 we compare these state-of-the-art latent subspace detection methods (DIP, DIP-CC and SADA) with the MMD-PCA based method which we used in our transductive transfer machine framework. As one can note these methods outperform our MMD-PCA subspace projection method at the cost of a higher computational complexity. Considering the additional steps in our full TTM pipeline (TRansGrad and TST), one can realise the importance of applying a lower cost subspace detection. However, all the above mentioned subspace detection methods can replace the first step of our pipeline (MMD-PCA shared space detection) and potentially improve the final classification performance.

Following the same experimental setting, we present further results for Caltech+office dataset. The difference with the previous experiments is that instead of SURF features we use the Deep Convolutional Activation Features (DeCAF) [41], which have shown promising performance for object recognition (see Section 2.2.3 for theoretical details on DeCAF features). Follow-
Table 5.5 Recognition accuracies with NN classifiers on target domains using different shared subspace projection methods.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C→ A</td>
<td>50</td>
<td><strong>51.8</strong></td>
<td>39</td>
<td>46.14</td>
</tr>
<tr>
<td>C→ W</td>
<td>47.6</td>
<td><strong>47.7</strong></td>
<td>36.8</td>
<td>37.97</td>
</tr>
<tr>
<td>C→ D</td>
<td>49</td>
<td><strong>51.4</strong></td>
<td>39.6</td>
<td>45.86</td>
</tr>
<tr>
<td>A→ C</td>
<td><strong>43.3</strong></td>
<td>43.2</td>
<td>35.3</td>
<td>40.61</td>
</tr>
<tr>
<td>A→ W</td>
<td>46.7</td>
<td><strong>47.8</strong></td>
<td>38.6</td>
<td>40</td>
</tr>
<tr>
<td>A→ D</td>
<td>42.8</td>
<td><strong>43.3</strong></td>
<td>37.6</td>
<td>31.85</td>
</tr>
<tr>
<td>W→ C</td>
<td>37</td>
<td><strong>37.1</strong></td>
<td>32.3</td>
<td>31.26</td>
</tr>
<tr>
<td>W→ A</td>
<td><strong>42.5</strong></td>
<td>41.1</td>
<td>37.4</td>
<td>31.94</td>
</tr>
<tr>
<td>W→ D</td>
<td>86.4</td>
<td>85.3</td>
<td>80.3</td>
<td><strong>89.17</strong></td>
</tr>
<tr>
<td>D→ C</td>
<td><strong>39</strong></td>
<td>35.8</td>
<td>32.4</td>
<td>33.39</td>
</tr>
<tr>
<td>D→ A</td>
<td>40.5</td>
<td><strong>41</strong></td>
<td>38</td>
<td>31.21</td>
</tr>
<tr>
<td>D→ W</td>
<td>86.7</td>
<td>84.02</td>
<td>83.6</td>
<td><strong>87.46</strong></td>
</tr>
</tbody>
</table>

In [93], we used the outputs from the 6th layer as the visual features, leading to 4,096-dim DeCAF6 features. In this set of experiments we compare our TTM and ATTM methods with methods which aim to solve the transductive transfer learning task by adapting the classifiers hyperplanes or by means of auxiliary classifiers, namely: the Adaptive Support Vector Machines (SVM-A) [162], Domain Adaptation Machine (DAM) [43] and DA-M2S [94].

In [43], the author proposed a multiple source domain adaptation method referred to as DAM by leveraging a set of pre-learnt classifiers independently learnt with the labeled patterns from multiple source domains. Specifically, DAM was introduced as a data dependent regulator based on smoothness assumption into Least-Squares SVM (LS-SVM), which enforces that the target classifier shares similar decision values with the auxiliary classifiers from relevant source domains on the unlabeled patterns of the target domain. For a single source domain scenario, the experiments were repeated 10 times by using randomly generated subsets of source and target domains and the mean performance is reported in Table 5.6. The DA-M2s method of [94] is an extension of the DAM method where from each RGB image data two nonlinear features are extracted, one describing the depth information and the other containing visual information. Using the Kernel Canonical Correlation Analysis (KCCA), the correlation between these two
types of features is maximized. The two described state-of-the-art methods are constructed on an adaptive SVM (SVM-A) classifier method [162]. The SVM-A proposes a general framework to adapt one or more existing classifiers of any type to a new target dataset.

### Table 5.6 Results on Caltech+office dataset using DeCAF features.

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>SVM-A [162]</th>
<th>DAM [43]</th>
<th>DA-M2S (w/o depth) [94]</th>
<th>JDA (1NN) [107]</th>
<th>TTM (NN)</th>
<th>ATTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>C→ A</td>
<td>85.70</td>
<td>83.54</td>
<td>84.73</td>
<td>84.27</td>
<td>89.77</td>
<td>89.98</td>
<td>92.17</td>
</tr>
<tr>
<td>C→ W</td>
<td>66.10</td>
<td>81.72</td>
<td>82.48</td>
<td>82.87</td>
<td>83.73</td>
<td>86.78</td>
<td>90.84</td>
</tr>
<tr>
<td>C→ D</td>
<td>74.52</td>
<td>74.58</td>
<td>78.14</td>
<td>75.83</td>
<td>86.62</td>
<td>89.17</td>
<td>92.99</td>
</tr>
<tr>
<td>A→ C</td>
<td>70.35</td>
<td>74.36</td>
<td>76.60</td>
<td>78.11</td>
<td>82.28</td>
<td>83.70</td>
<td>86.55</td>
</tr>
<tr>
<td>A→ W</td>
<td>64.97</td>
<td>70.58</td>
<td>74.32</td>
<td>71.04</td>
<td>78.64</td>
<td>89.81</td>
<td>89.15</td>
</tr>
<tr>
<td>A→ D</td>
<td>57.29</td>
<td>96.56</td>
<td>93.82</td>
<td>96.62</td>
<td>80.25</td>
<td>81.36</td>
<td>90.45</td>
</tr>
<tr>
<td>W→ C</td>
<td>60.37</td>
<td>85.37</td>
<td>87.88</td>
<td>86.38</td>
<td>83.53</td>
<td>80.41</td>
<td>83.44</td>
</tr>
<tr>
<td>W→ A</td>
<td>62.53</td>
<td>96.71</td>
<td>96.31</td>
<td>97.12</td>
<td>90.19</td>
<td>88.52</td>
<td>92.27</td>
</tr>
<tr>
<td>W→ D</td>
<td>98.73</td>
<td>78.14</td>
<td>81.27</td>
<td>77.60</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>D→ C</td>
<td>52.09</td>
<td>91.00</td>
<td>91.75</td>
<td>91.37</td>
<td>85.13</td>
<td>82.90</td>
<td>82.28</td>
</tr>
<tr>
<td>D→ A</td>
<td>62.73</td>
<td>76.61</td>
<td>79.39</td>
<td>78.14</td>
<td>91.44</td>
<td>90.81</td>
<td>91.65</td>
</tr>
<tr>
<td>D→ W</td>
<td>89.15</td>
<td>83.89</td>
<td>84.59</td>
<td>83.31</td>
<td>98.98</td>
<td>98.98</td>
<td>98.98</td>
</tr>
<tr>
<td>Avg</td>
<td>70.33</td>
<td>83.95</td>
<td>84.06</td>
<td>84.97</td>
<td>87.55</td>
<td>87.87</td>
<td>90.90</td>
</tr>
</tbody>
</table>

Note that in Table 5.6 the baseline without any transformation using DeCAF features and NN classifier is significantly better than the results of Table 5.4, simply because DeCAF features are better than SURF. As one can see our TTM and ATTM methods are both outperforming the other state-of-the-art approaches in most of the cases gaining 2.9% and 5.96% average performance enhancements over the best performing state-of-the-art method of DA-M2S(w/o depth) respectively. One should note that in both state-of-the-art approaches, DAM[43] and DA-M2S[94] the model have access to a few number of labelled samples from the target domain while our model is not benefitting from this.

We further study the possibility of correlations between **Global Dissimilarity** and averaged accuracy enhancement after applying the ATTM and also between **Clusters Dissimilarity** and averaged accuracy enhancement and plotted the results in Figure 5.5 and 5.6.

As one can note, no clear conclusion can be derived on availability of correlations between the two similarity measures and the averaged mean accuracy enhancements.

To validate that TTM can achieve an optimal performance under a wide range of parameter val-
Chapter 5. Transductive Transfer Machine

Figure 5.4: Bar chart demonstrating the same results of Table 5.6

ues, we conducted sensitivity analysis on MNIST→USPS, Caltech→Amazon and Webcam→Caltech. We ran TTM with varying values of the regulator $\gamma''$ of the TransGrad step, and the results are in figure 5.7(a). One can see that for all these datasets, the performance improves as $\gamma''$ grows but it plateaus when $\gamma'' = 5$. For this reason we used $\gamma'' = 5$ in all experiments of this chapter.

