Theoretical And Numerical Investigations Of Domain Adaptation Methods

David Mihai Mazurencu-Marinescu-Pele

München 2024

Theoretical And Numerical Investigations Of Domain Adaptation Methods

David Mihai Mazurencu-Marinescu-Pele

Dissertation
an der Fakultät für Physik
der Ludwig–Maximilians–Universität
München

vorgelegt von
David Mihai Mazurencu-Marinescu-Pele

München, den 12. Dezember 2024

Erstgutachter: Dr. Kosmas V. Kepesidis
Zweitgutachter: Dr. Dirk-André Deckert
Tag der mündlichen Prüfung: 17. Dezember 2024

Contents

Chapter 1

Introduction

1.1 Abstract

This thesis investigates the domain adaptation problem within the context of a supervised learning framework for disease detection. The study begins by outlining the mathematical foundations necessary for understanding domain adaptation, focusing on the PAC-learning framework and the VC dimension as measures of model complexity and generalization capability. A toy model is developed to simulate a domain shift, where an affine transformation is applied to a bivariate Gaussian dataset representing different disease states. Then, a particularly simple adaptation method is evaluated by comparing its performance across different learning scenarios, using various fractions of target domain data. A justification for its effectiveness is sought using the tools of Statistical Learning Theory. Finally, we numerically investigate several other adaptation methods that have already been implemented in publicly available packages, such as the ADAPT Python toolbox [1]. We conclude by proposing an alternative, original approach based on data augmentation, which has demonstrated success on a real-world dataset.

1.2 Motivation

The motivation behind choosing this topic is rooted in a very practical problem encountered in data science. We initially encountered this issue during the development of classification algorithms for a disease detection study. Specifically, the performance of our model suffered a significant drop when tested on data measured by the same device that produced the training samples, due to a shift in the data distribution. We refer to this as the domain adaptation problem, a broad term that encapsulates an effect prevalent in virtually all machine learning applications. Since this is a sensitive topic with significant potential for the future of medicine, every model that aspires to become more than a proof-of-concept must be made as reliable as possible, particularly in detecting and mitigating such shifts. In the remainder of this section, we provide a brief overview of the increasing role that data science plays in the medical field.

Data science and artificial intelligence have significantly transformed medical diagnostics by leveraging advanced computational techniques and vast amounts of data to enhance accuracy, efficiency, and personalization in healthcare [2, 3, 4, 5, 6, 7, 8]. One key area of application is predictive analytics for early disease detection, where machine learning models analyze patient data to predict the likelihood of diseases such as diabetes, cancer, and cardiovascular diseases. Early detection facilitates timely intervention and improves patient outcomes. These algorithms can classify patients into different risk categories based on their health data, enabling healthcare providers to prioritize care, particularly for high-risk individuals.

Another crucial application of data science in medical diagnostics is medical imaging analysis [9, 10, 11]. In radiology, for instance, deep learning models, especially convolutional neural networks (CNNs), are employed to interpret medical images such as X-rays, MRIs, and CT scans. These models can detect a wide range of abnormalities, including tumors, fractures, and infections, with remarkable accuracy. Additionally, in pathology, automated image analysis in histopathology helps identify cancerous cells and other anomalies in tissue samples, supporting pathologists in making accurate diagnoses.

Furthermore, data science plays a pivotal role in genomics and personalized medicine [12, 13]. Techniques used to analyze genomic data can identify genetic variants associated with diseases, which in turn guide the development of personalized treatment plans. In the field of pharmacogenomics, understanding how different patients respond to medications based on their genetic makeup allows for personalized drug prescriptions, minimizing adverse effects and maximizing therapeutic efficacy.

1.3 Additional information

A few words on the overall structure of the thesis are in order. Chapters 2 is written in as close a mathematical fashion as possible, following more or less standard literature in the field. Chapter 3 serves as a less formal intermezzo that introduces the actual problem and the toy model that aims to reflect it. Chapter 4 presents a possible theoretical justification of the FEDA method via two different approaches. Finally, Chapter 5 presents several adaptation methods and their application to a real-world dataset for the binary classification task of gender prediction, along with an original approach based on data augmentation.

The reference to each mathematical statement (definition, theorem, etc.) is either explicitly mentioned or provided just before the actual text as a number in brackets, corresponding to an item in the References section. For example, the VC generalisation bound theorem as it appears in [14] is introduced as "**Theorem 2.15** (VC generalisation bound). *[14]*".

All code used in the development of this thesis for model simulations, result visualizations and numerical experiments that involve artificially generated data are publicly available on GitHub at DavidMazurencuPele/DA-Methods-Investigation. Due to ethical considera-

tions, the remaining code is available upon reasonable request by contacting Dr. Kosmas V. Kepesidis at kosmas.kepesidis@physik.uni-muenchen.de.

Chapter 2

Theoretical Foundations

2.1 Introduction to Statistical Machine Learning

Following the approach outlined in [15], this section introduces the foundational concepts of *data generation*, *statistical environment*, *feature maps*, and the *learning process*, which together form the basis of statistical machine learning.

A *learning machine* refers to a system that processes data to approximate patterns and relationships within its environment, enabling predictions and decision-making under uncertainty. A *statistical environment* refers to the complete set of observable events or phenomena that could serve as input to the learning process. These events are essentially snapshots of the learning machine's environment over a particular time period and in a particular location, These can include anything from a patient's symptoms to stock market fluctuations or pixel values in an image. They are assumed to be generated by sampling from an unknown and inaccessible probability distribution, often referred to as the *environmental distribution*.

To represent these events numerically, a *feature map* is employed. A feature map is a function that maps each event in the statistical environment to a finite-dimensional real *feature vector*. This feature vector provides a numerical representation of the event's essential properties and is commonly referred to as a *data point*. A learning machine, equipped with this representation, aims not to memorize observed data but to approximate, via the learning process, the underlying environmental distribution as closely as possible. This approximation should enable the machine to generalize beyond the training data, making accurate predictions even on previously unseen samples.

While the concepts briefly discussed in this section provide an intuitive foundation for understanding the broader framework, we continue this chapter by discussing the central objects from a mathematical perspective. This is necessary in order to arrive at several fundamental results, which are invoked in the remaining part of this thesis.

2.2 Definitions and Notation

We begin our list of definitions by concretizing the collection of feature vectors using the language of probability theory. The primary references for the remainder of this chapter are [16], [14], [17], and [18]. For the proof of several implicit mathematical claims, we refer the reader to [19].

Definition 2.1 (Feature space)**.** *Let $\mathcal{X}$ denote the set of feature vectors, let $\mathcal{F}_{\mathcal{X}}$ be a σ-algebra on $\mathcal{X}$, and let $\mathbb{P}_{\mathcal{X}} : \mathcal{F}_{\mathcal{X}} \to [0,1]$ be a probability measure. Then we call the tuple $(\mathcal{X}, \mathcal{F}_{\mathcal{X}}, \mathbb{P}_{\mathcal{X}})$ the feature space, and we refer to it by the shorthand notation $\mathcal{X}$.*

The set $\mathcal{X}$ is typically a subset of a finite-dimensional real vector space $\mathbb{R}^d$, and $\mathbb{P}_{\mathcal{X}}$ captures the distribution of feature vectors induced by the data-generating process.

After establishing the structure of the feature space, we proceed in the same manner with the outputs of the learning machine.

Definition 2.2 (Label space)**.** *Let $\mathcal{Y}$ denote the set of possible labels, let $\mathcal{F}_{\mathcal{Y}}$ be a σ-algebra on $\mathcal{Y}$, and let $\mathbb{P}_{\mathcal{Y}} : \mathcal{F}_{\mathcal{Y}} \to [0,1]$ be a probability measure. Then we call the tuple $(\mathcal{Y}, \mathcal{F}_{\mathcal{Y}}, \mathbb{P}_{\mathcal{Y}})$ the label space, and we refer to it by the shorthand notation $\mathcal{Y}$.*

For example, in a *classification* problem, the outputs of a learning machine belong to a discrete label space, with $\mathcal{Y} = \{1, \dots, K\} =: [K]$, for some $K \in \mathbb{N}$.

Having defined the feature space $\mathcal{X}$ and the label space $\mathcal{Y}$, we now formalize the *supervised learning* setting.

Definition 2.3 (Supervised Learning (SL))**.** *Let $(\mathcal{X}, \mathcal{F}_{\mathcal{X}}, \mathbb{P}_{\mathcal{X}})$ be the feature space, and let $(\mathcal{Y}, \mathcal{F}_{\mathcal{Y}}, \mathbb{P}_{\mathcal{Y}})$ be the label space. The joint probability space of features and labels is defined as the tuple $(\mathcal{X} \times \mathcal{Y}, \mathcal{F}_{\mathcal{X}\times\mathcal{Y}}, \mathbb{P})$, where $\mathcal{X} \times \mathcal{Y}$ is the Cartesian product of the two sets, $\mathcal{F}_{\mathcal{X}\times\mathcal{Y}} = \mathcal{F}_{\mathcal{X}} \otimes \mathcal{F}_{\mathcal{Y}}$ is the natural product σ- algebra on $\mathcal{X} \times \mathcal{Y}$ and $\mathbb{P}$ is the joint probability measure, defined by*

$$\forall A \in \mathcal{F}_{\mathcal{X}}, \forall B \in \mathcal{F}_{\mathcal{Y}} : \ \mathbb{P}(A \times B) = \int_A \int_B d\mathbb{P}(x, y).$$

In the supervised learning setting, the data-generating process is assumed to sample independently and identically distributed (i.i.d.) pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ from $\mathbb{P}$. This motivates the following definition.

Definition 2.4 (Sample of size N)**.** *Consider a supervised learning scenario with the product probability space $(\mathcal{X} \times \mathcal{Y}, \mathcal{F}_{\mathcal{X}\times\mathcal{Y}}, \mathbb{P})$. For $N \in \mathbb{N}$, the Cartesian product of N such probability spaces is the tuple $((\mathcal{X} \times \mathcal{Y})^N, \mathcal{F}_{\mathcal{X}\times\mathcal{Y}}^N, \mathbb{P}^N)$, where:*

- $(\mathcal{X} \times \mathcal{Y})^N = (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \times \cdots \times (\mathcal{X} \times \mathcal{Y})$ *denotes the (N-fold Cartesian product.*

- $\mathcal{F}^N_{\mathcal{X}\times\mathcal{Y}} = \mathcal{F}_{\mathcal{X}\times\mathcal{Y}} \otimes \mathcal{F}_{\mathcal{X}\times\mathcal{Y}} \otimes \cdots \otimes \mathcal{F}_{\mathcal{X}\times\mathcal{Y}}$ *denotes (N-fold product σ-algebra), i.e., the smallest σ-algebra that makes all projections onto individual components measurable.*
- $\mathbb{P}^N$ *is the product probability measure, defined on $\mathcal{F}^N_{\mathcal{X}\times\mathcal{Y}}$, which satisfies:*

$$\mathbb{P}^N(A_1 \times A_2 \times \cdots \times A_N) = \prod_{i=1}^{N} \mathbb{P}(A_i),$$

for all $A_i \in \mathcal{F}_{\mathcal{X}\times\mathcal{Y}}$, $i = 1, \ldots, N$.

Then, a sample of size N is a collection of N independent and identically distributed (i.i.d.) random variables $\{Z_i = (X_i, Y_i)\}_{i=1}^N$ defined on the same product probability space $\mathcal{X} \times \mathcal{Y}$. Formally, the sample is represented as:

$$S_N = \{(x_i, y_i)\}_{i=1}^N \in (\mathcal{X} \times \mathcal{Y})^N.$$

The goal of a learning machine is to infer a *hypothesis* $h : \mathcal{X} \mapsto \mathcal{Y}$ that approximates the relationship between inputs (features) and outputs (labels).

Definition 2.5 (Hypothesis Space)**.** *Let $(\mathcal{X}, \mathcal{F}_\mathcal{X})$ be the feature space and $(\mathcal{Y}, \mathcal{F}_\mathcal{Y})$ the label space.The tuple $(\mathcal{H}, \mathcal{F}_\mathcal{H})$ is then called the hypothesis space, where $\mathcal{H} \subset \mathcal{Y}^\mathcal{X}$ is the set of hypotheses h and $\mathcal{F}_\mathcal{H}$ is a σ-algebra on $\mathcal{H}$.*

To formalize the learning process, we introduce the concept of a *learning algorithm*, which serves as the computational mechanism by which a learning machine operates. In this regard, we closely follow the definition of a *learner* in [20].

Definition 2.6 (SL algorithm)**.** *Let $(\mathcal{X}, \mathcal{F}_\mathcal{X})$ be the feature space, $(\mathcal{Y}, \mathcal{F}_\mathcal{Y})$ the label space, and $(\mathcal{H}, \mathcal{F}_\mathcal{H})$ the hypothesis space. Considering the supervised learning scenario, a supervised learning algorithm is a measurable mapping:*

$$\mathcal{A} : \bigcup_{N=1}^{\infty} (\mathcal{X} \times \mathcal{Y})^N \to \mathcal{H},$$

where measurability is defined with respect to $\mathcal{F}_\mathcal{H}$ and the σ-algebra generated by the union over N of $\mathcal{F}^N_{\mathcal{X}\times\mathcal{Y}}$.

The learning algorithm $\mathcal{A}$ maps a finite sample S of any size N to a hypothesis $h \in \mathcal{H}$, such that we can define

$$h_S \equiv h_{S_N} := \mathcal{A}(S_N). \tag{2.1}$$

as the hypothesis returned by the learning algorithm upon inspecting sample S_N.

For several SL scenarios one can define a deterministic function

$$c : \mathcal{X} \to \mathcal{Y},$$

called the *(deterministic) concept function*, which represents the true relationship that the learning algorithm aims to discover or approximate and that lies within the range of $\mathcal{A}$. However, not only might $c \notin \mathcal{H}$, but it might not exist at all. This can occur, for example, when dealing with a supposedly random outcome, as in the case of a dice roll, or when dealing with a highly non-linear, high-variability relationship between input and output, such as predicting age based on height.

The practitioner's prior knowledge constrains $\mathcal{H}$ to a proper subset of the entire function space $\mathcal{Y}^{\mathcal{X}}$, for example, by considering only linear transformations of the input.

Example 2.7 (Binary classification via separating hyperplanes)**.** *A binary classification task with $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{-1, 1\}$ that is performed on a linearly separable dataset, i.e., the two classes can be completely differentiated by a hyperplane $w = (w_0, \mathbf{w}) \in \mathbb{R}^{d+1}$, will have an associated hypothesis space*

$$\mathcal{H}_{\text{hp}} = \left\{ h \in \mathcal{Y}^{\mathcal{X}} \mid h : x \mapsto \text{sign}(w_0 + \mathbf{w} \cdot x),\ (w_0, \mathbf{w}) \in \mathbb{R} \times \mathbb{R}^d,\ \mathbf{w} \neq \mathbf{0} \right\}.$$

For a hypothesis h, the discrepancy between the predicted output $h(x)$ and the true label y is measured by a *loss function*

$$l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+ \cup \{0\},$$

which can take various forms depending on the specific task.

Example 2.8 (Common loss functions)**.** *Let $\mathcal{Y} = \mathbb{R}$. Then, among the loss functions that are heavily used in practice, we mention the following:*

- *0-1 loss: $l_{01} : \mathbb{R} \times \mathbb{R} \mapsto \{0, 1\},\ l_{01}(y, h(x)) = \mathbf{1}_{y \neq h(x)}$*
- *Mean squared error (MSE) loss: $l_{MSE} : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}^+ \cup \{0\},\ l_{MSE}(y, h(x)) = (y - h(x))^2$*
- *Hinge loss: $l_{HL} : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}^+ \cup \{0\},\ l_{HL}(y, h(x)) = \max(0, 1 - y \cdot h(x))$*

Most supervised learning algorithms operate by solving an optimization problem defined in terms of a quantity known as *the risk functional* $R[h]$. This functional evaluates the predictive power of a given hypothesis by taking the expectation of the loss function under the joint probability distribution:

$$R[h] := \mathbb{E}_{(X,Y) \sim \mathbb{P}}[l(Y, h(X))]. \tag{2.2}$$

Since the joint distribution $\mathbb{P}$ is typically inaccessible to us, the true risk cannot be computed. However, we can use the data sample S_N that we have at our disposal in order to construct a related quantity

$$\hat{R}_{S_N}[h] \equiv \hat{R}_S[h] := \frac{1}{N} \sum_{i=1}^{N} l(y_i, h(x_i)),$$

called *the empirical risk*.

The intuitive idea that the empirical risk should approximate the true risk is formally supported by the following result:

Theorem 2.9 (Expectation of the Empirical Risk)**.** *[17] For any hypothesis h and any given sample S, the following holds:*

$$\mathbb{E}_{S\sim\mathbb{P}^N}[\hat{R}_S[h]] = R[h]. \tag{2.3}$$

Proof. Using the linearity of the expectation operator, we get

$$\begin{aligned}\mathbb{E}_{S\sim\mathbb{P}^N}[\hat{R}_S[h]] &= \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{(x_i,y_i)\sim\mathbb{P}}\left[l(y_i,h(x_i))\right]\\ &= \mathbb{E}_{(x,y)\sim\mathbb{P}}[l(y,h(x))]\\ &= R[h].\end{aligned}$$

where we used that (x_i, y_i) are i.i.d., meaning that each term in the sum is identical. □

Since we now have a relevant computable quantity, namely $\hat{R}_S[h]$, the most natural approach to finding a hypothesis is to search for a minimizer:

$$\hat{h}_S := \underset{h\in\text{range}(\mathcal{A})}{\arg\min}\ \hat{R}_S[h].$$

This is known as *Empirical Risk Minimization* (EMR) and defining $\mathcal{F} := \text{range}(\mathcal{A}) \subseteq \mathcal{H}$, we assume that the algorithm will always output such a hypothesis h_S:

$$\forall h \in \mathcal{F} : \hat{R}_S(h_S) \leq \hat{R}_S(h).$$

However, the learned model h_S might not be a minimizer of the true risk, potentially leading to poorer performance on *test data*, i.e., data that is not part of the training process. To analyze this discrepancy further, we introduce the notion of a theoretically best hypothesis h^*, defined by

$$R[h^*] := \inf_{h\in\mathcal{H}} R[h].$$

Since the algorithm might not be able to exhaust $\mathcal{H}$, we introduce the *optimal hypothesis* $h_\mathcal{F}$, defined by:

$$R[h_\mathcal{F}] = \inf_{h\in\mathcal{F}} R[h]. \tag{2.4}$$

We can now inspect how badly does a given h do compared to the benchmark h^* using a trick called *error decomposition*. Thus, for all $h \in \mathcal{F}$:

$$R[h] - R[h^*] = (R[h] - R[h_S]) + (R[h_S] - R[h_\mathcal{F}]) + (R[h_\mathcal{F}] - R[h^*]), \tag{2.5}$$

where h_S is the output model after looking at S.

The last error term depends solely on how well can our algorithm approximate the best solution, so there is little one can do about it. Nevertheless, we can see that enlarging $\mathcal{F}$ could only decrease this type of error.

On the other hand, it is true that

$$R[h_S] - R[h] = (R[h_S] - R_S[h_S]) + (R_S[h_S] - R_S[h]) + (R_S[h] - R[h]) \tag{2.6}$$
$$\leq |R[h_S] - R_S[h_S]| + |R_S[h] - R[h]| \tag{2.7}$$
$$\leq 2 \sup_{\tilde{h} \in \mathcal{F}} \left| R[\tilde{h}] - R_S[\tilde{h}] \right|, \tag{2.8}$$

which implies that the second error term can only increase with a larger $\mathcal{F}$, since the above inequality must also hold for $h = h_{\mathcal{F}}$.

This trade-off is commonly referred to in the statistics community as the *bias-variance trade-off*, while in machine learning, it is recognized in terms of *underfitting* and *overfitting*. Specifically, if the model class is too restrictive, it results in higher bias and a larger gap between the achievable best and the theoretical best. As a result, the learned predictor may overlook the nuances of the training data, leading to poor performance on the training set but potentially better performance on the test set (underfitting). Conversely, a sufficiently rich model class increases variability, resulting in a greater deviation between what is learned and what could be learned. This suggests that the model has adapted too closely to the training data, thereby reducing its ability to generalize (overfitting).

So far, the whole discussion has been based on the assumption that one is dealing with i.i.d. train and test data, drawn from the same underlying distribution $\mathbb{P}$. However, this assumption is rarely met in real-world scenarios, where the non-stationarity of $\mathbb{P}$ is almost inevitable when transitioning from training to testing. A substantial amount of research has been dedicated to investigating and addressing this issue, which will be introduced in the next chapter. To fully understand it, we must first become familiar with the tools developed based on the foundations presented thus far. This is the purpose of the next section.

2.3 PAC-learning

Learning from data has historically been a successful endeavor. Alongside the practical advancements in the field, a substantial body of theory has developed to justify the heuristics that have guided practitioners since the field's birth. A central question that arises is: how can we guarantee that a learning algorithm will perform well on unseen data, given only a finite set of training examples? The Probably Approximately Correct (PAC) learning framework, introduced by Leslie Valiant in 1984 [21], provides a formal foundation for answering this question.

PAC learning offers a rigorous approach to understanding the conditions under which a hypothesis, learned from a finite set of examples, will generalize well to the entire input space. In this framework, learning is considered successful if, with high probability, the hypothesis is approximately correct, i.e., it has a low error rate on unseen data. This probability is computed using the probability measure $\mathbb{P}^N$ that governs the distribution of the samples seen during training.

The strength of PAC learning lies in its ability to quantify the trade-offs between the number of training samples, the complexity of the hypothesis class, and the accuracy and confidence of the learned model. By framing the learning process in probabilistic terms, PAC theory enables us to derive bounds on the *sample complexity*, that is, the number of training examples required to ensure that the learning algorithm achieves a desired level of performance.

Moreover, PAC learning provides insights into the feasibility of learning various classes of functions. It introduces key concepts such as the VC dimension (Vapnik-Chervonenkis dimension) [22], which measures the capacity of a hypothesis class and plays a crucial role in determining whether a class is PAC-learnable. Through the analysis of the VC dimension, PAC learning theory helps us understand the limitations of certain algorithms and guides the design of more efficient learning methods. The first part of the present section is dedicated to a discussion of this complexity measure.

The second part of this section will cover the Fundamental Theorem of Binary Classification, a result that links the complexity of a hypothesis class to the feasibility of learning within that class. In essence, the theorem implies that if the VC dimension of a hypothesis class is finite, there exists an algorithm that can learn any target function within that class with high probability, given a sufficient number of training examples. Conversely, if the VC dimension is infinite, no PAC learning algorithm can guarantee learning the class within the PAC framework, regardless of the number of samples.

2.3.1 PAC-learnability and the VC dimension

Let us recall that for a given hypothesis class $\mathcal{F}$ of a learning algorithm $\mathcal{A}$, the hypothesis $h_{\mathcal{F}}$ was defined in equation 2.4 as the best possible hypothesis within $\mathcal{F}$. Now, considering the hypothesis h_S returned by $\mathcal{A}$, and given the sample size N of S, we seek to evaluate how well the algorithm approximates $h_{\mathcal{F}}$ by examining the deviation $\epsilon \in (0,1)$, referred to as the *error tolerance*, between their respective true risks. Due to the inherent randomness in the selection of the sample S, we can only make a *probabilistic statement*, estimating this deviation using a *confidence parameter* $\delta \in (0,1)$. We present the following definition.

Definition 2.10 (Agnostic PAC-learning)**.** *[23] Given a set $\mathcal{F}$ of achievable hypotheses by an algorithm $\mathcal{A}$, we call $\mathcal{A}$ an $\mathcal{F}$-agnostic PAC-learning algorithm if and only if there exists a polynomial p such that:*

$$\forall \epsilon, \delta \in (0,1) : \forall \mathbb{P}_{\mathcal{X}\times\mathcal{Y}} : \forall N \geq p\left(\frac{1}{\epsilon}, \frac{1}{\delta}\right) : \Pr_{S\sim\mathbb{P}^N}(R[h_S] - R_{\mathcal{F}} \geq \epsilon) \leq \delta. \tag{2.9}$$

The difference that defines the event in the statement above was previously discussed in equation 2.6. There, we observed that an upper bound is related to another important quantity known as the *generalization gap*, which for a given hypothesis h is expressed as

$$|R_S[h] - R[h]|\,.$$

The bounds on this quantity are known in the literature as *generalization bounds*. As discussed in the closing remarks of section 2.2, this difference tends to increase with the complexity of the hypothesis class $\mathcal{F}$, primarily due to overfitting. But how should we define this complexity? A natural suggestion might be to consider the cardinality of the hypothesis space, $|\mathcal{F}|$. However, this approach is often inadequate. For instance, it is easy to imagine scenarios where this cardinality doesn't exist or is infinite, even in the simplest cases.

Consider, for example, a classification task with a one-dimensional feature space $\mathcal{X} = \mathbb{R}$, where $\mathcal{H}$ represents the set of all possible threshold functions. Since each threshold corresponds to a real number, the hypothesis space is uncountably infinite, making a cardinality-based bound ineffective. Furthermore, even if $\mathcal{H}$ were finite, selecting from a finite set of thresholds could lead to redundancies, as different thresholds might result in identical labelings, thus allowing for too loose bounds.

In practice, however, any hypothesis space $\mathcal{F}$ will be finite due to the discretization of data and the limited storage capacity of the device in use. However, this remains unsatisfactory because the number of possible hypotheses is often astronomically large, given the nature of the problems we typically face. Therefore, we must consider alternative measures, with the VC dimension emerging as a particularly suitable indicator. The remainder of this chapter will focus exclusively on the binary classification task.

Definition 2.11 (Bit sequences sets)**.** *[14] Let $\mathcal{F}$ be a hypothesis space over a feature space $\mathcal{X}$ and let $C \subseteq \mathcal{X}$, $C = \{x_1, \dots, x_r\}$. We call $\mathcal{F}_{|C}$ the set of r-bit sequences (also known as dichotomies)*

$$\mathcal{F}_{|C} := \{(h(x_1), \dots, h(x_r)) \mid h \in \mathcal{F}\} \subseteq \{0,1\}^r.$$

Moreover, we say that the set C is shattered by $\mathcal{F}$ if and only if $\mathcal{F}_{|C} = \{0,1\}^r$.

Example 2.12. *Let $\mathcal{X} = \mathbb{R}$ and let $\mathcal{F}$ be the set of all possible threshold functions parameterized by real numbers. For any set $C = \{a, b\}$ where $a < b$, the set of all possible 2-bit sequences that can be realized is $\{(0,0), (0,1), (1,1)\}$. On the other hand, any singleton $C = \{a\}$ is shattered by this $\mathcal{F}$.*

Definition 2.13 (Growth function)**.** *[23] Let $\mathcal{F}$ be a hypotheses set over feature space $\mathcal{X}$. The growth function $m_{\mathcal{F}} : \mathbb{N} \to \mathbb{N}$ is defined as*

$$m_{\mathcal{F}}(n) = \max_{\substack{C \subseteq \mathcal{X} \\ |C| = n}} \left|\mathcal{F}_{|C}\right|.$$

The growth function captures the *effective* number of hypotheses in $\mathcal{F}$. In terms of the example described above, given any $n = 2$ points on the real line, the effective number of hypotheses is 3, since, for example, the infinity of thresholds $t > b$ that produce the $(0,0)$ sequence are counted as one.

Definition 2.14 (VC dimension)**.** *[23] Let $\mathcal{F}$ be a hypotheses set over feature space $\mathcal{X}$. The VC dimension* $\mathrm{VC}(\mathcal{F})$ *is defined as*

$$\mathrm{VC}(\mathcal{F}) := \begin{cases} \max\{n \in \mathbb{N}_0 \mid m_{\mathcal{F}}(n) = 2^n\} & \text{if it exists,} \\ \infty & \text{otherwise.} \end{cases}$$

In other words, since 2^n is the maximum number of possible binary n-sequences, we say that $\mathcal{F}$ has $\mathrm{VC}(\mathcal{F}) = n$ if there exists at least one subset of n elements that is shattered by $\mathcal{F}$, and no subset of $n+1$ elements can be shattered by $\mathcal{F}$.

We are going to cite a result that provides a combinatorial bound on the growth function and that allows us to see the VC dimension from another point of view. The proof can be found on pages 74–75 in [17].

Lemma 2.15 (Sauer's lemma)**.** *[17] Let $\mathcal{H}$ be a hypothesis class with $d := \mathrm{VC}(\mathcal{H}) < \infty$. Then, for all $n \in \mathbb{N}$,*

$$m_{\mathcal{H}}(n) \leq \sum_{i=0}^{d} \binom{m}{i}. \tag{2.10}$$

Moreover, if $n > d+1$, then

$$m_{\mathcal{H}}(n) \leq \left(\frac{en}{d}\right)^d. \tag{2.11}$$

As a corollary to Sauer's lemma, the growth function can be expressed as follows:

$$m_{\mathcal{F}}(n) \begin{cases} = 2^n, & \text{if } n \leq d, \\ \leq \left(\frac{en}{d}\right)^d, & \text{if } n > d. \end{cases}$$

This allows us to interpret the VC dimension as the transition point from an exponential behaviour in the subset size n (of the effective number of hypotheses) to a polynomial one. Next, we will provide some examples of VC dimension computations.

Example 2.16 (Fixed slope linear classifiers)**.** *Consider $\mathcal{X} = \mathbb{R}^2$ and the hypothesis class $\mathcal{F}_m = \{h_c(x) = w \cdot x + c \mid c \in \mathbb{R}\}$ for some fixed slope $w \in \mathbb{R}^2$. Any singleton $C = \{x_1\}$ can be shattered by $\mathcal{F}_m$, but this no longer holds for more than one point. This is because the restriction of being allowed to only move vertically induces an order relation that makes the problem equivalent to the one shown in Example 2.12. More concretely,*

$$\forall c_1, c_2 \in \mathbb{R}^2 : c_1 \preceq_w c_2 \Leftrightarrow |w \cdot c_1| \leq |w \cdot c_2|.$$

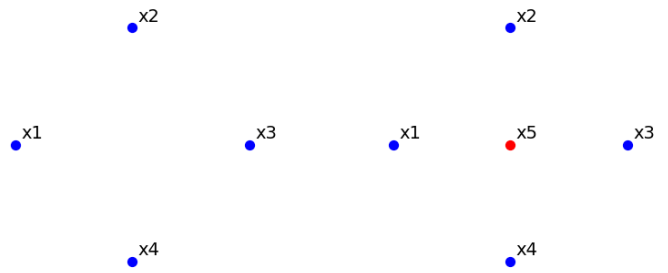
Hence, $\mathrm{VC}(\mathcal{F}_m) = 1$.

The previous example is relevant for the discussion concerning the second approach that we suggest in Chapter 4. There we make use of this geometric constraint (fixed slope) in order to justify the remarkable efficiency of the adaptation method described at the end of the next chapter.