We also ran TTM with varying number Gaussians $K$ in the TransGrad step for the target GMM. Theoretically as the number of GMM components increases the translations get more accurate and the performance becomes more stable. We plot the classification accuracy w.r.t. $K$ in figure 5.7(b). Note that for $K = 1$, TransGrad contributes to an improvement over the baseline, as it induces a global shift towards the target set. But in general, for values of $K$ smaller than the number of classes, we do not actually expect TransGrad to help, as it will shift samples from different classes towards the same clusters. This explains why the performance increases with $K$ for $K > 2$. Based on this result, we adopted $K = 20$ in all other experiments of this paper.
5.3. Experimental Evaluation

We have also compared the time complexity of our TTM algorithm against JDA [107] in the transfer task from MNIST digits dataset to USPS digits dataset. Both algorithms were implemented in Matlab and were evaluated on a Intel Core2 64bit, 3GHz machine running Linux. We averaged time measurements of 5 experiments. The JDA algorithm took $21.38 \pm 0.26$ seconds and our full TTM framework took $4.42 \pm 0.12$ seconds, broken down as: $0.40 \pm 0.01$ seconds to find the appropriate shared space using the MMD, $1.90 \pm 0.06$ to perform the sample-wise marginal distribution adaptations using TransGrad and finally $2.42 \pm 0.12$ seconds to apply the iterative conditional distribution adaptations (TST). The time complexity obviously will grow as for AJDA and ATTM due to kernel computation of the KDA classifier.

Figure 5.5: $D_{clusters}$ vs. averaged accuracy enhancement over the PCA baseline graph for Table 5.3 transfer tasks.
In this chapter, we introduced transductive transfer machines (TTM), which aim to adapt both the marginal and conditional distributions of the source samples so that they become more similar to those of target samples, leading to an improvement in the classification results in transfer learning scenarios. TTM’s pipeline consists of the following steps: first, a global linear transformation is applied to both source and target domain samples, so that their expected values are matched. Then we proposed a novel method that applies a sample-based transformation to source samples. This leads to a finer adaptation of their marginal distribution, taking into account the likelihood of each source sample given the target PDF. Finally, we proposed to iteratively adapt the class-based posterior distribution of source samples using an efficient linear transformation whose complexity only depends on the number of features. In addition, we proposed to use two unsupervised similarity measures, Global and Clusters Dissimilarities.
5.4. Summary

(a) TransGrad $\gamma''$

(b) TransGrad GMM components’ size

Figure 5.7: The effect of different $\gamma''$ values and number of GMM clusters in the TransGrad step of our framework on the final performance of the pipeline for three cross-domain experiments. Constant lines show the baseline accuracy for each experiment.

The former is used both to automatically determine the number of iterations needed and also to select the pipeline’s main learner model. The later measure, Clusters Dissimilarity, is used for adjusting the classifier’s parameters for the new target domain.

Our approaches outperformed state-of-the-art methods on various datasets, with a lower computational cost.
Chapter 6

Conclusions and Future Work
Chapter 6. Conclusions and Future work

6.1 Conclusion

By analysing the current state-of-the-art classification methods, it is clear that many machine learning approaches based on a large amount of samples reach impressive results on difficult datasets \[44, 126\]. However, most of them traditionally address each learning task in isolation and provide no guarantees when only a small amount of training samples is available, or more in general, if there is a mismatch between the training and the testing distributions \[127, 151\].

Ideally, we would like to build image classifiers that can exploit a large number of complex features. Working with such reach representation comes at an added cost: Training might require a large amount of supervised data but only a few labeled examples might be available for a given task. The lack of training samples subsequently results in a non-generative model for a new image test data of different quality. The problem becomes even more challenging when the intrinsic distribution of this test data differs from that of training and no labels are available for a guided adaptation. To address this problem we developed principled methods capable of exploiting prior knowledge on similarity of the domains and boosting the learning process by searching for a sequential self-controlled source feature transformations given a new unlabelled target data. In particular this thesis makes the following contributions:

- **Class Conditional Distribution Adaptation via Feature Space Transformations:**

  We introduced class-specific posterior adaptation transformations in Chapter 4 where the source domain features are re-weighted based on the ratio of the joint expectation of features and class labels in target and source domains. We then proposed a modification for a more complex transformation, based on translation and scaling of each feature, for each class label.

  Our results show that the proposed class-conditional adaptation is successful in capturing the correct transformation from the training space for a given unlabelled test data.

- **Marginal Distribution Adaptation via Local Transformations Using Target GMM Gradient Directions:**
To adapt the marginal distribution of source and targets domains, we proposed fine local transformations. The proposed sample-wise transformations use the likelihood of the source samples given the GMM model of the target distribution. The integration of this transformation with the Maximum Mean Discrepancy dimensionality reduction showed that the proposed local transformations are efficient in remodelling the global source distribution to one closer to the target distribution (Chapter 5).

- **Effective Unsupervised Measures for Cross-Domain Dissimilarity Evaluation:**

  Two unsupervised dissimilarity measures were introduced in Chapter 5 for evaluating the source and target distribution mismatch. The first measure, derived from sample-selection bias correction techniques, describes the global dissimilarity of the two domains and the second is representative of the average dissimilarity between the corresponding clusters of the two domains. The proposed measures help in facilitating the automation of a novel transductive transfer algorithm, Adaptive Transductive Transfer Machine.

- **Adaptive Transductive Transfer Machine, an Efficient Pipeline for Transductive Transfer Learning:**

  Finally we proposed the time efficient Transductive Transfer Machine pipeline (Chapter 5) which integrates our proposed transformations for a comprehensive adaptation. The algorithm first searches for a global transformation such that the marginal distributions of the two domains become more similar and then with the same objective applies a set of local transformations to each transformed source domain sample. Finally in an iterative scheme, TTM aims to reduce the difference between the conditional distributions in source and target spaces where a class-based transformation is applied to each of the transformed source samples. The complexity of the latter iterative step in the pipeline is linear on the number of features in the space, i.e., $O(f)$.

  The pipeline was additionally equipped with two unsupervised measures which automatically control the necessity of sequential transformations and number of iterations.
The two measures were further used for TTM’s model adaptation which resulted in an adaptive version of TTM, Adaptive Transductive Transfer Machine (ATTM).

- **Transfer Learning with Transductive Transfer Forest:**

  We introduced transductive random forest in Appendix A by proposing modifications to the tree node optimisation criteria. We enhanced the model further by introducing node-level cross-domain mean and variance adaptations.

  Although the synthetic experimental evaluations showed a successful performance and the proposed methods outperformed the basic random forest classifier in a wide range of experiments, the proposed method failed in competing with the other state-of-the-art methods in terms of time complexity and performance. Experiments of combining the proposed transductive random forest with feature space transformations method of Chapter 4 confirmed the incompatibility of the random forest based models with feature transformation adaptations.

### 6.2 Future Work

**Feature Enhancement:** As discussed earlier the domain adaptation problems can be solved in two basic ways: *Feature Enhancement* where the aim is to transform the feature spaces so that two source and target distributions better match each other and *Model Adaptation* methods which adapt the trained model on source to better fit a new target data. In our proposed ATTM algorithm we partially tried to combine these two solutions.

It is worth pointing out that both TTM and ATTM are general frameworks with applicability beyond object recognition and could be easily applied to other domains, even outside Computer Vision.

For future work, we suggest studying combinations of TTM with semi-supervised learning methods and feature learning algorithms. Another exciting direction is to combine TTM with voting classification algorithms for problems with multiple source domains.
The use of an ensemble of classifiers and clusterers to generate a more consolidated classifier which has been introduced in a couple of recent works [1], [57], [111] can be thought of as a Model Adaptation alternative to our proposed classifier selector and adaptation solution of ATTM. The rationale behind Gao et al.’s [57] method is that classifiers can be built using different learning algorithms on the same source domain and hence due to the inductive bias of the specific learning technique or the distributional differences among the available different training domains, they can achieve different knowledge resulting in different advantages in learning. Thus the idea is to combine knowledge from these classification models by assigning per model and per example weights to maximise their combined accuracy on the new target domain. To do so the approach takes advantage of the fact that target samples in a same cluster are more likely to share the same class labels. In this method if none of the mapped local structures are similar to the original local structure in the target domain, the predicted label is obtained by voting among its neighbours inside the same local structure of the test set. We propose that instead of doing so, under these circumstances, it is better to rely on our proposed TTM (or ATTM) feature enhancement method proposed in Chapter 5. These probability distributions could then be refined with the help of a cluster ensemble which provides supplementary constraints for classifying the test objects. But if no appropriate classifier model is found then we rely on the result of the method proposed in Chapter 5.