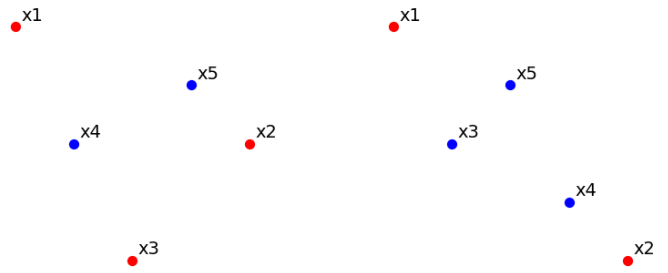
Example 2.17 (Axis-aligned rectangles in $\mathbb{R}^2$)**.** *Consider a hypothesis set of axis-aligned rectangles* $\mathcal{F}_{\text{rec}}$*, i.e., hypotheses of the form*

$$h_{a_1a_2b_1b_2}(x) = \begin{cases} 1, & \text{if } (x^1 \in [a_1, a_2]) \wedge (x^2 \in [b_1, b_2]), \\ 0, & \text{otherwise.} \end{cases}$$

We can always identify a set $C = \{x_1, x_2, x_3, x_4\}$ *of four distinct points where* x_1 *and* x_2 *have the extreme values in the* x*-dimension, and* x_3 *and* x_4 *have the extreme values in the* y*-dimension. It is straightforward to see that the set* C *can be shattered by* $\mathcal{F}_{\text{rec}}$*. However, for any set of five points, one of three situations occurs. First, it might be possible to select four distinct points, each representing an extremal component, with the fifth point* x_5 *necessarily lying within the convex hull formed by these four points, making the labeling* $(1, 1, 1, 1, 0)$ *unachievable. Second, if one point* x_1 *has both components as extremal values and two points* x_2 *and* x_3 *each have one extremal component, then the dichotomy* $(1, 1, 1, 0, 0)$ *cannot be realized. Lastly, if only two points* x_1 *and* x_2 *account for all four extremal components, then the sequence* $(1, 1, 0, 0, 0)$ *becomes impossible. Examples of these scenarios are illustrated in Figure 2.1. Consequently, the VC dimension of* $\mathcal{F}_{\text{rec}}$ *is* $\text{VC}(\mathcal{F}_{\text{rec}}) = 4$*.*



(a) All dichotomies possible (b) $(1, 1, 1, 1, 0)$ impossible



(c) $(1, 1, 1, 0, 0)$ impossible (d) $(1, 1, 0, 0, 0)$ impossible

Figure 2.1: Examples of subsets in the VC computation of axis aligned rectangles.

The VC dimension is notoriously difficult to compute. However, there are a few results that make its computation straightforward, such as the following:

Theorem 2.18 (VC dimension for linear spaces)**.** *[23] Let $\mathcal{G}$ be a finite-dimensional $\mathbb{R}$-vector space of functions of the form $g : \mathcal{X} \to \mathbb{R}$. Let $\phi : \mathcal{X} \to \mathbb{R}$ be an additional fixed function. Then, for some threshold function $\tau : \mathbb{R} \to [0, 1]$, the hypothesis space*

$$\mathcal{F} := \{x \mapsto \tau\left(\phi(x) + g(x)\right) \mid g \in \mathcal{G}\} \subseteq \{-1, +1\}^{\mathcal{X}}$$

has VC dimension

$$\mathrm{VC}(\mathcal{F}) = dim_{\mathbb{R}}(\mathcal{G}).$$

Coming back to the generalization gap, we can now cite one of the most important results in the theory of learning. The proof can be found in the Appendix of [24].

Theorem 2.19 (VC generalisation bound)**.** *[14] Let $\mathcal{H}$ be a hypothesis class with VC dimension d. Consider a sample S of size N drawn independently from a distribution $\mathbb{P}$ over the feature space. For any hypothesis $h \in \mathcal{H}$, holds with probability at least $1 - \delta$:*

$$R[h] \leq \hat{R}_S[h] + \sqrt{\frac{8d \log\left(\frac{N}{d}\right) + 8 \log\left(\frac{4}{\delta}\right)}{N}}.$$

The power of this statement lies in its generality. It proves that every hypothesis from a class with finite VC dimension will eventually generalize well, even if $|\mathcal{H}| = \infty$. The reverse of the medal is that the bound proves to be too loose in most practical applications. Nevertheless, it can still serve as a useful rule of thumb in comparing models, owing to the fact that smaller VC dimensions lead to better generalization.

Now that we are familiarised with probabilistic statements involving learnability and VC dimensions, we will discuss a fundamental result in the next subsection.

2.3.2 The fundamental theorem of binary classification

We complete our mathematical discussion with the most important theoretical result concerning the task of binary classification:

Theorem 2.20 (Fundamental theorem of binary classification)**.** *[23] Let $\mathcal{F} \subseteq \{-1, 1\}^{\mathcal{X}}$ be a hypothesis class, S a sample of size N drawn according to $\mathbb{P}^N$ and R the 0-1 risk. Then, the following are equivalent:*

I. $\mathrm{VC}(\mathcal{F}) < \infty$.

II. There is a polynomial $p\left(\frac{1}{\delta}, \frac{1}{\epsilon}\right)$ such that for any measure $\mathbb{P}$ and pair $(\epsilon, \delta) \in (0, 1)^2$:

$$N \geq p\left(\frac{1}{\delta}, \frac{1}{\epsilon}\right) \Rightarrow \mathbb{P}^N[\exists h \in \mathcal{F} : |\hat{R}_S[h] - R[h]| \geq \epsilon] \leq \delta. \tag{2.12}$$

III. There is a polynomial $p\left(\frac{1}{\delta}, \frac{1}{\epsilon}\right)$ *and* $\mathcal{A} : S \mapsto h_S$ *such that for any measure* $\mathbb{P}$ *and pair* $(\epsilon, \delta) \in (0,1)^2$*:*

$$N \geq p\left(\frac{1}{\delta}, \frac{1}{\epsilon}\right) \Rightarrow \mathbb{P}^N[|\hat{R}_S[h_S] - R_{\mathcal{F}}| \geq \epsilon] \leq \delta. \tag{2.13}$$

IV. There is a polynomial $p\left(\frac{1}{\delta}, \frac{1}{\epsilon}\right)$ *such that for any measure* $\mathbb{P}$*, any pair* $(\epsilon, \delta) \in (0,1)^2$ *and any empirical risk minimiser* h_S^{ERM}*:*

$$N \geq p\left(\frac{1}{\delta}, \frac{1}{\epsilon}\right) \Rightarrow \mathbb{P}^N[|\hat{R}_S[h_S^{\mathrm{ERM}}] - R_{\mathcal{F}}| \geq \epsilon] \leq \delta. \tag{2.14}$$

A comprehensive proof can be found in [17]. This result naturally fuses the concepts of learnability and VC dimension, as the theorem implies that a hypothesis class with a finite VC dimension is PAC-learnable. In other words, there exists a learning algorithm that can, with high probability, produce a hypothesis whose risk is close to the minimum possible within the class, given a sufficiently large sample size. Conversely, if the VC dimension is infinite, the class is not PAC-learnable, as no polynomially bounded sample size can guarantee such performance.

The theorem also provides a bound on the sample complexity required to achieve a specified accuracy and confidence level in learning, which is directly tied to the VC dimension. Specifically, the number of samples N needed grows polynomially with $1/\epsilon$ and $1/\delta$, and is also influenced by the VC dimension of the hypothesis class. This relationship between sample complexity and VC dimension is fundamental to understanding the feasibility of learning within the PAC framework.

Chapter 3

The domain adaptation problem

We aim to introduce the domain adaptation problem using the terminology developed in Chapter 2. In that chapter, we assumed the existence of a probability measure $\mathbb{P}$ on the product space $\mathcal{X} \times \mathcal{Y}$, which governed the data generation process. A natural relaxation of this assumption allows the distribution to vary when transitioning from training to testing. This approach directly accounts for any significant observed decline in the generalization performance of a machine learning algorithm. For instance, a straightforward method to test for the presence of such a distribution shift, beyond simple visualization techniques, is to train a classifier that distinguishes between the two datasets. Based on the classifier's performance, one can determine whether applying domain adaptation (DA) methods could mitigate the identified discrepancy or if other factors are affecting the model's behavior. The general framework for addressing this type of problem is known as Transfer Learning (TL) and for a comprehensive overview of the field, we recommend consulting the survey by Pan et al. [25].

As a first step towards generalization, we define objects called *domains*, which are tuples denoted as $\mathcal{D}_i = (\mathcal{X}_i, \mathbb{P}_{\mathcal{X}_i})$, consisting of a feature space $\mathcal{X}_i$ and a (marginal) distribution $\mathbb{P}_{\mathcal{X}_i}$. Similarly, a learning task, particularly in a deterministic supervised learning scenario, can be described as a tuple $\mathcal{T}_j = (\mathcal{Y}_j, c_j)$. The domains and tasks used in the training process are referred to as *source domain(s)* and *source task(s)*, while their counterparts in the testing scenario are referred to as *target domain(s)* and *target task(s)*. The indices i and j represent positive integers corresponding to the number of such domains or tasks. However, since we are only considering one of each, it is notationally more convenient to use $i, j \in \{s, t\}$.

Using these definitions for domains and tasks, there are multiple possible ways in which they can differ. Based on the categorization of TL techniques indicated in [25], what we will focus exclusively on is a specific setting known as *transductive transfer learning*, as is going to become evident from the description of the toy model in section 3.2. For a visual representation of how DA fits within the broader field of transfer learning and its relationship to the setup discussed in Chapter 2, please refer to Figure 3.1.

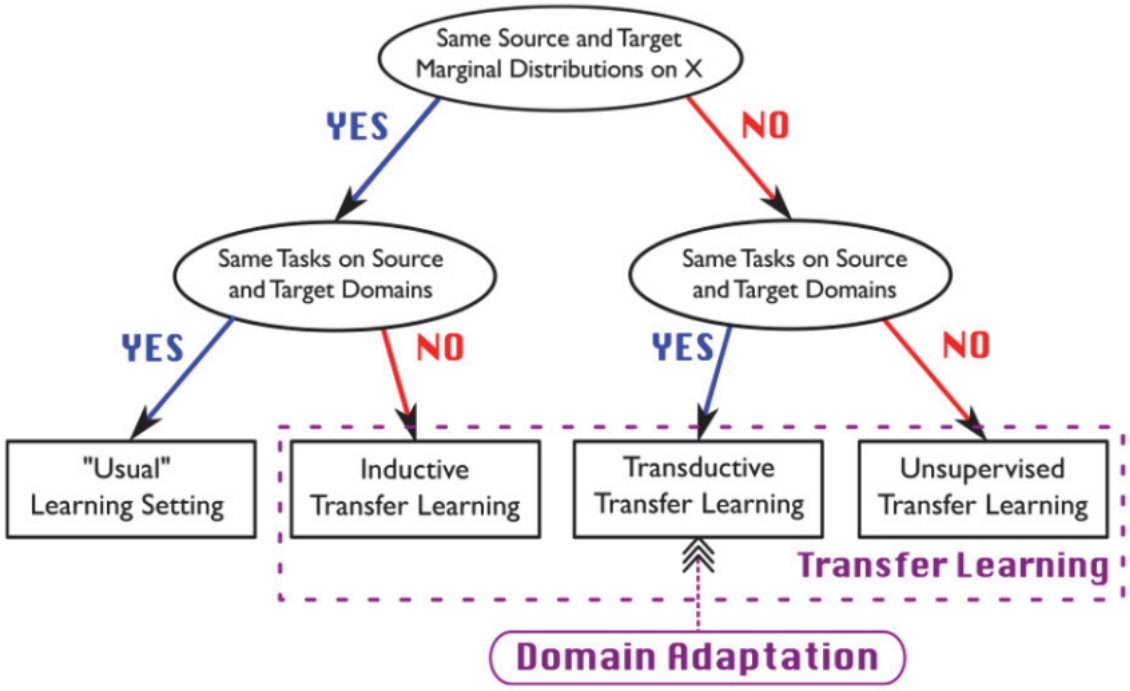


Figure 3.1: The DA problem in the TL framework. The mathematical setup of Chapter 2 concerns only the node outside the purple box. Taken from [26].

3.1 Real-life example

We begin by discussing a challenge encountered in the real-world application of a classification algorithm in a disease detection study.

Consider an experimental group that possesses a set of N ideally preserved blood samples. These samples were collected as part of a case-control study aimed at detecting a specific disease. Each blood sample i is labeled with a value y_i, indicating the presence or absence of the disease: "1" for sick patients (cases) and "0" for healthy patients (controls).

Furthermore, suppose the group has a measurement device that captures all relevant characteristics of a blood sample as a sequence of numbers, such as the absorbance at each wavenumber of a laser pulse directed at the sample. These measurements can be organized into a feature vector x_i, forming a pair (x_i, y_i) that represents the disease status of patient i based on the physical properties of their blood. The key assumption is that the device captures sufficient information to be used in a subsequent supervised classification task.

The group conducts an initial round of measurements at time t_1 for the N samples. The data collected by the device is used to train a logistic regression model, which achieves a ROC-AUC of approximately 0.92 on a held-out portion of the samples that were not used during training. Later, the group performs a second round of measurements at time t_2.

The practitioners aim to use the previously trained model to make predictions on the new measurements of the *same samples*. However, when evaluated using the same metric, the new ROC-AUC drops to around 0.82. This 10% reduction is attributed to a shift in the probability distribution of the measured features, inherent to the measurement process. A similar pattern is observed when using two distinct devices, which are presumed to be as similar as possible. This is not surprising, as treating a single device at two different time points can effectively be considered equivalent to using two different devices in terms of the measurement process.

Addressing this domain adaptation challenge is essential in areas where the reliability of a model can significantly impact human lives, such as in disease detection or self-driving cars.

3.2 The toy model

The aim of this section is to create a conceptually straightforward and easy-to-visualize toy model that clearly demonstrates the domain adaptation (DA) problem introduced above. To achieve this, we generate two datasets that differ by an affine shift, resulting in a manageable change in their underlying distributions. The data drift will be physically described in alignment with the previous experiment.

We begin by simulating an approximately linearly separable dataset of size $N = 1000$, sampled from two bivariate Gaussian distributions, $\mathcal{G}(\mu_1, \Sigma_1)$ and $\mathcal{G}(\mu_2, \Sigma_2)$, corresponding to the two classes: controls ($y = 0$) and cases ($y = 1$). For convenience, we artificially added a "constant-1 feature" to account for the bias term in the model. Consequently, in our toy model, we consider a source domain $\mathcal{D}_s = (\mathcal{X}_s, \mathbb{P}_{\mathcal{X}_s})$, where $\mathcal{X}_s \cong \mathbb{R}^2 \times \{1\} \cong \mathbb{R}^2$, and a Gaussian mixture distribution given by

$$\mathbb{P}_{\mathcal{X}_s} = \frac{1}{2}\mathcal{G}(\mu_1, \Sigma_1) + \frac{1}{2}\mathcal{G}(\mu_2, \Sigma_2).$$

We then applied an affine transformation $\Delta : \mathcal{X}_s \to \mathcal{X}_s$ of the form

$$\Delta : x \mapsto \lambda R_{\vec{\varphi}}(x) + \mathbf{t},$$

representing the shift experienced by the source dataset, where $\lambda \in \mathbb{R}$ is a scaling parameter, $R_{\vec{\varphi}}$ denotes a rotation parameterized by $\vec{\varphi} \in \mathbb{R}$, and $\mathbf{t} \in \mathbb{R}^2$ is a constant translation. The resulting scenario is depicted in Figure 1a in Appendix A. The physical interpretation that connects the toy model to the real-life setup begins with the unknowable objective characteristics of each measurement sample. Their objective existence is posited by considering an oracle that possesses the ground truth information the devices aim to retrieve. These characteristics are transformed into features corresponding to the source and target distributions through inaccessible mappings, $\tilde{D}$ and $\overline{D}$, which represent the two measurement processes. Assuming these transformations are invertible, the shift Δ can be

expressed through an appropriate composition. The resulting scheme is depicted in Figure 3.2.

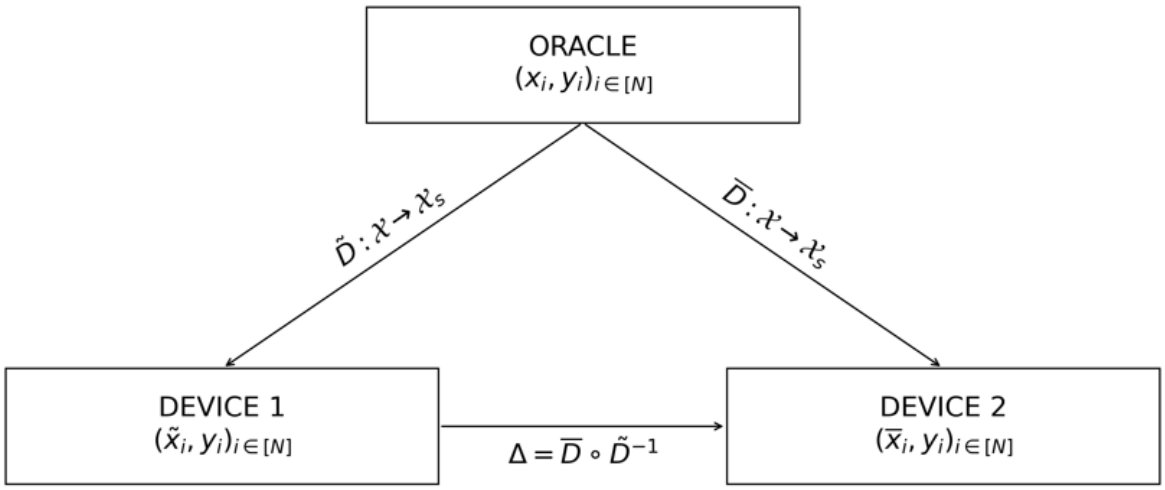


Figure 3.2: Physical interpretation of the drift.

We trained a logistic regression model (Figure 3.3) on the source dataset to determine the parameters of the separating plane, specifically the components of the model's weight vector. Given the properties of the cross-entropy loss function,

$$l_{\mathrm{CE}}(\hat{y}, y) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})), \tag{3.1}$$

where

$$\hat{y} = \hat{y}(w, x) = \frac{1}{1 + \exp(-w \cdot x)} \tag{3.2}$$

is the predicted output before applying a threshold function, the resulting optimization problem is convex. As a result, gradient descent is guaranteed to converge to the global minimum. Accordingly, we selected a small set of hyperparameters that achieved good in-distribution accuracy (approximately 96%): a learning rate of $\alpha = 0.1$, number of iterations $n_{\mathrm{iter}} = 500$, and train-heldout ratio $\mathrm{thr} = 0.8$. After training, the output hypothesis is given by

$$h_{\tilde{a}}(x) = \tau(\tilde{a} \cdot x),$$

where $\tilde{a} \in \mathbb{R}^3$ represents the learned parameters and $\tau : (0, 1) \to \{0, 1\}$ is the composition of the threshold function

$$\tau_{0.5}(x) = \begin{cases} 1 & \text{if } x > 0.5, \\ 0 & \text{otherwise,} \end{cases}$$

with the sigmoid function defined in Eq. 3.2. Since the resulting plane has virtually no predictive power on the target data, we employ an adaptation method introduced in [27]. This is the focus of the next section.

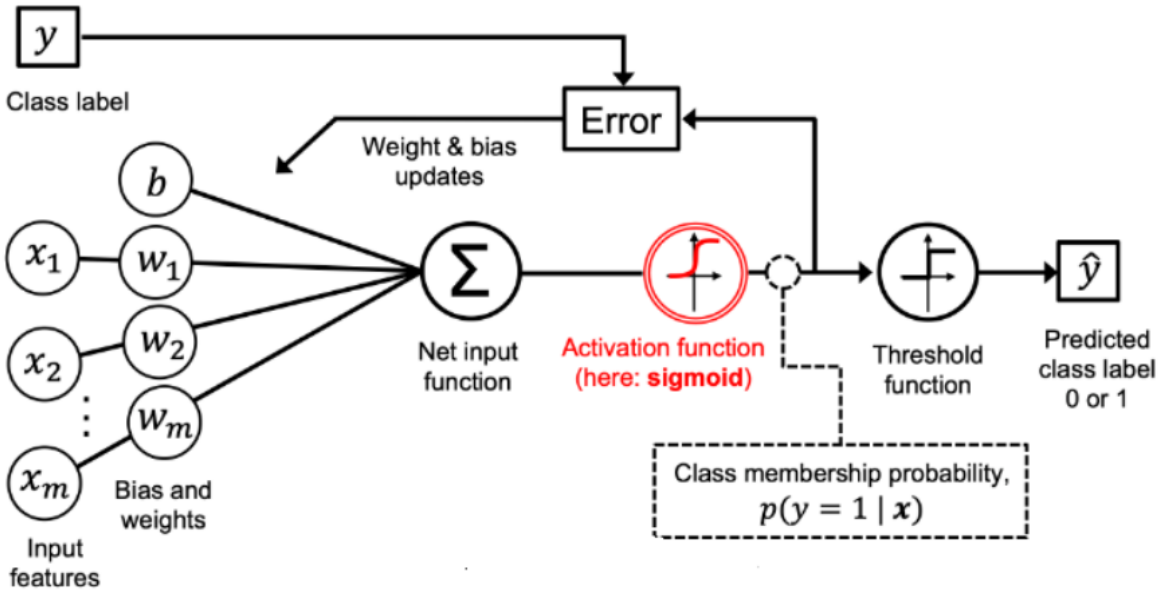


Figure 3.3: The design of a generic logistic regression model, taken from [28].

3.3 The FEDA method

The "Frustratingly Easy Domain Adaptation" (FEDA) method seeks to enhance performance on target data by making straightforward modifications to the feature space. This approach falls under the category of *feature-based* methods, as opposed to *instance-based* and *parameter-based* techniques, which focus on reweighting labeled training data and directly adjusting model parameters, respectively. The original paper introduces it in the context of a natural language processing task, specifically sentiment classification. Despite achieving better-than-baseline performance, a rigorous explanation for its success is still lacking, as highlighted by the author in a subsequent paper [29]: "Theoretically analyzing the superior performance of EA and EA++ and providing generalization guarantees is an interesting line of future work." and "It would be interesting to formally frame these approaches and see whether their empirical performance can be justified within a theoretical framework."

Feature adaptation in FEDA is achieved through two *augmentation maps*, $\tilde{\phi}$ and $\overline{\phi}$:

$$\tilde{\phi} : x \mapsto (x, x, \mathbf{0}),$$
$$\overline{\phi} : x \mapsto (x, \mathbf{0}, x).$$

They map each feature vector from their respective domains to a tripled version, where each of the three components represents a different type of general feature. Intuitively, the first component captures features common to both domains, the second component is reserved for source-specific features (hence $\overline{\phi}$ maps it to $\mathbf{0}$), and the third component is dedicated to target-specific features (hence $\tilde{\phi}$ maps it to $\mathbf{0}$).

As with any adaptation method, the central question is how this approach compares in performance to fully retraining on the target data, assuming full retraining is feasible. Specifically, given access to a small annotated subset $S_t^M \subseteq S_t$ of target data of size $M := \gamma N$, where $\gamma \ll 1$, can a clever manipulation of the feature space help reduce the gap in model accuracy? Although our toy model provides access to all labeled data in the target domain, we will use this as a reference scenario (II) only. To evaluate the adaptation technique, we developed additional learning scenarios (I, III, IV) based on different training sets, selecting the most effective one for detailed analysis in the next chapter. The scenarios are as follows:

I. Train on source and annotated target $S_s \cup S_t^M$

II. Fully retrain on target S_t

III. Train on augmented annotated target $\overline{\phi}(S_t^M)$

IV. Train on all augmented source data and annotated target data $\tilde{\phi}(S_s) \cup \overline{\phi}\left(S_t^M\right)$

The form of the learned hypothesis in the augmented cases differs as follows. To leverage the previously learned information, represented by $\tilde{a}$, we fix it as the first component of the new transformation in both cases. In scenario III, we set the middle component to be a constant zero vector, allowing only $\bar{c} \in \mathbb{R}^3$ to be updated. Scenario IV follows the same approach, except that the middle component is not constrained and is allowed to vary freely.

Each scenario is repeated three times, using different fractions of target points: 5%, 10% and 20% . The performance of the models is summarized in Figure 2, Appendix A. It is evident that the third model achieved results closest to full retraining, with an impressive accuracy of approximately 95%, nearly identical to the initial hypothesis on the source data. The two decision boundaries are shown in Figure 1b. In the next chapter, we will explore the effectiveness of this method for such small values of γ, both theoretically and heuristically.

Chapter 4

The effectiveness of FEDA

We would like to return now to the question formulated at the end of Chapter 3. We will focus our efforts on justifying the efficiency of the particularly simple adaptation method described, using the tools provided by the PAC-learning framework.

4.1 A first approach

The third (III) learning scenario of 3.3 showed that only a small fraction γ of target points is needed in order to match the performance of a full retraining. An intuitive explanation would imply that there is no need for a large number of training points, since the model capitalizes on the information already learned in the source training. Consequently, this second learning problem should be *less complex* than the original one, and we hope to capture this behavior in the VC dimensions of the following hypothesis classes:

I. $\mathcal{H}_{\text{hp}} = \{\tau \circ A_a \mid A_a : x \mapsto \mathbf{a} \cdot x + a_0,\ (a_0, \mathbf{a}) =: a \in \mathbb{R}^{\mathfrak{d}+1}\}$

II. $\mathcal{H}^a_{\mathfrak{D}^{-1}} = \{\tau \circ A_a \circ \Delta \mid \Delta \in \mathfrak{D}^{-1}\}$

where $\mathfrak{d} = \dim_{\mathbb{R}} \mathcal{X}$ and $\mathfrak{D}^{-1}$ is the set of inverse drifts, assuming that all admissible data shifts are invertible, which is indeed the case for our toy model. Denoting by D the quantity $\text{VC}(\mathcal{H}_{\text{hp}})$ and by d the quantity $\text{VC}(\mathcal{H}^a_{\mathfrak{D}^{-1}})$, our hope is to find an expression of the form

$$\gamma = \gamma\left(\frac{d}{D}\right). \tag{4.1}$$

Using the result on the VC dimension of linear spaces of functions from Theorem 2.18, we infer that the casual hyperplane learning problem I has $D = \mathfrak{d} + 1$. Upon further investigation of the second class $\mathcal{H}^a_{\mathfrak{D}^{-1}}$, one realizes that the chosen form of Δ provides enough "freedom" in adjusting the initially learned hyperplane a that the two classes become equivalent. Hence, $\text{VC}(\mathcal{H}_{\text{hp}}) = \text{VC}(\mathcal{H}^a_{\mathfrak{D}^{-1}})$, and an expression of the form in Eq. 4.1 would have no explanatory power, since in this setup, we expect $\gamma \approx 1$.

Nevertheless, we believe that finding such an expression for γ can still be instructive, as it becomes relevant when considering models other than logistic regressors, such as neural networks. In such cases, it will typically be true that $D \gg d$, allowing us to draw meaningful conclusions about the FEDA method, particularly regarding the learning scenarios in which it ceases to be efficient. Therefore, the first step is to define a measure of efficiency, which requires a clearly specified *goal* and a *threshold*.

Suppose that an initial source training yields a hypothesis h_{I} with performance $\hat{R}_s[h_{\mathrm{I}}]$. The goal is to implement a method that returns a model h_{II} with empirical performance $\hat{r}_t[h_{\mathrm{II}}]$ satisfying

$$\hat{R}_s[h_{\mathrm{I}}] \approx \hat{r}_t[h_{\mathrm{II}}]. \tag{4.2}$$

As is usually the case in DA problems, there is an associated cost c for labeling a single target data point. Depending on an available budget B, this goal should be achieved by spending as few resources as possible. We can then call an adaptation method *efficient* by employing a threshold of efficiency, say $\gamma_{\mathrm{thr}} = 20\%$, representing a maximum fraction of the budget spent by an efficient method. For example, in our real-life application of Section 3.1, B could be given by the worst possible scenario, i.e., a full retraining. Taking $c = 10$ \$ makes the cost of acquiring a labeled target dataset of size $M = N = 1000$ be $C := Mc = 10{,}000$ \$, which is the size needed for a full retraining to satisfy condition 4.2. We have empirically determined that in this example, FEDA is more than efficient (Figure 2), achieving a performance satisfying 4.2 with only $\gamma = 5\% < \gamma_{\mathrm{thr}}$ or $M = 50$, yielding a cost of only $C = 500$ \$.

Of course, condition 4.2 alone does not rule out the possibility of a lucky choice of target data. Therefore, what we are really interested in is making sure that the true risks obey

$$R[h_{\mathrm{I}}] \approx r[h_{\mathrm{II}}]. \tag{4.3}$$

We know that both quantities are related to their empirical counterparts by generalisation bounds. Hence, one way to enforce this approximate equality is by making the error functions in the VC generalisation bound for binary classification (Theorem 2.19) match:

$$\epsilon(N, D, \delta) \overset{!}{=} \epsilon(M, d, \delta) = \epsilon(\gamma, d, \delta), \tag{4.4}$$

which is equivalent to solving for γ the following equation:

$$\sqrt{\frac{8D\log\left(\frac{N}{D}\right) + 8\log\left(\frac{4}{\delta}\right)}{N}} = \sqrt{\frac{8d\log\left(\frac{M}{d}\right) + 8\log\left(\frac{4}{\delta}\right)}{M}}. \tag{4.5}$$

This is because the number of source data points N is fixed, since an initial training cannot be bypassed. So is the confidence level δ associated to the probability with which the bound holds true, along with the VC dimensions D and d. Manipulating Equation 4.5 leads to the implicit equation

$$\gamma_\delta^N = \frac{d}{D}\left(\frac{N}{D}\right)^{\gamma_\delta^N D/d - 1}\left(\frac{4}{\delta}\right)^{\left(\gamma_\delta^N - 1\right)/d}. \tag{4.6}$$

Cast in this form, this quantity represents the fraction of target data needed to ensure good generalization performance (condition 4.3), given the well-performing (condition 4.2) hypotheses h_{I} and h_{II}, the confidence level δ, the size of the source dataset N, and the complexities of the learning tasks, D and d.

Returning to our original example, let us fix $\delta = 0.05$ and $\gamma_{\mathrm{thr}} = 20\%$. The numerical solution of Equation 4.6, obtained using the Newton-Raphson method, is shown in Figure 4.1 for several values of source data points N. This plot confirms the intuition that for similarly complex problems ($d/D \approx 1$), one needs about the same number of training points to achieve similar performances, regardless of how many samples were available in the initial training. In fact, the (small) N-dependence becomes apparent only in the lower and middle regimes, where once again we encounter an expected result: using more points results in a better fitted hyperplane. The method did not converge for larger values of N.