To decide between the two sets of class label estimations, that of the method in Chapter 5 and that of [57], we need to assign weights according to their reliability using a validation set.

The block diagram of the system to be developed can be seen in Figure 6.1. The top section of the diagram which provides the set of labels $Y_1$, shows the steps of the Gao et al.’s [57] method and the bottom section is a framework for our method that produces the set of labels $Y_2$ and finally a thresholding step deciding the final label set $Y_f$. 
Chapter 6. Conclusions and Future work

The future direction for the proposed Transductive Transfer Forest of Appendix A can be to use it for a new class discovery task. This can be done by increasing the weight of the unsupervised information gain in the node split criteria eq. A.15 and additionally removing the restriction in the node expansions which results in leaves containing unlabeled samples (probably of new class) which are far away from the labeled distributions. Alternatively one can follow the forest-based lower-dimensional subspace detection of Rodner et al. [135] and use the proposed TRF of Appendix A to extract the features which are selected for node split optimisation as the subspace bases.
Appendix A

Transductive Random Forest
This appendix is focusing on an attempt to perform transductive transfer learning with random forest. As the attempt proved to be unsuccessful, the discussion has been relegated to this Appendix. However, in spite of the negative results, the work is considered as a contribution, as some of the findings about the suitability of the random forest for the transductive transfer learning are believed to be quite general. Section A.1 describes the related work. We then review the background for decision trees in Section A.2 and propose modifications to the Criminisi et al.’s [30] semi-supervised random forest in Section A.3 to make it more generative for domain adaptation tasks. Semi-supervised random forest are discussed in Section A.4. In Section A.5, we combine the feature space class-conditional transformations of Chapter 4 with this modified semi-supervised RF which we call it Transductive Random Forest, TRF. In this scenario the training data is composed of a set of labeled transformed source samples and a set of unlabeled target data. In Section A.6 we introduce a novel adaptive version of the TRF, ATRF. Experimental results demonstrating the efficacy of the proposed method are presented in Section A.7 which is followed by a short summary in Section A.8.

A.1 Related Work

Classifier ensembles have already been considered for Transfer Learning in [112, 2, 81, 34]. Of particular interest in this appendix are the methods which are based on Decision Trees. The most well-known randomised ensemble of decision trees known as Random Forest (RF), has been shown to achieve a successful performance in vision applications [72, 73, 30].

Random Forest seems to have been first introduced by T.K. Ho for handwritten digit recognition [70] and the usage has been extended for a wide range of machine learning applications including semi-supervised learning proposed by Criminisi et al. [30] in 2011.

Kamishima et al. [84] applied a bagging approach for transferring the learning capabilities of a model to different domains. In their work, a high number of trees was learned on data from both source and target domains and a pruned version of the final ensemble was used to predict examples of the target domain. The pruning step was used in order to avoid negative
transfer. Although the authors used training examples from both source and target domains, no experiments were conducted to quantify the amount of data needed to achieve a reasonable accuracy of the transferred classifier.

Rodner et al. [135] proposed to use the random decision forests for transferring feature relevance. In their inductive transfer approach, in order to estimate the underlying feature relevance with a few labeled samples available in the target space, they first train the decision forest using all the labeled examples of either source and target sets. The next step is to count the times a specific feature is used for optimisation of a split node of each of the trees in the forest. The idea is that the feature with the high occurrence is likely to be relevant for a given transfer task.

While most of the works on adapting RF for transfer learning are focused on inductive transfer (a few labeled target data is available for induction), in this appendix we propose modifications to RF facilitate transductive transfer learning.

A.2 Decision Trees

Decision trees are model combination techniques typically used for classification. They have been around since 1970 and their recent revival is due to the discovery that ensemble of randomly trained trees tend to produce much higher accuracy on previously unseen data. This approach known as random decision forest will be discussed in more details in Section A.3.

In this section we only focus on individual tree training. The underlying idea of decision tree used for classification is that they recursively split the input space into subregions based on their node split functions known as the weak learners. As its name infers, decision tree is a tree used for making decisions.

For instance, imagine we have a simple dataset presented as a collection of the points in 2D space with two features TI and PE (Figure A.1(a)). We can start by testing first the TI feature and based on its value decide to proceed either to Left or Right child node (see Figure A.1(b)). At the next tree depth, the other descriptor (PE) can be used in a similar way and so on until we
reach a leaf node. The full training involves sending all training data into the tree for optimising the parameters of the split node functions.

<table>
<thead>
<tr>
<th>TI</th>
<th>PE</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>M2</td>
<td>good</td>
</tr>
<tr>
<td>2.0</td>
<td>M1</td>
<td>bad</td>
</tr>
<tr>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>4.5</td>
<td>M5</td>
<td>?</td>
</tr>
</tbody>
</table>

(a) Simple dataset with two descriptors: TI and PE (figure reproduced from [116].)

(b) Greedy, recursive partitioning along the data descriptors (based on illustration from [116]).

Figure A.1: (a) Input data represented as a collection of the points in a 2-dimensional space. (b) During the classification process, a split (internal) node applies a test to a given input data and sends it to the appropriate child node. The process is repeated until a leaf (terminal) node is reached. Training a decision tree involves sending all training data into the tree for optimising the parameters of the split node functions.

As with any classifier, at a high level, the functioning of a decision tree can be separated into an off-line (training) phase and an on-line one (testing) [30].

A.2.1 Tree training (off-line)

The aim of the off-line, training phase is to optimise parameters of the split functions associated with all the internal split nodes as well as the leaf predictors [30]. For this optimisation, a
A.2. Decision Trees

A popular measure is information gain which is based on entropy to achieve a better class separation. In other words we need to define a criteria that guarantees that each node-split achieves a better class separation in children nodes \textit{i.e.} the two resulting children subsets have lower entropy than their parent node which results in information gain.

![Figure A.2: Tree testing (runtime). Diagram reproduced from [30].](image)

**The weak learner model**

Each split node in a tree is associated with a binary split function:

\[
h(x, \theta) \in \{0, 1\} \tag{A.1}
\]

with 0 and 1 indicating \textit{false} and \textit{true} respectively. The weak learner model of \(j^{th}\) node in the tree is characterized with its parameter \(\theta_j = (\Phi_j, \psi_j, \tau_j)\) where \(\psi_j\) defines the geometric primitive used to separate the data (the type of the hyperplane or surface that is used for separation purpose at internal node levels), \(\tau_j\) is the threshold for inequalities used in the binary test and finally the parameter \(\Phi_j\) which functions as a filter to select a subset of the features used for node splitting.
The training objective function

During the training process the aim is to find the optimal parameter \( \theta^* \) for all the split nodes. This is done by maximising an information gain objective function:

\[
\theta_j^* = \arg \max_{\theta_j} I_j \tag{A.2}
\]

where

\[
I_j = I(X_j, X_j^L, X_j^R, \theta_j) \tag{A.3}
\]

The symbols \( X_j, X_j^L \) and \( X_j^R \) are the sets of the points before the split and the subsets sent to Left and Right child nodes respectively.

A common choice to compute the information gain in a classification task is the Shannon entropy [146] which is a measure describing how appropriate are the class distributions within a set of labeled samples. Mathematically the Shannon entropy of a set of labeled samples \( X \) is defined as:

\[
H(X) = - \sum_c V(X|y = c) \log V(X|y = c) \tag{A.4}
\]

where \( V(X|y = c) \) is the cardinality of the set of samples labeled as \( c \) divided by the total number of labeled samples.

Consequently, the information gain of node \( j \) using the Shannon entropy can be formulated as:

\[
I_j = H(X_j) - \sum_{i \in \{L,R\}} \frac{|X_j^i|}{|X_j|} H(X_j^i) \tag{A.5}
\]

By maximising the above information gain equation with respect to \( j \), a better class separation is obtained via the split in node \( j \) in which the sum of the children sets entropies is lower than their parent set.
Stopping criteria

Another aspect in training the decision tree is the choice of when to stop growing individual tree branches. One of the common criteria is to define a maximum depth $D$. Alternatively, one can impose a minimum information gain or even a limitation over the minimum number of the available training data for training an individual tree node split.

A.2.2 Tree testing (runtime)

Given a previously unseen data point $x$ a trained decision tree hierarchically applies the learnt node split functions and depending on the result of each one of the binary tests the data is sent to the right or left child (see Figure A.2). The process is repeated until a leaf node is reached. Usually the leaf node contains a predictor which associates either a class label or a class associated probability value to the input $x$.