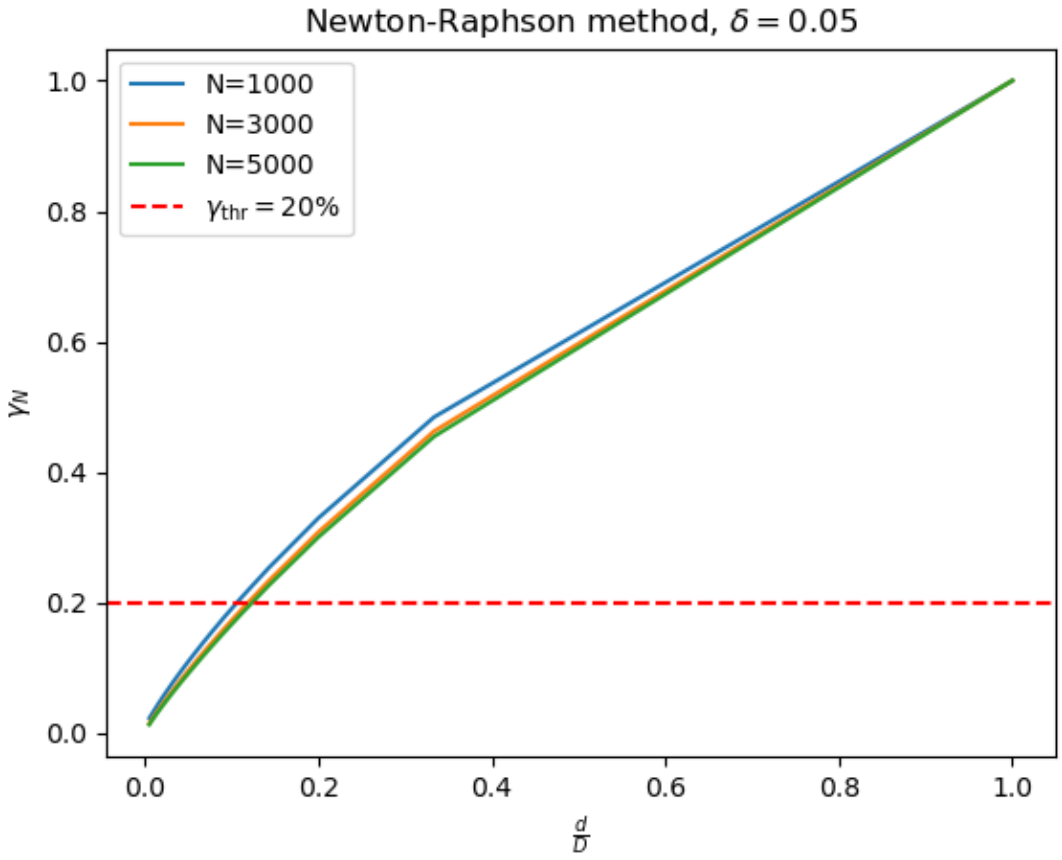


Figure 4.1: The behaviour of γ_δ^N in terms of the complexity ratio d/D.

Another conclusion that can be drawn from the behavior of γ_δ^N is that FEDA is guaranteed to be efficient, in terms of our (subjective) previously defined threshold, if the originally

trained model is at least 10 times more complex than learning an affine shift. This seems to be the case for a wide range of currently used, state-of-the-art classifiers. It remains, then, to explain this efficiency in our simple toy model as well.

4.2 A second approach

We will heuristically argue why only a small fraction of target data points is needed in our toy model to achieve source-matching performance, without relying on potentially difficult-to-compute VC dimensions. In order to illustrate the idea, we firstly restrict the set of possible affine shifts and allow only translations by a constant vector, as in Figure 4.2. The red decision boundary represents the initially learned hypothesis on the source dataset, while the magenta plane is the optimal separation in the target set.

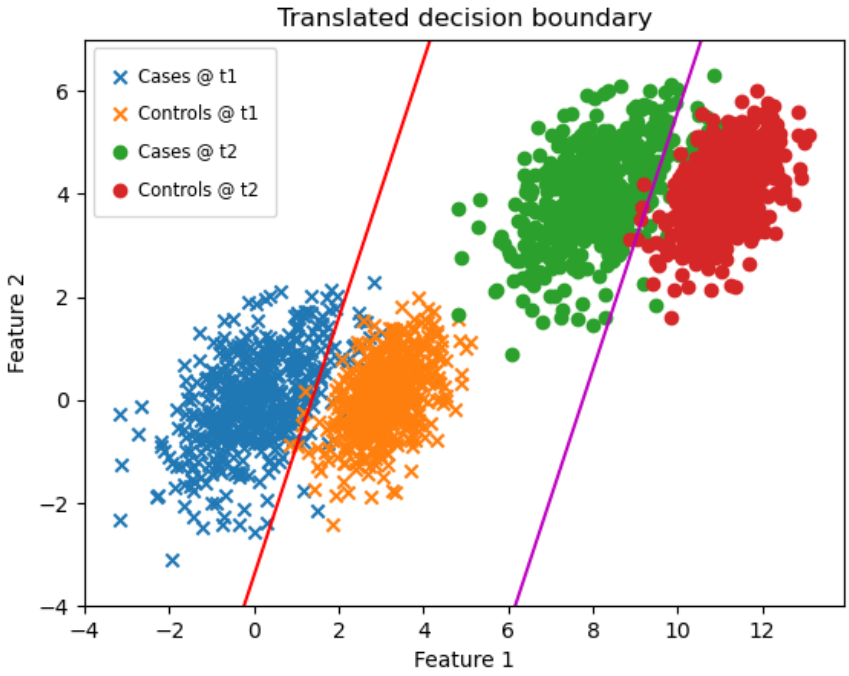


Figure 4.2: The target is the translation of the source by a constant amount.

Adaptation in this setup amounts to adjusting the bias term such that the initial plane converges to the optimal boundary. Since we do not expect the adaptation to be perfect, that is, to end up with the translated source boundary, we impose a convergence criterion of being within an $\epsilon > 0$ interval away from $h_{w^{(0)}}$. The adjustments are performed through a sequence of updates that begin by randomly (uniformly) selecting a target point and determining whether it is an outlier with respect to the current hypothesis. If the randomly selected point is not an outlier, i.e., it is correctly classified, then another point is selected.

If the new point is indeed an outlier, the update is performed such that the new plane now passes through the identified outlier. This process is repeated until the accuracy on the remaining target data either exceeds or falls within the tolerated value away from the source-level performance.

Mathematically speaking, the description above is equivalent to introducing a sequence of random variables $X_i^{(k)}$ that correspond to sampling a point x_i from the target distribution at every update step k, such that

$$X_i^{(k)}(x_i) = \begin{cases} 1, \text{if } h_{w^{(k)}}(x_i) \neq y_i \\ 0, \text{ otherwise.} \end{cases}$$

For a sufficiently large shift (such as the one in Figure 4.2), the initial probability of picking an outlier with respect to the source boundary $w^{(0)}$ is

$$p_0 := P\left(X_i^{(0)} = 1\right) = \frac{1}{2}, \tag{4.7}$$

since the numbers of cases and controls are equal and thus half of the test points are misclassified. This means that we initially expect to update the boundary after seeing at most two target points.

After the first update, the decision boundary improves, reducing the number of outliers. Let

$$p_k := P\left(X_i^{(k)} = 1\right)$$

denote the probability of selecting an outlier with respect to $w^{(k)}$. As the boundary becomes more aligned with the target distribution, p_k decreases. Furthermore, let Z_k be a random variable that corresponds to the number of points randomly chosen in the k-th update step. Then, the probability of needing m iterations until an outlier is picked is given by the geometric distribution

$$P(Z_k = m) = (1 - p_k)^{m-1} p_k. \tag{4.8}$$

Thus, we define the expected number of target points needed per update step k by

$$M_k := \lceil \mathbb{E}[Z_k] \rceil = \left\lceil \frac{1}{p_k} \right\rceil, \tag{4.9}$$

where the expected value is rounded up to the next integer value, and the last equality follows due to the well-known geometric series property for $|x| < 1$:

$$\sum_{n=1}^{\infty} n x^{n-1} = \frac{1}{(1-x)^2}. \tag{4.10}$$

Consequently, the number of points that need to be seen before the next update, which is inversely proportional to p_k, increases. If we denote the last update step by k^*, then the total number of expected target points,

$$M = \sum_{k=1}^{k^*} M_k,$$

will play the same role as in the previous section, i.e., the number of training target points required for a successful adaptation. If N denotes the number of source training points, the fact that $M \ll \gamma_{\text{thr}} N$ can be attributed to the rapidly decaying probabilities of finding an outlier with every update step k.

Figure 4.3 depicts an example of the approach discussed so far, in which only two updates ($k^* = 2$) had to be performed before converging to the optimal boundary. Only seven iterations ($M = 7$) were needed before the probability of detecting an outlier dropped enough (Figure 4.4) to achieve a performance comparable to the source one, leading to a fraction $\gamma_{k^*} = 0.7\% \ll \gamma_{\text{thr}}$.

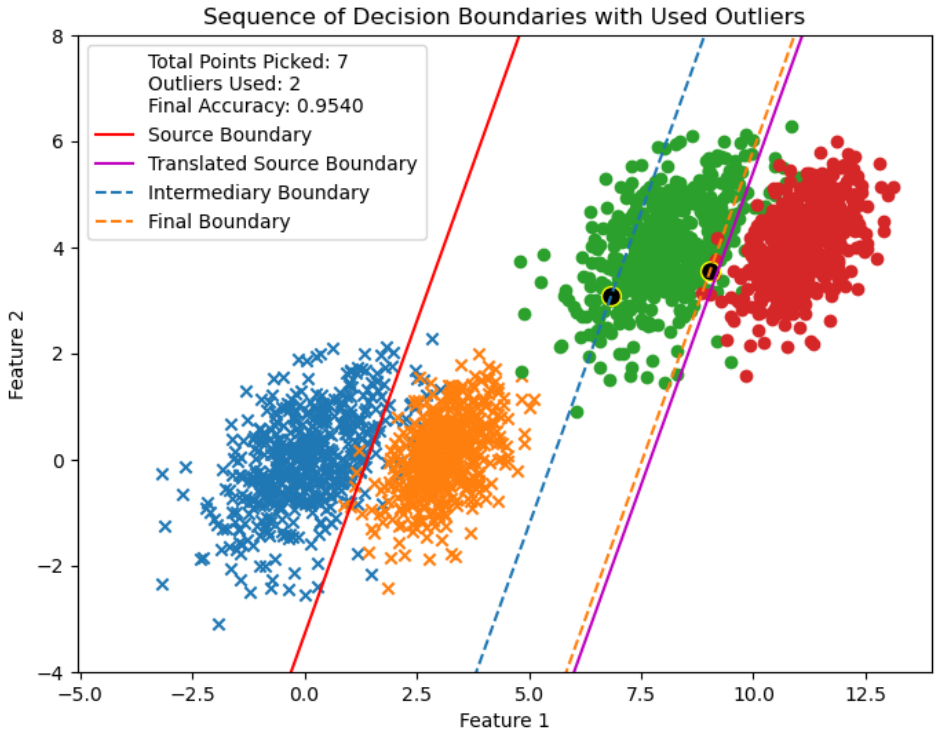


Figure 4.3: Example of a sequence of updates. The second boundary is already within an admissible tolerance value away from the optimal boundary.

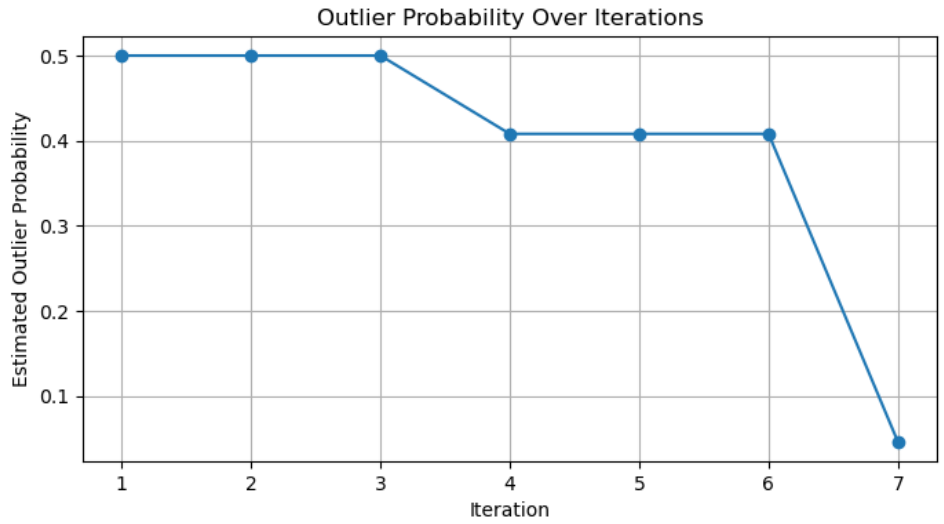


Figure 4.4: The probability of picking an outlier is constant within each update step.

We performed 1000 simulations in order to compute γ_{k^*}, twice. The results are shown in Figure 4.5 and Figure 4.6.

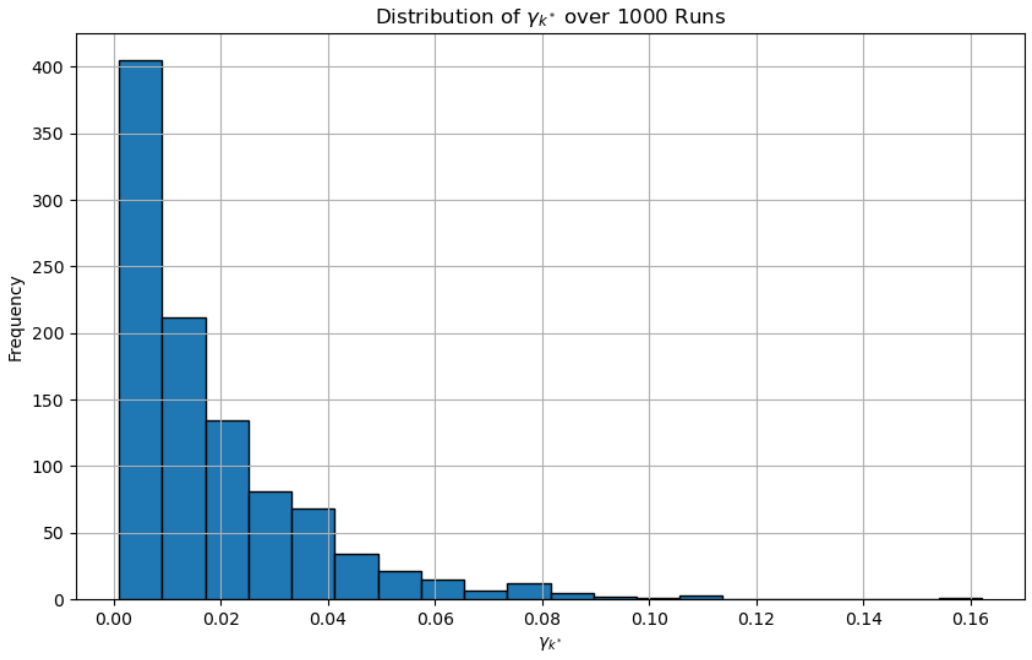


Figure 4.5: Histogram showing the distribution of γ_{k^*} over the 1000 runs.

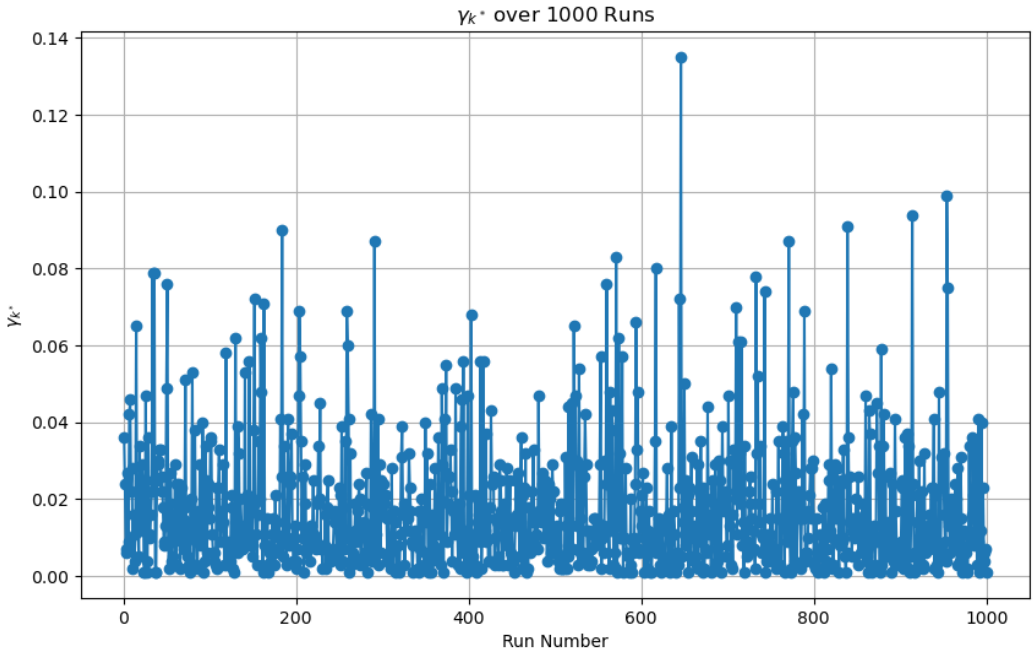


Figure 4.6: The values of γ_{k^*} over 1000 runs.