The leaf prediction model

During training, information that is useful for the prediction of labels for the unlabeled test set will be learned for all leaf nodes. The assumption is that each leaf stores the empirical distribution over the classes associated to the subset of the training data that has reached that leaf. The probabilistic leaf predictor model for the tree is then $p(y = c | x)$ with $y \in \{Y\}$ indexing the classes e.g. $p(y = c | x \in X_{Leaf}) = V(x \in X | y = c)$.

A.3 Random Forest

A Random decision forest, is an ensemble of randomly trained decision trees. The ensemble model in practice prevent a single tree from over-fitting the training data. The idea is first introduced by T.K. Ho for handwritten digit recognition [70] and the usage has been expanded for a wide range of machine learning applications [30]. Random Forests (RF) have become
commonplace in many computer vision applications. Their popularity is mainly driven by their high computational efficiency during both training and evaluation while still being able to achieve state-of-the-art accuracy.

In this section we explain the theory of the RF.

A.3.1 The training objective function of Random Forest

While the training objective function of the trees within the RF architecture has the same form of a single tree model (eq. A.5), the probabilistic leaf predictor model for the $t^{th}$ tree in the forest of $T$ trees can be modeled as $p_t(y|x)$ with $y \in \{Y\}$ indexing the classes.

A.3.2 The randomness model

The key aspect of decision forest is the fact that its component trees are all randomly different from one another [30]. This improves their generalisation power over a single tree model. The randomness can be injected into the trees during the training phase via either of the following ways:

1. randomly choosing a subset of the training data for training individual trees

2. random optimisation of the nodes.

While the first randomness technique is implicit, the second technique basically means that if $\Theta$ is the entire set of all possible parameters $\theta$ then when training the $j^{th}$ node we only make available a subset $\Theta_j \subset \Theta$ of such values [30]. The second randomness insertion approach is more precise in a sense that we use all the training samples and take into account the whole input data characteristics inducing margin-maximisation but one should note that these two techniques are not mutually exclusive and can be used together.

In the case of random optimisation of nodes, two typical randomized forest architectures have been suggested. The first is called the Random Forest and the second is named Extremely
A.4. Semi-supervised Random Forest

Randomised Forest. As for the first forest, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set (item 1 above). In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features but the best split among a random subset of the features (item 2 above). As a result of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree which results in overfitting problem) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model [72].

In Extremely Randomised Forest, randomness goes one step further in the way splits are computed. As in Random Forest, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule (an alternative approach for item 2 above proposed in [72]).

A.3.3 The ensemble model: RF

Once the split node functions are optimised and all the trees in a RF are trained, a general probabilistic classification rule \( p(y|x) \) needs to be described that can be applied to previously unseen test input for label inference. For the RF classifier where after training the trees we have only labeled samples in the leaves, the probabilistic leaf predictor model of the \( t \)th tree in the forest of \( T \) trees can be described as \( p_t(y|x) \) with \( y \in \{Y\} \) indexing the classes. Consequently, the ensemble model can be formulated as:

\[
p(y|x) = \frac{1}{T} \sum_{t=1}^{T} p_t(y|x)
\]

(A.6)

A.4 Semi-supervised Random Forest

The extension to the usage of Random Forests to Semi-Supervised Learning problems have been proposed recently [105, 30]. Leistner et al. [105] developed a multi-class margin defi-
In this section we briefly introduce the Semi-supervised Random Forest (SSRF).

A.4.1 The Training objective function for Semi-supervised Random Forest

In the case of the Semi-supervised random decision forest (SSRF), the training data is composed of a combination of the labeled ($x^s \in S$) and unlabeled samples ($x^u \in U$) where we are interested in transferring existing ground-truth labels to the unlabelled (and already available) data. Therefore, the objective function $I_j$ must encourage both class separation of the labeled training data as well as separating different high density regions from one another (accounting for unlabeled test samples). This is achieved by introducing an extra information gain term which corresponds to each node’s unlabeled data distribution:

$$I_j = I_j^{U\cup S} + I_j^S$$  \hspace{1cm} (A.7)

In the above equation $I_j^S$ is a supervised term that depends only on the labeled training data which can be computed using eq. A.5. On the other hand $I_j^{U\cup S}$ stands for the unsupervised term and depends on all data ($U \cup S$). The existence of the unlabeled training set prompts us to define an unsupervised entropy which does not depend on the labels voting. To define this entropy Criminisi [30] suggested to assume that the distribution of the samples in each node can be modeled with a Gaussian function. Since the determinant of the covariance function of an f-variate Gaussian is proportional to the volume of the ellipsoid containing the data, it can be a good estimate of the data density. This implies that the unsupervised differential entropy of a set of samples ($X$) can be represented as:

$$H(X) = \frac{1}{2} \log((2\pi e)^f|\Lambda(X)|)$$  \hspace{1cm} (A.8)
A.4. Semi-supervised Random Forest

where \( \Lambda \) is the associated \( f \times f \) covariance matrix and \( |\Lambda| \) is its determinant. Consequently, the unsupervised gain of node \( j \) containing the set of samples \( X_j \) can be defined via differential entropies over continuous parameters:

\[
I_{U \cup S}^U = \log |\Lambda(X_j)| - \sum_{i \in \{L,R\}} \frac{|X_i|^j}{|X_j|} \log |\Lambda(X_j^i)|
\]  
(A.9)

A.4.2 Label propagation for SSRF

For the SSRF scenario where training data contains labeled and unlabeled samples, an initial label propagation from labeled samples to unlabeled data points is required before applying the ensemble leaf prediction model of eq. A.6.

Label propagation from labeled data, \( X^S \) to unlabeled data, \( X^U \) can be achieved directly via the following minimisation:

\[
c(U) : c(\arg \min_{X^S} D(U, X^S)), \forall X^U \in X^U
\]  
(A.10)

The function \( c(\cdot) \) indicates the class index associated with a point and the generic geodesic distance \( D(\cdot, \cdot) \) is defined as:

\[
D(U, S) = \min_{x_0, x_n} \sum_{i=0}^{L(\Gamma)-1} d(x_i, x_{i+1}),
\]  
(A.11)

with \( \Gamma \) the geodesic path (represented as a discrete collection of the points), \( L(\Gamma) \) its length, \( \{\Gamma\} \) the set of all possible geodesic paths with the initial and end points \( x_0 = x^U, x_n = x^S \) respectively. According to [30] the local distances \( d(\cdot, \cdot) \) are defined as symmetric Mahalanobis distances

\[
d(x_i, x_j) = \frac{1}{2}(d_{ij}^T \Lambda_{l(x_i)}^{-1} d_{ij} + d_{ij}^T \Lambda_{l(x_j)}^{-1} d_{ij})
\]  
(A.12)

with \( d_{ij} = x_i - x_j \) and \( \Lambda_{l(x_i)} \) the covariance associated with the leaf reached by the point \( x_i \).
A.4.3 The ensemble model: SSRF

Once labels are predicted for our unlabeled target samples, a general probabilistic classification rule \( p(y|x) \) needs to be defined that can be applied to previously unseen test input for label inference. In a forest with \( T \) trees we have \( t \in \{1,\ldots,T\} \) and during the random training process all trees are learned independently and possibly in parallel resulting in their corresponding partitions of the feature space. After label propagation a class label is also attached to all available data (with different trees possibly assigning different classes to the samples in \( U \)). During testing, a Semi-supervised classification tree \( t \) yields as output the posterior \( p_t(y|x) \) [30]. Just as what we expect in a supervised forest, counting the examples of each class arriving at each leaf defines the tree posteriors \( p_t(y|x) \). The assumption is that the input point \( x \) is available during the training hence, using all real labels and propagated ones, the forest output is the usual posterior mean over the whole trees in the forest (eq. A.6).

A.5 Transductive Random Forest

In Section A.3 we explained theoretical principles of Criminisi et al.’s [30] Semi-supervised forest (SSRF). In this section we propose a novel semi-supervised random forest by adapting the information gain function of SSRF for transductive learning. We coin the proposed model Transductive Random Forest (TRF). Similarly to SSRF, the TRF is trained with a combination of the labeled (source data subset in the TL problem or a transformed version of them) and unlabeled (target) data. In the next section we propose an integration of TST (the source space transformation method of Section 4.3.2) with both SSRF and TRF.

A.5.1 The training objective function

As described in Section A.3 during the tree training process the aim is to find the optimal parameter \( \theta^* \) for all the split nodes and this can be obtained by maximising an information gain
A.5. Transductive Random Forest

objective function:

\[ \theta_j^* = \arg \max_{\theta_j} I_j \quad (A.13) \]

with

\[ I_j = I(X_j, X_j^L, X_j^R, \theta_j) \quad (A.14) \]

where the symbols \( X_j, X_j^L \text{ and } X_j^R \) present the set of the points before the split and the subsets sent to Left and Right child nodes respectively. In a TL problem setting, the set of training data contains a combination of the labeled source and unlabeled target samples possibly from a different spacial distribution to source distribution and hence, \( X_j = X_j^{src} \cup X_j^{trg} \). As a result the objective function \( I_j \) must encourage both separation of the labeled training data as well as separating different high density regions from one another (accounting for unlabeled target/test samples in the case of transfer learning). As the source and target distributions in transductive transfer learning are different from each other, we propose a new information gain term in eq. A.7 which guarantees the separation of the high-density target regions.

Additionally we introduce scalar weighting parameters \( \alpha \) and \( \beta \) such that \( \alpha + \beta \leq 1 \). This enables us to control the influence of the final three measure based on information gain in the model based on the characteristics of the studied problem. We further proposed a normalisation of these weighting parameters to balance out the final effect. Presenting the set of labeled source samples, \( X^{src} \text{ with } S \) and the set of unlabeled target data, \( X^{trg} \text{ with } U \), the final information gain can be formulated as:

\[ I_j = \alpha I_j^{U\cup S} + (1 - \alpha - \beta) I_j^S + \beta I_j^U \quad (A.15) \]

Here the information gain \( I_j^S \) is the supervised term depending only on the labeled source training data, defined in eq.A.16 while \( I_j^U \) defined as eq.A.18, is the information gain derived only using the unlabeled target distribution and \( I_j^{U\cup S} \) stands for the unsupervised term and depends on all data (eq.A.17). The scalar parameters \( \alpha \) and \( \beta \) are user defined and specify the relative weights between the terms.
Appendix A. Transductive Random Forest

As explained in Section A.3 for the supervised information gain, $I_j^S$ we adopt the Shannon entropy [146], and for the two remaining unsupervised information gains, $I_j^U$ and $I_j^{U,S}$ we use mutual entropy introduced by eq. A.8:

$$I_j^S = -\sum_c p(X_j|Y_j = c) \log p(X_j|Y_j = c) + \sum_{i \in \{L,R\}} \left( \frac{|X_j^i|}{|X_j|} \sum_c p(X_j^i|Y_j^i = c) \log p(X_j^i|Y_j^i = c) \right), \quad (A.16)$$

$$I_j^{U,S} = \log |\Lambda(X_j)| - \sum_{i \in \{L,R\}} \frac{|X_j^i|}{|X_j|} \log |\Lambda(X_j^i)| \quad (A.17)$$

and finally:

$$I_j^U = \log |\Lambda(X_j)| - \sum_{i \in \{L,R\}} \frac{|X_j^i|}{|X_j|} \log |\Lambda(X_j^i)| \quad (A.18)$$

A.5.2 The randomness model

Adding an extra information gain and the weighting parameters $\alpha$ and $\beta$ enables the TRF for an extra possible source of randomness into the forests during the training phase in addition to the ones introduced in A.3.2. Specifically speaking one can:

- randomly choose a subset of the possible information gain weighting parameters ($\alpha$ and $\beta$).

The three techniques are not mutually exclusive and can be used all together where the first randomness source derives from using a subset of the training data for training each individual tree in the forest, the second means that if $\Theta$ is the entire set of all possible parameters $\theta$, then when training the $j^{th}$ node we only make available a small subset $\Theta_j \subset \Theta$ of such values as discussed in Section A.3.2 and [30] and the third technique suggests that for each tree in the forest, the non-negative $\alpha, \beta$ parameters of eq. A.15 be randomly initialised.
A.6 Adaptive Transductive Random Forest

Inspired by the TL techniques available in literature, in this section we introduce the Adaptive Transductive Random Forest (ATRF) as an enhanced TRF with node adaptation capacity. The ATRF algorithm in principle is quite similar to TRF with an additional source to target distribution adaptation step at forest nodes.

Two node-level distribution adaptations are introduced where the first, $ATRF_1$, is based on a simple global mean adjustment between the two domains:

$$\hat{X}_{src}^j = X_{src}^j - E(X_{src}^j) + E(X_{trg}^j) \quad (A.19)$$

And the second, $ATRF_2$, is based on adjusting both means and variances of the two source and target distributions:

$$\hat{X}_{src}^j = (X_{src}^j - E(X_{src}^j)) \sigma_{src}^j \sigma_{trg}^j + E(X_{trg}^j). \quad (A.20)$$

Both proposed ATRF algorithms perform a mean or both mean and variance matching between the two source and target distributions at each node level and then search for optimal decision boundaries based on a union set of the transformed source samples and unlabeled target samples. Specifically, at the node level before applying the information gain equation (eq. A.15), the means of the source and target subsets of data in a specific node are first matched and then based on the mutual entropy computed by eq. A.15 the node splits are defined.

A.7 Experiments and Results

In this section we first evaluate the semi-supervised random forest and geodesic label propagation method. We then demonstrate the effectiveness of combining the implemented methods of Chapter 4 with SSRF and the proposed TRF. Finally we show experimental evaluations of the
ATRF performance. To evaluate the performance of the proposed approaches, we tested them on a wide range of synthetic and public datasets.

A.7.1 Two-Moons Synthetic Dataset Experiments

The experiments in this section are designed to visually show the effect of the semi-supervised random tree explained earlier in A.5 using a toy problem called Two-moons. Our results are compared with the state-of-the-art domain adaptation techniques which are discussed in [59]. The synthetic dataset has been generated following the same settings as of [59].

Semi-Supervised Tree

The Two-Moons binary classification task is composed of two inter-twinning semicircle-shaped data distributions. To adapt this problem for a domain adaptation task, a rotation angle set of 20°, 30°, 40°, 50° is applied to the target data. Figure A.3 shows a graphical comparison between the basic supervised tree classifier and the semi-supervised tree (SST). In addition, in order to compare the proposed geodesic-based label propagation of Section A.4.2 with a KNN label propagation model we performed a KNN label propagation on the tree classifier outcomes where the number of the neighbors \( k \), is set to 3. In Figure A.3, the red and blue circles represent the two classes of labelled source samples and the cross marker stands for unlabeled data. The colors in the case of the unlabeled target data shows how well they have obtained the correct labels after applying the relevant technique. As one can note, Criminì et al.’s semi-supervised tree classifier outperforms all the other approaches [30]. This suggests that the use of full covariances for the label propagation was advantageous.
A.7. Experiments and Results

Figure A.3: The first column demonstrates the baseline result after applying a basic Tree classifier on 4 rotation angles. From top to bottom, 20°, 30°, 40° and 50°. The red and blue circles represent the labeled source samples and the cross marker stands for unlabeled data and their color shows their classification results. The second column shows the KNN label propagation applied on the Tree classification results. Finally the third column shows the result of applying the semi-supervised Tree with geodesic label propagation. Note that the tree’s depth is fixed to 5 for all the experiments.
Table A.1 Average accuracy results for 4 rotation angles captured from [59] (The row representing our results is made using the synthetic dataset generated locally following the same setting as of [59]). As can be seen our approach outperforms all the other methods in 3 out of 4 rotation angles.

<table>
<thead>
<tr>
<th>Rotation angle</th>
<th>20°</th>
<th>30°</th>
<th>40°</th>
<th>50°</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>89.6</td>
<td>76</td>
<td>68.8</td>
<td>60</td>
</tr>
<tr>
<td>TSVM [82]</td>
<td>100</td>
<td>78.8</td>
<td>74.6</td>
<td>70.9</td>
</tr>
<tr>
<td>DASVM [21]</td>
<td>100</td>
<td>78.4</td>
<td>71.6</td>
<td>66.6</td>
</tr>
<tr>
<td>DASF [117]</td>
<td>98</td>
<td>92</td>
<td>83</td>
<td>70</td>
</tr>
<tr>
<td>DA-PBGD [59]</td>
<td>97.7</td>
<td>97.6</td>
<td><strong>97.4</strong></td>
<td>53.2</td>
</tr>
<tr>
<td>Semi-supervised Tree [30]</td>
<td><strong>100</strong></td>
<td><strong>99.7</strong></td>
<td>90.4</td>
<td><strong>90</strong></td>
</tr>
</tbody>
</table>

To better evaluate the efficiency of the proposed semi-supervised tree classifier, we further compared it with other state-of-the-art methods in Table A.1. In this experiment we generated the synthetic dataset following the same settings as of [59]. As can be seen, Crimini et al.’s semi-supervised tree approach outperforms all the other methods in 3 out of 4 rotation angles.