In summary, the rapid decrease in the estimated outlier probability p_k after each update leads to fewer necessary updates, which in turn keeps the total number of points seen M relatively low. Early updates, when relatively few points are needed per update, account for most of the boundary's improvement. This explains why, despite only sampling a small fraction of the target dataset, the adapted boundary can achieve a performance comparable to that of the source-level boundary.

This description of the adaptation process aligns with a constructivist viewpoint, as it begins with an initial boundary and iteratively tracks the update steps until convergence is achieved. However, there is an equivalent way of approaching this process by leveraging the simple relationship between the minimum accepted accuracy a_ϵ (corresponding to the boundary $h_{w^{(\epsilon)}}$) and the probability p_ϵ of detecting an outlier with respect to $h_{w^{(\epsilon)}}$:

$$a_\epsilon = 1 - 2p_\epsilon. \tag{4.11}$$

The above relationship arises because we assume that the selection of target data points is conducted according to a uniform distribution. We can then bypass the sequence of updates by looking directly at the number of points that need to be selected before we arrive at an outlier with respect to the desired boundary $h_{w^{(\epsilon)}}$. This can be represented by a random variable K satisfying

$$P(K = k) = (1 - p_\epsilon)^{k-1} p_\epsilon. \tag{4.12}$$

As we have seen before, the expected value of K is given by

$$\mathbb{E}[K] = \frac{1}{p_\epsilon}. \tag{4.13}$$

The cumulative probability of a variable obeying a geometric distribution with parameter p_ϵ is given by

$$P(K \leq N) = 1 - (1 - p_\epsilon)^N. \tag{4.14}$$

Given $\delta \in [0, 1]$, setting the rightside term of this equation to a specific value $1 - \delta$, i.e., the probability with which we would like our inequality to hold, enables us to derive a bound on N:

$$P(K \leq N) \geq 1 - \delta \Leftrightarrow (1 - p_\epsilon)^N \leq \delta \tag{4.15}$$

$$\Leftrightarrow N \ln(1 - p_\epsilon) \leq \ln \delta \tag{4.16}$$

$$\Leftrightarrow N \geq \frac{\ln \delta}{\ln(1 - p_\epsilon)}. \tag{4.17}$$

For $\delta = 0.05$ and an accuracy of $a_\epsilon = 90\%$, these results indicate that, on average, 20 target points are required to achieve an accuracy of 90%. Additionally, with 95% confidence, no more than 60 points are needed to attain the same performance. Considering that the number of source samples used was $N = 1000$, this explains why, for successful adaptation, the majority of the target fractions in the simulations fall well below 10%.

Chapter 5

Numerical experiments

This chapter delves into the numerical experiments conducted to analyze the behavior and performance of various domain adaptation methods implemented in the ADAPT package [1], an open source library implementing the main transfer learning and domain adaptation methods.

For a binary classification task (sex prediction) we tested 12 adaptation methods on a real-world dataset consisting of blood-based spectroscopic measurements. The measurements are part of a longitudinal sample collection framework of healthy individuals, with study code H4H_HU_2020_Sample Collection (approval reference number 2754-11/2020/EÜIG) and were performed by two identical devices in two different facilities, one in Garching (Germany) and one in Szeged (Hungary), resembling the situation described as a real-life example in Chapter 3. We split the dataset into four parts: the source data consists of measurements performed at the Szeged site, which are further divided into train and test sets, while target data is the one measured in Garching, again split into train and test sets.

An L2-regularized logistic regression model is fitted to the source training set. Its performance on both the within-site (source) and the cross-site (target) test sets is evaluated using two metrics: accuracy and ROC-AUC. These results serve as benchmarks for all subsequent adaptation methods. In order to construct a meaningful ranking of the methods, we have consistently used the same subset of target data in the adaptation process. As expected, some methods exhibited a more pronounced trade-off between source and target performance, which may not be desirable. Ideally, adaptation should maximize target performance while maintaining strong performance on the original source task. To encode this goal, we designed a custom performance metric that favors methods maximizing target performance while penalizing large discrepancies between target and source evaluations. For a given adaptation method i, the formula is defined as

$$CS(i) = w_{\text{acc}} \cdot acc_t(i) + w_{\text{auc}} \cdot auc_t(i) - w_{\text{d}} \cdot (|acc_t(i) - acc_s(i)| + |auc_t(i) - auc_s(i)|), \quad (5.1)$$

where acc_t (acc_s) is the accuracy on the target (source) test data, auc_t auc_s denotes the ROC-AUC score on the target (source) test data and w_{acc}, w_{auc} and w_{d} are the importance weights associated to each term in the equation. For example, if the primary interest is to

maximize the ROC-AUC score on the target test data, the weights for the accuracy and the discrepancy between source and target can simply be set to 0. The corresponding values of this metric are presented in the "Composite Score" column of Table 5.1, where, for the sake of simplicity, we used $w_{\text{acc}} = w_{\text{auc}} = w_{\text{d}} = 1$. The plots containing the accuracy and ROC-AUC comparisons for each method discussed are presented in Appendix B.

5.1 The ADAPT package

The algorithms provided in the ADAPT package aim to resolve problems related to transfer learning, such as sample bias, fine-tuning and domain adaptation. Following the logic of their implementation, the methods are categorized into three families: feature-based, instance-based and parameter-based. This classification is based on the nature of the transformation suffered by the base hypothesis (base estimator), i.e., the hypothesis learned initially using only the source data. Within the same family, the meta-structure of the adaptation algorithm is essentially the same, with possible differences coming from the particularities of the internal transformations, as well as from the potential usage of labeled target data.

Feature-based methods try to find a feature space transformation that aligns the source and target distributions in an encoded representation. The adapted estimator is then the hypothesis learned on the transformed data. On the other hand, instance-based methods look for suitable weights that aim to account for the differences in the two distributions. The weighting procedure is not considered an actual transformation of the dataset, but rather as an amplification of the individual losses of certain training instances. Finally, parameter-based methods transform the parameters of pre-trained models (fitted on training source data) such that the new estimator performs better on the target domain. In the following, we are going to briefly describe each method that we utilized in descending order of their respective composite scores. Their performances are summarized Table 5.1.

Table 5.1: Comparison of adaptation methods provided by the ADAPT package

Method Family	Adaptation Method	Composite Score	Source Test		Target Test	
			Accuracy	ROC-AUC	Accuracy	ROC-AUC
Feature-Based	FA	1.74	0.87 → 0.84	0.90 → 0.90	0.66 → 0.87	0.74 → 0.92
	SA	1.74	0.87 → 0.87	0.90 → 0.93	0.66 → 0.89	0.74 → 0.90
	PRED	1.66	0.87 → 0.79	0.90 → 0.87	0.66 → 0.88	0.74 → 0.94
	TCA	1.43	0.87 → 0.84	0.90 → 0.91	0.66 → 0.79	0.74 → 0.80
	fMMD	1.22	0.87 → 0.86	0.90 → 0.90	0.66 → 0.74	0.74 → 0.75
	CORAL	1.21	0.87 → 0.87	0.90 → 0.90	0.66 → 0.71	0.74 → 0.77
Instance-Based	BW	1.76	0.87 → 0.87	0.90 → 0.93	0.66 → 0.86	0.74 → 0.92
	KMM	1.59	0.87 → 0.75	0.90 → 0.84	0.66 → 0.78	0.74 → 0.85
	KLIEP	1.18	0.87 → 0.83	0.90 → 0.87	0.66 → 0.66	0.74 → 0.78
	LDM	1.01	0.87 → 0.46	0.90 → 0.71	0.66 → 0.54	0.74 → 0.63
	ULSIF	0.98	0.87 → 0.83	0.90 → 0.88	0.66 → 0.63	0.74 → 0.71
Parameter-Based	LinInt	1.71	0.87 → 0.82	0.90 → 0.90	0.66 → 0.95	0.74 → 0.99

5.1.1 Feature-based methods

The Feature Augmentation (FA) Method

a The feature-based method with the highest composite score is FA, an alternative abbreviation for FEDA. While the ADAPT implementation differs slightly from the one used in the toy model example, the underlying principle remains the same, as described in the original paper [27]. The toy model exemplified its effectiveness in binary classification tasks. Although this chapter does not primarily focus on the efficiency of the methods (defined by the fraction of labeled target data required for source-matching performance), we observe that the method was successful also on real-world datasets. Moreover, the adapted estimator performed better on the target domain than the initial hypothesis did on source data.

The Subspace Alignment (SA) Method

The SA adaptation method was introduced by Fernando et al. [30] and performed almost as well as the FA method. This method first performs a Principal Component Analysis (PCA) on the two domains in order to find the subspaces spanned by the returned eigenvectors. The number of components d returned by the PCA needs to be specified when calling SA's fit method. Table 5.1 contains the results corresponding to the value of this parameter that maximized the composite score. After determining the respective subspaces (the d basis vectors of each subspace, denoted by X_S and X_T), a subspace-alignment transformation matrix M^* is computed via the following optimization problem:

$$M^* = \arg\min_M \|X_S M - X_T\|_F^2, \tag{5.2}$$

where $\|\cdot\|_F$ is the Frobenius norm. Due to its invariance under orthonormal transformations, one can use that $X_S^t X_S = I_d$ in order to obtain the closed-form solution

$$M^* = X_S^t X_T. \tag{5.3}$$

The source data is then projected on the aligned subspace, while the target data projected on the original subspace. Adaptation then consists in training a classifier on these transformed source and target datasets.

The PRED Method

PRED is introduced as a baseline model in the same paper that proposed the FEDA algorithm [27]. The method begins by training a base estimator solely on source data. The fitted model is then evaluated on the target data, and the predictions are used as additional features to construct an augmented target feature space. The adapted model is subsequently represented by an estimator trained on this augmented target data. In the ranking based on the composite score, PRED occupies the third position, primarily due to a decrease in accuracy on the source data. However, its performance on the target data is comparable to that of the top two methods.

The Transfer Component Analysis (TCA) Method

Similarly to the SA method, TCA, introduced in [31], addresses domain adaptation by finding a common subspace where the discrepancy between source and target domain data distributions is minimized. This is achieved by learning a set of transfer components in a Reproducing Kernel Hilbert Space (RKHS) using Maximum Mean Discrepancy (MMD) as a measure of distributional distance [32]. This distance measure is desirable due to its non-parametric nature, as opposed to parametric ones, which might require a computationally expensive density estimation. More concretely, given two sets of random variables $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$ and a class of real-valued functions $\mathcal{F}$, the empirical estimate of the maximum mean discrepancy is defined as

$$\mathrm{MMD}[\mathcal{F}, X, Y] := \sup_{f \in \mathcal{F}} \left(\frac{1}{n} \sum_{i=1}^{n} f(x_i) - \frac{1}{m} \sum_{i=1}^{m} f(y_i) \right). \tag{5.4}$$

Without delving deeper into the details, it was proven [32] that selecting a reasonably rich $\mathcal{F}$, such as the unit ball in an RKHS, is enough to conclude that the MMD measure vanishes for identical distributions.

The method transforms the data from both domains into this subspace, ensuring that the transformed distributions are closer while preserving the data's geometric and statistical properties. As in the case of other feature-based methods, the adapted model will then consist in a supervised classifier trained on the embedded source data.

The Feature Selection with MMD (fMMD) Method

The fMMD method was introduced in the paper "Feature Selection for Transfer Learning" [33] and is based, just like TCA, on the MMD distance. If $\phi : \mathcal{X} \mapsto \mathcal{H}$ is a feature map and $\mathcal{H}$ a universal RKHS, the distance between the source and target domains can be empirically computed using the norm

$$\mathrm{dist}(X_S, X_T) := \left\| \frac{1}{n_S} \sum_{i=1}^{n_S} \phi(X_{S_i}) \frac{1}{n_S} \sum_{i=1}^{n_T} \phi(X_{T_i}) \right\|_{\mathcal{H}}^2. \tag{5.5}$$

Using the kernel trick, i.e., given a kernel function k, it holds that

$$\forall x_i, x_j \in \mathcal{X} : k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}}, \tag{5.6}$$

one can rewrite equation 5.5 as

$$\mathrm{dist}(X_S, X_T) = \mathrm{tr}(KL), \tag{5.7}$$

where K is the composite matrix defined on the source, target and across domains, while L is the matrix accounting for the corresponding coefficients. The objective is to identify the features that contribute the most to the distance between the two distributions. This

is done by introducing a diagonal weighting matrix W that induces a transformed kernel function k',

$$k'(x_i, x_j) = \left\langle \phi\left(W^{1/2}x_i\right), \phi\left(W^{1/2}x_j\right)\right\rangle_{\mathcal{H}}. \tag{5.8}$$

One would then look for the W that minimizes this distance in the solution of the following convex optimization problem

$$W^* = \arg\min_W -\mathrm{tr}(K'L) \quad \text{subject to} \begin{cases} \mathrm{diag}(W)^t\,\mathrm{diag}(W) \leq 1, \\ W > 0. \end{cases} \tag{5.9}$$

Finally, for a given weight threshold λ, one is able to distinguish between variant ($w_i > \lambda$) and invariant ($w_i < \lambda$) features. Once again, the adapted model consists in a classifier trained on the transformed source data, i.e., the source instances after removing the variant features.

The CORrelation ALignment (CORAL) Method

The final feature-based method that we investigated is the CORAL method, proposed in [34]. It aims to mitigate the dataset shift by aligning the second-order statistics (covariance matrices) of the source and target distributions. This is done by computing the linear transformation A that minimizes the distance

$$\|C_{\hat{S}} - C_T\|_F^2,$$

where $C_{\hat{S}}$ is the covariance matrix of the transformed source features, C_T is the covariance matrix of the target data and $\|\cdot_F^2\|$ is once again the Frobenius norm. For the technical details of the solution to this optimization problem, we refer the reader to the original paper. After the source data is transformed, a classifier is trained on the adjusted features.

This algorithm is relatively simpler than the ones described above, but its performance on the target data is the least impressive. This is most likely due to the fact that distance used almost vanishes on our datasets, resulting in very little adaptation.

5.1.2 Instance-based methods

The Balanced Weighting (BW) Method

The BW method was initially introduced [35] as a means to leverage abundant and low-quality auxiliary data in order to improve classification accuracy in scenarios where primary data is scarce. This case is particularly relevant for the domain adaptation problem, since one can treat the auxiliary data as data sampled from a source distribution, while the insufficient primary data is sampled from the target distribution. This supervised DA method is straightforward: it suggests replacing the initial (possibly regularized) risk functional, defined during the training of the base estimator, with a balanced version that takes into account the target data. Specifically, if $l(y_S, h(X_S))$ is the loss function used in the fitting of the base estimator, then the new optimization problem is given by

$$\min_h \ (1-\gamma)l(y_S, h(X_S)) + \gamma l(y_T, h(X_T)), \tag{5.10}$$

where $\gamma \in [0, 1]$ is a free parameter of the method. The degree of balancing was determined as the importance coefficient given to the target data that yielded the best performance relative to the composite score.