The best performing method among the state-of-the-art methods of Table A.1 is Probably Approximately Correct Bayesian (PAC-Bayesian) approach that seeks for suitable weights to be given to each hypothesis in order to build a majority vote. In [59], the authors prove a new DA bound in the PAC-Bayesian context and advocate that the design of a DA-PAC Bayesian algorithm be based on the minimisation of the proposed bound. Doing so, they look for a weighted majority vote that takes into account a trade-off between three quantities. The first two quantities being, as is usual in the PAC-Bayesian approach, are the complexity of the majority vote (measured by a Kullback-Leibler divergence) and its empirical risk (measured by the average errors on the source sample). The third quantity is the capacity of the majority vote to distinguish some structural difference between the source and target samples.

The results confirm that the semi-supervised tree classifier is more robust to different rotation angles and outperforms all the other methods in three rotation angles.
Semi-Supervised Tree+TST and Semi-Supervised Tree+reweight

In the previous section we have tested the performance of Semi-supervised tree on the Two-Moons synthetic data. In this section we will set up a series of simple translation \((dx)\) and scaling \((s)\) transformations instead of the rotations of the previous section.

To better visualise the transformation estimations, at each time only one class undergoes some deformations or displacements and the second class remains untouched. Two sets of transformations are introduced. The first is a simple scaling operation along the x-axis and the second is a horizontal translation. Figure A.4 displays these setups, where in each sub-figure the blue and red distributions represent the two source classes and the Magenta and Cyan colors stand for target classes.

As explained earlier in this appendix, in order to combine the semi-supervised tree (SST) with either TST or reweight domain adaptation method of Chapter 4, we first train a semi-supervised tree classifier of fixed-depth 5, with a combination of the unlabeled target samples and labeled source samples to get initial prediction for target labels and then we will apply the TST (or reweight) transformation matrix to transformed source distribution. The obtained transformation parameters are then applied to the source samples to reduce the distance between the source and target domain distributions. The next step is to retrain the SST using this transformed source samples for a final target label prediction.

To verify the transformation matrix we have plotted each transfer parameter with respect to the actual displacement in our training data. According to the theory for the reweight method and in a two dimensional space scenario, the scaling transfer matrix has two horizontal and vertical scaling factors represented with \(s_x\) and \(s_y\) respectively. As for the TST transfer method, we have one parameter for scaling and one parameter for translation along each axis. Figure A.5 presents these evaluations.

One can see that the transfer learning transformation parameters agree with the initial target class transformations except in the extreme limits where the transformed target class either falls quite far from its corresponding class in the source domain or it displaces in a way that has
Figure A.4: Experimental setup: Blue and red distributions stand for the source classes and the Magenta and Cyan colors represent the target samples. The first column shows the introduced scaling transformation with varying scaling factors and the second presents the $x$-axis translations.
Figure A.5: Transfer parameters obtained from applying the transfer learning algorithm with respect to initial-space transformations. The first two graphs from left to right demonstrates the variations of the scaling and translation factors for the scaled data and the remaining two graphs shows the relevant factors for the translated target sets.
more intersection with a wrong class of source distribution. In conclusion, the results show that the combination of the SST with the TST or reweight boosts the performance when compared to using supervised tree with TST or reweight. But in general none of the two transformation schemes can be concluded as an ultimate winning method.
A.7. Experiments and Results

TRF: Evaluation of the Weighting Factors of the Node Split Optimisation Criterion

The experimental results, presented so far, showed that in a few synthesised scenarios of rotated, scaled and translated transfer tasks the semi-supervised tree and its combination with TST and reweight fail to either correctly propagate the labels or estimate the introduced correct transfer factor. We call these cases extreme tests. The extreme tests include $dr = 50^\circ$, $ds = 0.5$ and $dx = 0.3\pi$ of rotated, scaled and translated transfer tasks respectively.

In this section we will show how the information gain weights setting of the Transductive Tree (TRF with a single tree) can affect these extreme tests. Specifically, we investigate the influence of the $\alpha$ and $\beta$ (eq. A.15) transductive weight parameter choices on the transductive tree classification performance. Figure A.6 illustrates the results. The evaluation is carried out on the datasets where the TRF classifier was failing.

Figure A.6: The effect of the $\alpha (U \cup S)$ and $\beta (U)$ parameters of eq. A.15 on the performance for the datasets where the semi-supervised method was not as successful as expected. Only the upper triangle of each of image graphs on the second row is acceptable due to the fact that all the scale factors for the different criteria of eq. A.15 have to be positive values and sum to 1.
As expected, when the distributions of two target classes are well separated (like the case for the rotated, \(dr = 50^\circ\) and scaled data, \(ds = 0.5\)), high values of \(\beta\) (unlabeled criterion weight in eq. A.15) result in performance improvement. In these cases the higher value of the \(\beta\) will increase the influence of the unlabeled target entropy in tree’s node objective functions, hence resulting in a more accurate clustering of the unlabeled target distribution.

The same does not hold for the translated test (\(dx = 0.3\pi\)) since the translated target class suffers from a high intersection with an incorrect source class distribution. As a result, giving more weight to the unsupervised data \((U \cup S)\) criterion, \(\alpha\), works more efficiently. This is due to also taking into account the labeled source data in defining the node splits. Not surprisingly, the case when \(\alpha = \beta = 0\) also results in a high accuracy since relying only on the labeled samples gives a better discrimination between the two classes.

The comparison of the three extreme tests’ accuracy results of Figure A.6 shows that a common choice of \(\alpha = 0\) and \(\beta = 1\) can lead to achieve an acceptable performance in all three problem settings. This suggests that in a TL problem, the unsupervised information gain term, \(I_{j}^{U \cup S}\) (in eq.A.15), can be removed. This indeed will also reduce the time complexity of the training phase.

In conclusion, an optimal choice of \(\alpha\) and \(\beta\) parameters depends on data distribution variation from source to target. In a TL scenario due to the absence of labels in target domain, we do not have any knowledge about the distribution variations. Hence as discussed earlier in Section A.5.2, we proposed that \(\alpha\) and \(\beta\) parameters for each tree in the forest be chosen randomly.

**Transductive Random Forest**

As explained in Section A.5.2, we proposed a random initialisation of the weight parameters in eq A.15. In this section, following the transductive weight parameters analysis of the previous section we study the effect of this randomness on the TRF performance in extreme tests i.e. \(dr = 50, s = 0.5\) and \(dx = 0.3\pi\).
A.7. Experiments and Results

In these experiments we compared the performance of the supervised tree and Transductive tree with equal weights for all three information gain criteria and TRF with random weight setting. Because of the intrinsic randomness in the TRF model, the results are reported based on the mean and the standard deviation (STD) of 10 iterations of the same experiment while the number of the trees is fixed to 5. The numerical results are reported in Table A.2.

<table>
<thead>
<tr>
<th>dataset</th>
<th>supervised tree</th>
<th>Transductive tree (I = I^S + I^{U+S} + I^U)</th>
<th>TRF (random selection for (\alpha, \beta) values) \ T = 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>(dr = 50^\circ)</td>
<td>68.3</td>
<td>90.0</td>
<td>92.89 (\pm) 1.32</td>
</tr>
<tr>
<td>(s = 0.5)</td>
<td>89.6</td>
<td>91.7</td>
<td>93.84 (\pm) 4.94</td>
</tr>
<tr>
<td>(dx = 0.3\pi)</td>
<td>79.0</td>
<td>67.0</td>
<td>75.73 (\pm) 3.31</td>
</tr>
</tbody>
</table>

As one can see the proposed TRF outperforms the other methods. The inserted randomness in the model is helping in accomplishing a more generalisable classifier.

A.7.2 Experiments on the Chars74K Dataset

The experiments in this section are designed to visually present the effect of the transfer learning method of Section 4.3.2 on high dimensional data. We used the characters dataset Chars74k of [38]. The dataset and its acquisition are fully described in Section 3.2. Out of 64 possible classes of upper and lower case English letters and zero to nine digits, we evaluate our method on 10 classes of digits. So in the following experiments we have ten classes and we only use the Normal fonts for the first set of experiments.

The formerly described dataset is computer generated. In order to make it more realistic we have inserted a 0-mean Gaussian noise of variance \(\pm 25\) to the source data while the target inserted noise has a variance varying between 0 and 150 with a step size equal to 5 grey levels. We have also set up experiments where two domains have the same noise variance level but with different noise mean values. Sample images of the digit 0 are presented in Figure. A.7.
We have reevaluated the performance of the proposed TRF+TST domain adaptation approach on the described subset of Chars74k data. Figure A.8 presents these results. The graphs on the first row demonstrate the mean accuracy versus the variations in the standard deviation ($\sigma$) and means ($\mu$) for the two proposed experimental scenarios. The graphs in the second row present the deviation from the ground truth values. The horizontal axis shows the inserted input noise (from left to right variations in standard deviation and means) and vertical axis represents the ratio of the target noise standard deviation to source noise standard deviation (STD) in the bottom left image and the target estimated mean value in the bottom right figure.