The Kernel Mean Matching (KMM) Method

Kernel Mean Matching (KMM) is a method designed to correct sample selection bias by reweighting the training data points such that their weighted distribution better matches the test distribution in a high-dimensional Reproducing Kernel Hilbert Space (RKHS). The optimal weights $\beta(x) \in \mathbb{R}^{n_S}$ are found by solving a quadratic programming problem that minimizes the discrepancy between the two means while maintaining normalization constraints:

$$\begin{aligned} \min_{\beta} \quad & \frac{1}{2}\beta^\top K\beta - \kappa^\top \beta, \\ \text{subject to} \quad & \left|\sum_{i=1}^{n_S} \beta_i - n_S\right| \leq n_S\epsilon, \\ & 0 \leq \beta_i \leq B, \quad \forall i = 1, \ldots, n_S. \end{aligned} \tag{5.11}$$

where K is the kernel matrix over the training data, κ is a vector reflecting the relationship between training and test data, whose components are given by

$$\kappa_i = \frac{n_S}{n_T} \sum_{x_j \in X_T} k(x_i, x_j). \tag{5.12}$$

The B and ϵ free parameters have been set by looking at the best performance in terms of the composite score. The bound on the range of possible weight values is necessary in order to avoid over-reliance on individual samples. The adapted model is the estimator trained on the weighted training data.

The Kullback–Leibler Importance Estimation Procedure (KLIEP) Method

The KLIEP method, introduced in [36], starts by assuming that the importance weights for each source sample x are given by the linear relationship

$$\beta(x) = \sum_{i=1}^{n_T} \alpha_i K(x, x_i), \tag{5.13}$$

where x_i are the target training data, x is a source instance, α_i are the linear expansion coefficients and $K(x, x_i)$ are basis functions, usually in the form of kernel models. Further assuming that $\mathcal{X}_S = \mathcal{X}_T = \mathcal{X}$, these weights are supposed to reflect the pointwise ratio between the target and source distributions, such that one might estimate the target probability from the source distribution:

$$\hat{\mathbb{P}}_T(x) = \beta(x)\mathbb{P}_S(x) \tag{5.14}$$

One can then compute the Kullback-Leibler divergence of the estimated and true target distributions:

$$\mathrm{KL}\left(\mathbb{P}_T(x)||\hat{\mathbb{P}}_T(x)\right) = \int_{\mathcal{X}} \mathbb{P}_T(x) \log \frac{\mathbb{P}_T(x)}{\beta(x)\mathbb{P}_S(x)} \mathrm{d}x = \tag{5.15}$$

$$= \int_{\mathcal{X}} \mathbb{P}_T(x) \log \frac{\mathbb{P}_T(x)}{\mathbb{P}_S(x)} \mathrm{d}x - \int_{\mathcal{X}} \mathbb{P}_T(x) \log \beta(x) \mathrm{d}x. \tag{5.16}$$

The α_i expansion coefficients are found by minimizing the KL divergence, or, equivalently, by maximizing the empirical approximation of the α_i-dependent term:

$$J := \frac{1}{n_T} \sum_{j=1}^{n_T} \log \left(\sum_{i=1}^{n_T} \alpha_i K(x_j, x_i) \right). \tag{5.17}$$

The constraint of this optimization problem comes from the following normalization condition:

$$\sum_{j=1}^{n_S} \beta(x_j) = \sum_{j=1}^{n_S} \sum_{i=1}^{n_T} \alpha_i K(x_j, x_i) = n_S. \tag{5.18}$$

In order to get the adapted model, a new classifier should be trained with the reweighted instances.

The Linear Discrepancy Minimization (LDM) Method

In a similar fashion to the CORAL method, LDM, introduced in [37], seeks to align the covariance matrices of the source and target data by a suitable transformation of the source. However, instead of looking for an embedding transformation, LDM computes the weights that contribute to the minimization of the maximum discrepancy between reweighted source data and target. The objective function is given by

$$\min_{\|w\|_1=1, w>0} \max_{\|u\|=1} \left| u^\top M(w) u \right|, \tag{5.19}$$

where

$$M(w) = \frac{1}{n_T} X_T^\top X_T - X_S^\top \mathrm{diag}(w) X_S. \tag{5.20}$$

After computing the weights, the adapted model is obtained by training the specified base estimator on the reweighted source data.

The Unconstrained Least-Squares Importance Fitting (ULSIF) Method

Following almost the same recipe as the KLIEP method, ULSIF [38] is another procedure of estimating the importance weights without explicitly estimating the density functions. Keeping the same linear expression as in equation 5.13, the difference comes from using a different objective function for finding the expansion coefficients α_i. More concretely, instead of the KL divergence, ULSIF minimizes the squared error between the estimated and the true values of the importance weights:

$$J = \int_{\mathcal{X}} \mathbb{P}_S(x) \left(\hat{\beta}(x) - \beta(x) \right)^2 \mathrm{d}x. \tag{5.21}$$

Further manipulation results in the following form of the optimization problem, after the empirical approximations have been introduced:

$$\max_{\alpha} \frac{1}{2}\alpha^\top H\alpha - h^\top\alpha + \frac{\lambda}{2}\alpha^\top\alpha, \tag{5.22}$$

where

$$H_{kl} = \frac{1}{n_S}\sum_{x_i\in X_S} K(x_i, x_k)K(x_i, x_l) \tag{5.23}$$

$$h_k = \frac{1}{n_T}\sum_{x_i\in X_T} K(x_i, x_k). \tag{5.24}$$

The regularization parameter λ controls the tradeoff between minimizing the squared loss and avoiding overfitting. To find its optimal value, ULSIF implements leave-one-out cross-validation (LOOCV). The computational efficiency comes from the existence of a closed-form solution:

$$\alpha^* = (H + \lambda I_{n_S})^{-1}h. \tag{5.25}$$

The adapted model is fitted using the reweighted source instances.

5.1.3 Parameter-based methods

The Linear Interpolation (LinInt) Method

The LinInt model has appeared as a baseline in the FEDA paper [27], along with the PRED method, and it essentially consists in predicting labels as a weighted combination of the outputs of a source-only (SrcOnly) model and a target-only (TgtOnly) model. The adapted hypothesis has the form

$$h_{\text{LinInt}} = \alpha h_{\text{src}} + (1-\alpha)h_{\text{tgt}}, \tag{5.26}$$

where the interpolation parameter α is learned with a linear regression model trained on the $(1-\gamma)$ fraction of the input target data. The method's free parameter γ represents the fraction of the input target data that ought to be used when training the TgtOnly model. We settled for the value that indicated the best performance with respect to our composite score.

5.2 An original data-augmentation method

In the continuation of this numerically focused chapter, we propose an original adaptation method [39] that is based on a different type of transformation compared to those used in the methods discussed thus far. In order to better explain the line of thought at the origin of this approach, a more complete description of the dataset is in order.

We are considering the same task as before, namely, binary classification. As mentioned earlier in this chapter, the data we have been working with consists of N blood-based spectra measured using two seemingly identical, spatially separated devices. In the language of domain adaption, this corresponds to assigning a "source" and a "target" label for each site. Each measured spectrum belongs to an individual $i \in \mathcal{P} = \{1, \ldots, P\}$ and accounts for a specific visit $j \in \mathcal{V} = \{1, \ldots, V\}$. All blood samples, collected in any given visit of an individual, are measured by the two devices. Out of the total number of individuals, we select a subset of H subjects that generates n_C spectra which are to be used as a calibration set between the two sites. Then, there will remain n_S source spectra and n_T target spectra, $n_S = n_T$, such that

$$N = n_C + n_S + n_T. \tag{5.27}$$

For a given individual i and a visit j, a data point in our dataset is represented by a random vector

$$x_{ij}^{\alpha} = \left(x_{ij}^{\alpha 1}, \ldots, x_{ij}^{\alpha W}\right) \in \mathbb{R}^W, \tag{5.28}$$

where $\alpha \in \{S, T\}$ specifies the domain and W is the number of features, commonly referred to as wavenumbers in spectroscopy. We can then define the components of the covariance matrix over a set of visits of an individual i by

$$[\Sigma_i^{\alpha}]_{mn} := \frac{1}{|\mathcal{V}^t|} \sum_{j=1}^{|\mathcal{V}^t|} \left(x_{ij}^{\alpha m} - \langle x_i^{\alpha m} \rangle\right) \left(x_{ij}^{\alpha n} - \langle x_i^{\alpha n} \rangle\right), \tag{5.29}$$

where $\mathcal{V}^t \subseteq \mathcal{V}$ refers to the subset of visits that is used during training. In the same fashion, with a slight abuse of notation we define the components of the mean spectrum of each individual by

$$\langle x_i^{\alpha m} \rangle := \frac{1}{|\mathcal{V}^t|} \sum_{j=1}^{|\mathcal{V}^t|} x_{ij}^{\alpha m} \tag{5.30}$$

The idea of our method is the following. Using the elementwise-averaged covariance matrix over the calibration set,

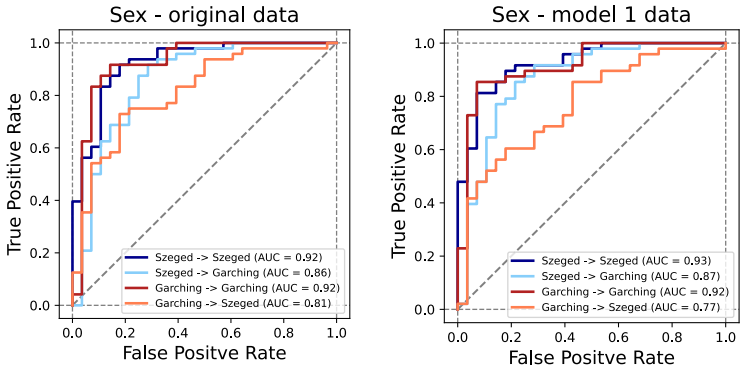
$$[\overline{\Sigma}^C]_{mn} := \frac{1}{H} \sum_{i=1}^{H} [\Sigma_i]_{mn}^C, \tag{5.31}$$

together with the mean spectrum of each individual as defined in equation 5.30, one can associate to each individual a multivariate Gaussian distribution (MVG) with the first two moments specified by these quantities:

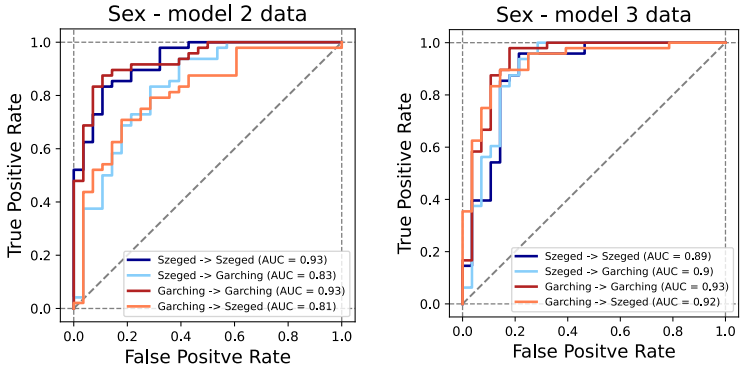
$$\mathcal{G}_W\left(x_{ij}^{\alpha} \mid \langle x_i^{\alpha} \rangle, \overline{\Sigma}^C\right) = \frac{\exp\left(-\frac{1}{2}(x_{ij}^{\alpha} - \langle x_i^{\alpha} \rangle)^t \overline{\Sigma}^{-1} (x_{ij}^{\alpha} - \langle x_i^{\alpha} \rangle)\right)}{(2\pi)^{W/2} |\overline{\Sigma}|^{1/2}}. \tag{5.32}$$

The associated MVGs can then be used to simulate visits for each individual in the training set. The adapted model is then given by a classifier trained on the augmented dataset.

The results of our method are shown in Figure 5.1. The first subfigure illustrates the performance of the base estimator, i.e., the one trained on the original data. The following two subfigures correspond to slightly different attempts at adaptation, using different forms of the covariance matrix that defines the MVG. The performance of the method described so far in this section is shown in the last subfigure (Model 3). We observe that initially the base estimator had a ROC-AUC score of 92% on test data coming from the same distribution, with slightly more modest values in cross-site prediction: 86% and 81%. After adaptation, we notice an increase in the target performances that is comparable to the adapted method's within-site scores, demonstrating the success of our approach.



(a) Base estimator (b) Model 1



(c) Model 2 (d) Model 3

Figure 5.1: Performances of several models. The base estimator is shown in subfigure a), while the method described in this section is illustrated in d). Figure reproduced from the original paper [39].

Chapter 6

Conclusions

This thesis set out to address a significant challenge in the field of machine learning: the domain adaptation problem, particularly as it applies to sensitive applications like disease detection. Our research was driven by the practical observation that machine learning models often suffer from a substantial performance drop when applied to data that, although related, comes from a different distribution than the training data. This issue is particularly critical in medical diagnostics, where models need to generalize reliably across different patient populations, measurement times, or even devices.

To tackle this problem, we first explored a specific domain adaptation method known as Frustratingly Easy Domain Adaptation (FEDA). The method's simplicity contrasts with its effectiveness, making it an attractive option for practitioners who require robust solutions without the complexity of more intricate models. The FEDA method works by augmenting the feature space in a way that helps the model bridge the gap between the source and target domains with minimal additional data. The next step was to employ a simple toy model designed to simulate a domain shift. This model involved generating a dataset from two bivariate Gaussian distributions and applying an affine transformation to simulate a shift in the data distribution. This setup provided a controlled environment to test the FEDA method and compare it with other learning scenarios, such as full retraining on the target domain data.

As we have seen, the FEDA method can achieve near-optimal performance using only a small fraction of the target domain data. Specifically, the method has demonstrated the ability to reach performance levels comparable to full retraining with as little as 5% of the target data. This outcome is particularly significant in scenarios where acquiring labeled target data is costly or logistically challenging. We also looked into the theoretical foundations that explain why FEDA is so effective. By analyzing the VC dimensions of the involved hypothesis classes, it was demonstrated that the method is guaranteed to work with very little target data if the complexity ratio between the two classes is sufficiently small. However, this was not the case for the simple model used in the initial training, i.e., logistic regression combined with general affine shifts, as the ratio in this instance turned out to be 1. Consequently, we developed a heuristic explanation for the observed

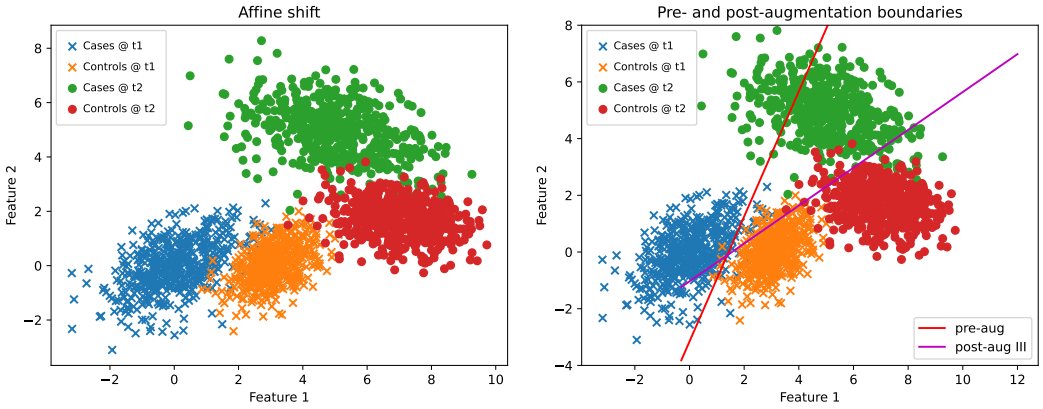
efficiency, showing that the rapid reduction in the probability of encountering misclassified target points after each update plays a crucial role in the method's success.

The previous chapter focused on testing various domain adaptation (DA) methods on a real-world dataset, also for the task of binary classification. To this end, we explored several models implemented in the ADAPT Python toolbox and analyzed their performance in terms of accuracy, ROC-AUC, and a custom composite score. Additionally, we provided a brief overview of each method, emphasizing the similarities and structural differences that justified their classification into three families: feature-based, instance-based, and parameter-based. Next, we proposed an alternative adaptation method that, unlike those described in the first section, involves a different type of transformation. Instead of identifying a feature-space transformation, this approach augments the training dataset by fitting multivariate Gaussian distributions using the individual mean spectra and the elementwise-averaged covariance matrix over a calibration set, followed by sampling visits for each subject. Training on the enhanced dataset yielded a model with an improved target performance, almost equal to the within-site results, thus demonstrating the success of our method.