In the second stage of the experiments, we have repeated the same setup as of the previous ones except that the injected source noise STD level is changed to $\pm 5$ and $\pm 20$ respectively. The results of these settings are shown in Figure A.9 first and second rows respectively.

The results show that our proposed TRF+TST method is reasonably successful in detecting the correct transfer from source to target in these simple visualisation experiments but not always useful to improve the performance accuracy.
A.7. Experiments and Results

Figure A.8: Evaluation of the TST Transfer learning method on the performance of the TRF classifier. The first two graphs on the top present a comparison between the baseline of the RF and the effect of using the transfer learning transformation with two different transfer rates (see eq. 4.8). Note that the source data is suffering from a fixed ±25 Gaussian noise throughout all these experiments.
Figure A.9: Evaluation of the effect of the TST Transfer learning method on the performance of the random decision forest classifier. The first and third row correspond to the case where the source noise variance is set to 20 and the second and forth rows represent the results for the case of source noise variance equal to 5 grey levels.
Normal→Bold Font Experiments: Understanding Failure with Trees for Classification

In this section we increase the difficulty level of the experiments by defining the domain adaptation task between the Normal and Bold font samples. In the first set of Normal→Bold transfer experiments we train a supervised Tree for a binary classification task of distinguishing the 0 and 1 digits from each other. As explained earlier in Section 3.2 the features in this dataset are the vectorised intensity values of the image. There are 254 possible fonts on each class and the difference between the source and target domain is based on being a normal or bold font correspondingly. After classifying the training set using the decision tree, 7 pixels/features are selected for defining the node split thresholds. The tree classifier in both training steps of TST algorithm had been set to a single supervised tree of depth 4, with all the features considered for the node split optimisation. Figure A.10 presents the averaged source, target and TST transformed source samples of both 0 and 1 classes. First column presents the mean of the source samples for classes 0 and 1 classes where the pixels/features which have been selected for defining the node split thresholds are highlighted in white. The middle column shows the mean of the samples in the target domain (note the different intensity range for these features in the $src$ with respect to the $trg$). The last column presents the mean of $TST$-transformed source samples where the selected feature for tree split (after tree retraining step in TST) is shown in white. $G(x = 104, y = 0) = 255$ and $G(x = 104, y = 1) = 254.98$ which consequently results in wrong thresholding at 254.99 value for class separation.

As it is seen (last column of the Figure A.10) after applying the $TST$ transformation on source samples and retraining the tree classifier, the first-optimum split feature is selected at background area of the image ($x_{104}$). While this background pixel can be assumed as a good discriminant for the transformed source distribution, it will clearly fail in correctly classifying the target classes.

Figure A.11 presents the $TST$ algorithm transfer-parameter estimations: The tree classifier’s depth is fixed ($depth = 4$) during all training phases but since in the retraining step after transformation it finds the first-best split on the first depth, it does not expand further resulting
Figure A.10: The first column presents the mean of the source samples for 0 and 1 classes with the node split pixels/features shown in white. The middle column shows the images of the corresponding classes in the target domain. The last column represents the transformed source images where the feature which is selected for node split thresholding (after tree retraining step in TST) is shown in white (note the difference in $G(x = 104 | y = 0) = 255$ and $G(x = 104 | y = 1) = 254.98$. This selection consequently results in bad thresholding at 254.99 value). The tree in both training steps is a single supervised tree of depth 4 while all the features are considered for the node optimisations.

In an incorrect target classification. Indeed, in the transformed src the background feature (pixel intensity) 104 of the two source classes has different values (fig. A.10 last row). As one can see (fig. A.11), not only the variances of two classes but also the target expectations are different between the two classes. These different feature-transfer factors eventually affect the transformed intensity value of features (see eq. 4.9) in the transformed source space and results in a bad node-split threshold which consequently corrupts the target classification.

In order to avoid this fine separation of the tree in the retraining step after the TL algorithm, a basic and simple solution is to somehow avoid this feature. This can be imposed by simply injecting some randomness in selecting the features by either selecting them randomly (using random forest instead of a single tree) or even going one step further and draw random splits for each of the randomly selected features (i.e. applying extremely randomised forest).

As this problem rises due to the redundant feature of our space and our final goal is to have an
A.7. Experiments and Results

159

accurate classifier for the target domain, we need to look for some pivot features of this new, unseen domain. Specifically, there are some features with zero or close to zero variances in the target domain (the previously mentioned feature 104 of fig. A.10 is one of them). These features of target space which are so called target redundant features, should be avoided during the tree training phase optimisations. Removing them will not change the target domain but it has two advantages. First, it is a sensible method of dimensionality reduction that speeds up the tree training phase. Second, it stops the already mentioned problem which our forest classifier was encountering. To do so, we have selected the target features which have non-zero variances:

\[
\text{if } \text{STD}(x^i) \leq th, \text{ then remove } f \text{ from the feature set } F \quad (A.21)
\]

or alternatively for this dataset we can even go one step further and rewrite the above equation as:

\[
\text{if } \text{STD}(x^i) \leq th \cap \sigma(X^{src}) \leq th, \text{ then remove } f \text{ from } F \quad (A.22)
\]

For each domain in the above equations, all the available samples are taken into account, re-
gardless of their labels.

**Table A.3** Comparison among different classifiers performance on the Chars74K dataset. Only the two 0 and 1 classes are selected for these experiments. In the case of the *extremely random forest* classifier, in order to take into account the effect of the injected randomness, the results are presented as the mean and standard deviation of the accuracies of 10 runs of the same classifier.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>TR =0</th>
<th>TR=0.5</th>
<th>TR=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree: $depth = 4$,</td>
<td>93%</td>
<td>79%</td>
<td>55%</td>
</tr>
<tr>
<td>ExtraTree: $depth = 4$,</td>
<td>98.03 ± 0.65</td>
<td>62.09 ± 19.36</td>
<td>65.25 ± 20.66</td>
</tr>
<tr>
<td>ExtraTree: $depth = 4$, feature selection based on target and source pivot feature (eq. A.22)</td>
<td>96.63 ± 2.32</td>
<td>91.65 ± 5.04</td>
<td>90.06 ± 7.9</td>
</tr>
</tbody>
</table>

The results are presented in Table A.3. As one can see in Table A.3 the proposed feature selection helps in avoiding the redundant features during the node optimisation step hence resulting in an enhanced performance for both transfer rates 0.5 and 1.

Since in the case of the Chars74k dataset, we know that the relevant information is held in pixels that their intensity values change from 0 to 255, we set the threshold value to $th = 50$.

**The Effect of the Principal Component Analysis**

Further evaluations for our proposed TST domain adaptation approach are reported in this section where we compared our method with the state-of-are method of [107].

As a preprocessing step we project the data via principal components of the combination of the source and target data. We further investigate the effect of using the NN classifier instead of a supervised Tree. The results are presented in Table A.4.

We compare the supervised Tree instead of the nearest neighbour after TL step in the second row of the Table A.4 and as expected, the performance diminishes, but we are still gaining some improvement by applying the TST domain adaptation.

The results of Table A.4 show that the PCA projection enhanced the performance but the tree classifier in comparison to the NN classifier is clearly failing.
A.7. Experiments and Results

Table A.4 Applying the PCA on the Char74 full-dataset of 10 classes and trying the $TST$ transformations for domain adaptation.

<table>
<thead>
<tr>
<th>Initial estimator: Tree Retraining classifier: NN</th>
<th>$\theta : 0$</th>
<th>$\theta : 0.5$</th>
<th>$\theta : 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>98.7</td>
<td>97.36</td>
<td>93.23</td>
</tr>
<tr>
<td>Initial estimator: Tree Retraining classifier: Tree</td>
<td>70.39</td>
<td>73.62</td>
<td>72.52</td>
</tr>
</tbody>
</table>

Transductive Random Forest with Node Adaptation

In this section we examine the proposed Adaptive Transductive Random Forest (ATRF) method which we described earlier in Section A.6. Table A.5 demonstrates some results comparing the original random forest with its transductive and adaptive transductive versions. As stated before, the $ATRF_1$ applies a simple mean alignment between the source and target subsets arriving at each node of the forest. In the $ATRF_2$ on the other hand, we try to adjust both means and variances of the source and target subsets arriving at forest nodes.

Table A.5 The accuracy of different generations of the Extremely Randomised Forest on the Chars74k subset. The classifiers are all trained on the same subset of Chars74k dataset composed of 14 samples in each 10 classes of 0-9 digits. 140 samples of Normal characters are used for training while 140 samples of the Bold font data are acquired for testing (target).

<table>
<thead>
<tr>
<th>Classifier</th>
<th>RF</th>
<th>TRF</th>
<th>$ATRF_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>76.43%</td>
<td>97.86%</td>
<td>99.29%</td>
</tr>
</tbody>
</table>

As one can see the performance improved after using the TRF and $ATRF_1$ with respect to a basic RF classifier. In the next set of experiments we will evaluate the performance of the proposed methods on public datasets of USPS, MNIST and COIL20. The cross-domain transfer task is as discussed in Chapter 5.

As one can see in Table A.6 the iterative version of the $TST + NN$ transfer algorithm is more successful compared to other methods. The results in Table A.7 shows that the $ATRF_2$ which benefits from more accurate node adaptations is performing better than both $TST + NN$ and $ATRF_1$ in COIL20 transfer tasks (note that this result is biased by the high similarity of COIL1 and COIL2 domains) but it is still performs drastically worse than the $TST + NN$ in
Table A.6 A comparison of three different transductive transfer learning approaches. The JDA stands for the Joint Distribution Adaptation method of [107]. The original data (without any PCA dimensionality reduction) is used as for the transductive transfer extremely randomised forest (ERF) method.

<table>
<thead>
<tr>
<th>Exp.</th>
<th>JDA [107] (NN)</th>
<th>Baseline ERF $T = 10$</th>
<th>$T = 100$</th>
<th>$ATRF_1$ PCA-comp:1</th>
<th>$ATRF_1$ PCA-comp:0.7</th>
<th>$ATRF_1$ PCA-comp:0.9</th>
<th>TST (NN) iter $= 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST→USPS</td>
<td>67.22</td>
<td>49.64</td>
<td>52.79</td>
<td>49.08</td>
<td>37.3</td>
<td>74.34</td>
<td>78.72</td>
</tr>
<tr>
<td>USPS→MNIST</td>
<td><strong>59.65</strong></td>
<td>17.00</td>
<td>17.87</td>
<td>15.74</td>
<td>25.37</td>
<td>47.78</td>
<td>50.70</td>
</tr>
<tr>
<td>COIL1→COIL2</td>
<td>89.31</td>
<td>82.78</td>
<td>85.42</td>
<td>83.89</td>
<td><strong>89.86</strong></td>
<td>88.89</td>
<td>88.75</td>
</tr>
<tr>
<td>COIL2→COIL1</td>
<td>88.47</td>
<td>82.08</td>
<td>83.89</td>
<td>85.42</td>
<td>86.94</td>
<td>87.78</td>
<td><strong>89.02</strong></td>
</tr>
</tbody>
</table>

the MNIST→USPS and USPS→MNIST tasks.

A.8 Summary

In this appendix, we showed how the decision tree classifier can be adapted for solving a transductive transfer learning problem. We introduced transductive random forest (TRF) by proposing modifications to the tree node optimisation criteria. We evaluated our method by testing

Table A.7 Comparison of ATRF with different node transformations. The first, $ATRF_1$, is where we use a simple mean adjustment for node transformation criteria at the ATRF. The transformation is processed one step further compared to the forest in the second row by adjusting both means and variances of the two src and trg distributions at each node of the forest using $ATRF_2$. The third row shows the performance of the $TST$ adaptation where NN classifier is used for learning. Number of trees in the forest is fixed to $T = 10$ for all the experiments.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>MNIST→USPS</th>
<th>USPS→MNIST</th>
<th>COIL1→COIL2</th>
<th>COIL2→COIL1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ATRF_1$</td>
<td>37.3</td>
<td>25.37</td>
<td>89.86</td>
<td>86.94</td>
</tr>
<tr>
<td>$ATRF_2$</td>
<td>42.34</td>
<td>23.96</td>
<td><strong>92.36</strong></td>
<td><strong>87.92</strong></td>
</tr>
<tr>
<td>TST + NN</td>
<td><strong>74.34</strong></td>
<td><strong>47.78</strong></td>
<td>88.89</td>
<td>87.78</td>
</tr>
</tbody>
</table>
its performance on a number of illustrative synthetic and real datasets and showed that the proposed technique outperforms the basic random forest classifier in a range of experiments. We further introduced an adaptive transductive random forest (ATRF) by proposing mean and variance adaptations in the node levels.

We have also combined the proposed RF-based algorithms with the feature transformation method of Chapter 4.

We compared our proposed TRF and ATRF with the state of the art method of [107]. Experiments of combining the proposed TRF with domain adaptation method of Chapter 4 confirmed the incompatibility of the random forest based models with feature transformation adaptations. This is mainly due to the high sensitivity of the node split function to the feature intensities. In addition, due to the node-criteria optimisation and pair-wise distance computation of the label propagation step, TRF and ATRF both suffer from high computational complexity which in practice makes them incapable of dealing with data of high dimensionality. The high complexity issue of TRF and ATRF is caused by the exhaustive covariance computation of the unsupervised information gain terms in eq. A.15. This in turn stops the use of more trees and large training sets which are necessary for training a better generalisable tree-based classifier.

In summary, Forests are good for large training datasets, but the complexity of transduction scales badly with the size of the dataset.
Appendix B

Additional Results

B.1 Iterative Algorithms for TST Domain Adaptation

As discussed earlier in Section 5.2.4, here we compare three iteration scenarios for our TST domain adaptation algorithm. In the first scenario we iteratively adapt the source and target distributions regardless of class labels. In each iteration the source distribution is transformed in a way that its mean and STD get closer to the target mean and STD (Method1).

In the second scenario the transformation becomes more class specific by estimating a set of pseudo labels for the target samples it tries to find a class specific transformation for the source clusters in each iteration (Method2).

The third scenario is a mixture of the two last ones meaning that at the first step we adjust the source and target distributions and then in the next iterations, the transformations become class specific as in the second scenario (Method3). Table B.1 shows a comparison between these three TL algorithms. In these experiments we have used KPCA in advance and selected the number of components such that the amount of STD that needs to be explained is greater than 97%. The classifier which is used is the Nearest Neighbor.
Appendix B. Additional Results

Table B.1 Comparison of three different iterative Transfer Learning Scenarios. The details of these three methods can be found in literature. NN classifier is used in all scenarios and the KPCA is performed on all datasets as a preprocessing step. The results in the table show the best performance in 50 iterations. Stopping criterion has yet been designed for these methods.

<table>
<thead>
<tr>
<th>TL method</th>
<th>MNIST→USPS</th>
<th>USPS→MNIST</th>
<th>COIL1→COIL2</th>
<th>COIL2→COIL1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>57.22</td>
<td>37.28</td>
<td>86.25</td>
<td>86.11</td>
</tr>
<tr>
<td>Method1</td>
<td>58.05</td>
<td>46.13</td>
<td>88.47</td>
<td>86.94</td>
</tr>
<tr>
<td>Method2</td>
<td>68.95</td>
<td><strong>59.19</strong></td>
<td>87.50</td>
<td>89.16</td>
</tr>
<tr>
<td>Method3</td>
<td><strong>71.54</strong></td>
<td>57.68</td>
<td><strong>88.61</strong></td>
<td><strong>89.86</strong></td>
</tr>
</tbody>
</table>

**Iteration on TST with NN Classifier** Figure B.1 shows the effect of the iterations on the TST+NN methods:

**Iteration on the TST+NN domain adaptation method:** The third row of the Figure B.1 show the results of using a more conservative approach by selecting a smaller transfer rate value ($\theta = 0.1$) throughout the all iterations.

In order to define a measure for automatically stopping the iteration for a given dataset, we proposed a measure based on the amount of the change in the transformed source matrix. In particular we suggest the use of the Frobenius norm between the covariances of the transformed source matrices of two consecutive iteration steps. The effect of this automatic divergence detection algorithm is presented in the forth row of Figure B.1. An alternative would have been the KL divergence on the covariances, but for KL divergence computation it is necessary to get non-zero determinant covariance matrices.
Figure B.1: Evaluation of the effectiveness of different iteration schemes where the proposed \textit{trans} + \textit{scal} transformation is used along with NN classification. The graphs of the first row are corresponding to the case where the transfer rate is fixed to 1.0 (full transfer). Second row graphs are depicting the experiment where the transfer rate is increasing gradually from 0.1 to 1.0 (eq.4.8). The third row experiment corresponds to a more conservative parameter selection by fixing the transfer rate \(\theta\) to 0.1 for all the iterations. In the forth row we have followed the same setting as of the third row but the iteration is automatically stops when the changes in the transformed source is less than a specific threshold.
Bibliography


Bibliography


[135] E. Rodner and J. Denzler. Learning with few examples by transferring feature relevance. In Joachim Denzler, Gunther Notni, and Herbert Süsse, editors, Proceedings of the 31st Annual Symposium of the German Association for Pattern Recognition (DAGM2009),