One natural extension of this work would be to test these domain adaptation methods on tasks beyond binary classification. The ADAPT package already includes methods specifically tailored for linear regression, enabling their application to a broader range of predictive tasks involving continuous outputs. Furthermore, it also offers methods custom-designed for neural networks, which could be explored for more complex, non-linear problems. Widening the range to include multi-class classification or unsupervised learning tasks, such as clustering or dimensionality reduction, would provide insights into the generalizability and adaptability of the approaches in various domains.

Appendix A

Details on the toy model



(a) Affine transformation in the toy model.

(b) The decision boundaries before and after adaptation.

Figure 1: Subfigure a) Toy model dataset. Subfigure b) Pre- and post- adaptation decision boundaries.

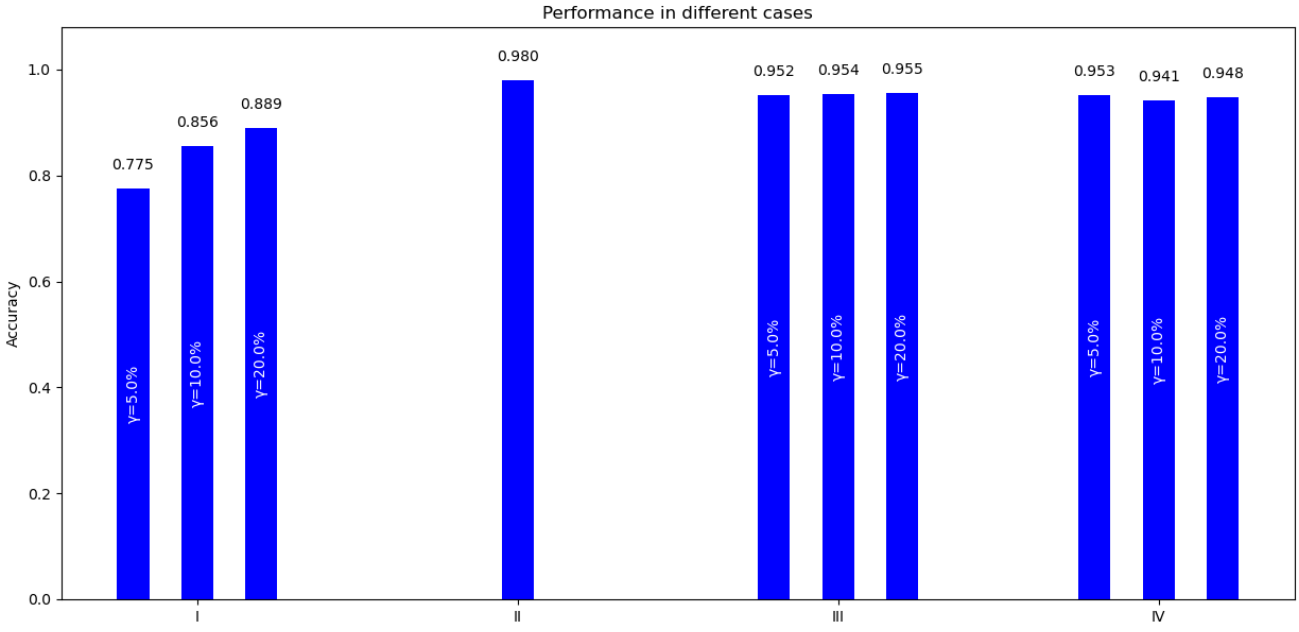
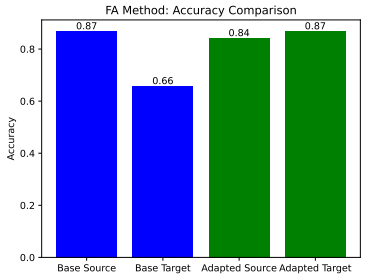


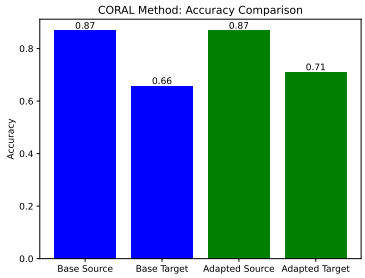
Figure 2: The four learning scenarios and the accuracies of the models for varying fractions γ.

Appendix B

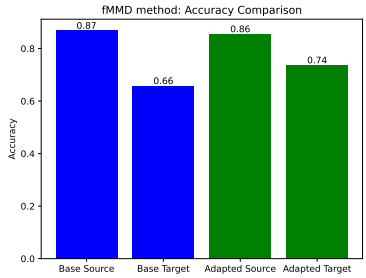
ADAPT package results



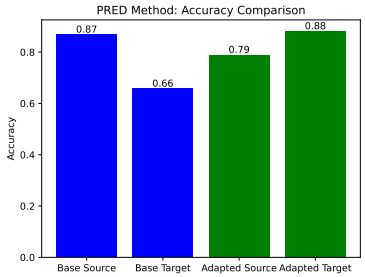
(a) FA Method



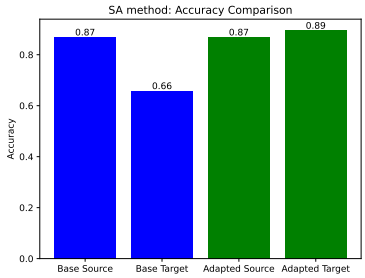
(b) CORAL Method



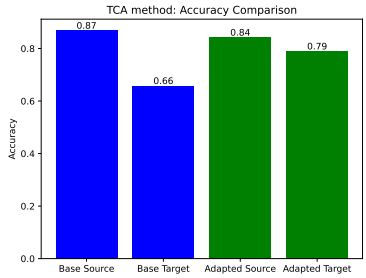
(c) fMMD Method



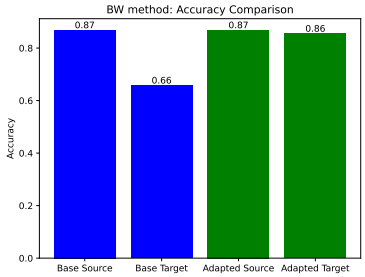
(d) PRED Method



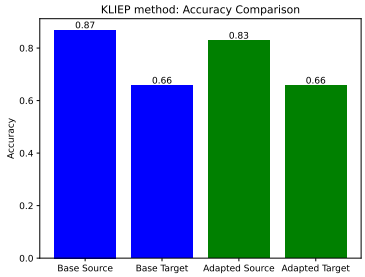
(e) SA Method



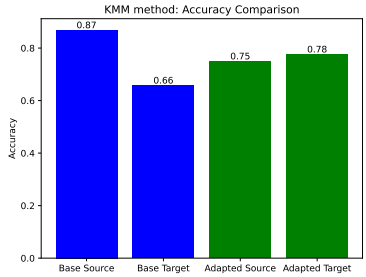
(f) TCA Method



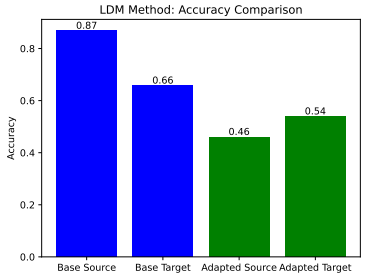
(g) BW Method



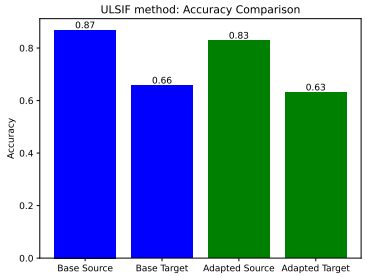
(h) KLIEP Method



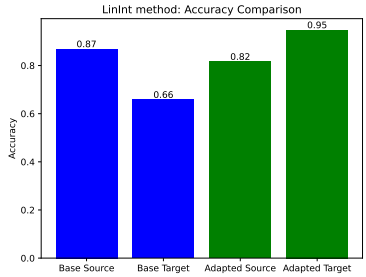
(i) KMM Method



(j) LDM Method

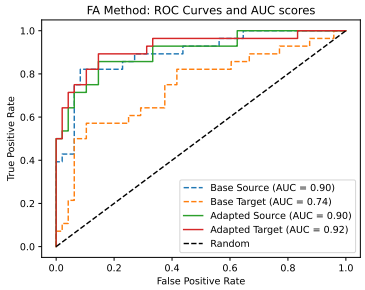


(k) ULSIF Method

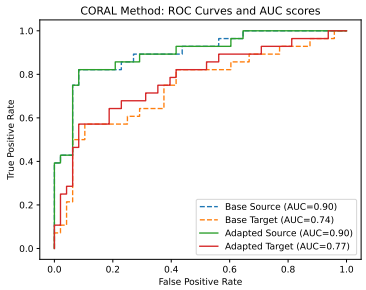


(l) LinInt Method

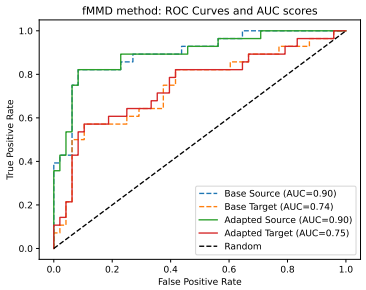
Figure 3: Comparison of accuracies.



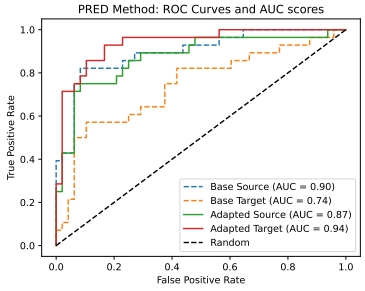
(a) FA Method



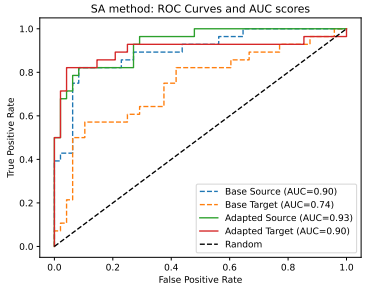
(b) CORAL Method



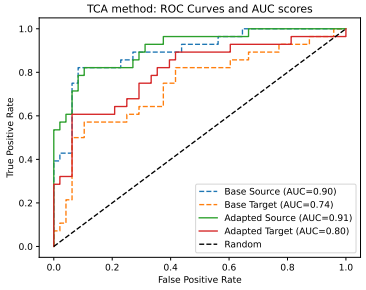
(c) fMMD Method



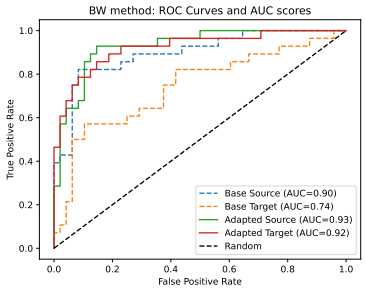
(d) PRED Method



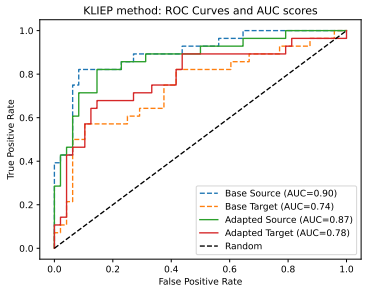
(e) SA Method



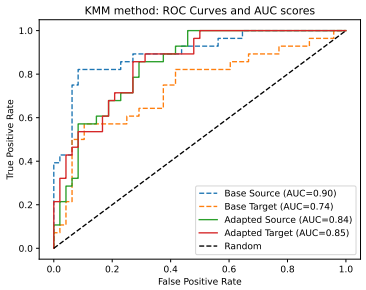
(f) TCA Method



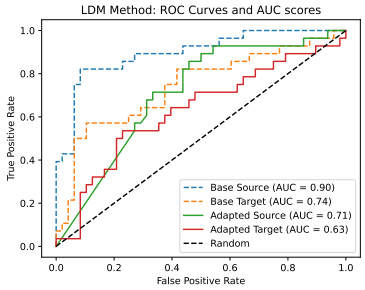
(g) BW Method



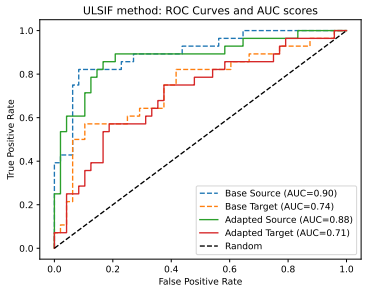
(h) KLIEP Method



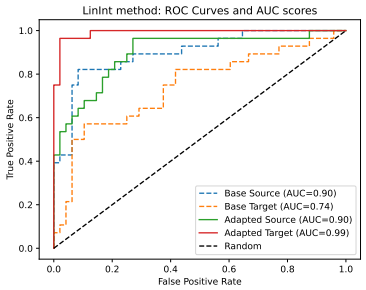
(i) KMM Method



(j) LDM Method



(k) ULSIF Method



(l) LinInt Method

Figure 4: Comparison of ROC-AUC scores.

Bibliography

[1] Antoine de Mathelin, François Deheeger, Guillaume Richard, Mathilde Mougeot, and Nicolas Vayatis. Adapt: Awesome domain adaptation python toolbox. *arXiv preprint arXiv:2107.03049*, 2021.

[2] Marzyeh Ghassemi, Tristan Naumann, Peter Schulam, Andrew L. Beam, and Irene Y. Chen. A review of challenges and opportunities in machine learning for health. *AMIA Summits on Translational Science Proceedings*, 2020:191–200, 2020.

[3] Eric J Topol. High-performance medicine: the convergence of human and artificial intelligence. *Nature Medicine*, 25(1):44–56, 2019.

[4] Alvin Rajkomar, Jeffrey Dean, and Isaac Kohane. Machine learning in medicine. *New England Journal of Medicine*, 380(14):1347–1358, 2019.

[5] Kosmas V. Kepesidis, Mircea-Gabriel Stoleriu, Nico Feiler, et al. Assessing lung cancer progression and survival with infrared spectroscopy of blood serum. *Research Square*, 2024. Preprint (Version 1).

[6] Tarek Eissa, Kosmas V. Kepesidis, Mihaela Zigman, and Marinus Huber. Assessing lung cancer progression and survival with infrared spectroscopy of blood serum. *Analytical Chemistry*, 95(16):6523–6532, 2023.

[7] Kosmas V. Kepesidis, Masa Bozic-Iven, Marinus Huber, and et al. Assessing lung cancer progression and survival with infrared spectroscopy of blood serum. *BMC Cancer*, 21:1–9, 2021.

[8] Marinus Huber, Kosmas V. Kepesidis, Liudmila Voronina, and et al. Infrared molecular fingerprinting of blood-based liquid biopsies for the detection of cancer. *eLife*, 10:e68758, 2021.

[9] Alexander S. Lundervold and Arvid Lundervold. An overview of deep learning in medical imaging focusing on mri. *Zeitschrift für Medizinische Physik*, 29(2):102–127, 2019.

[10] Geert Litjens, Thijs Kooi, Babak E. Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017.

[11] Anant Madabhushi and George Lee. Image analysis and machine learning in digital pathology: Challenges and opportunities. *Medical Image Analysis*, 33:170–175, 2016.

[12] Maxwell W. Libbrecht and William Stafford Noble. Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, 16:321–332, 2015.

[13] Riccardo Miotto, Fei Li, Brian A. Kidd, and Joel T. Dudley. Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific Reports*, 6:26094, 2016.

[14] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, second edition, 2018. ISBN: 978-0-262-03940-6.

[15] Richard M. Golden. *Statistical Machine Learning*. CRC Press, 2020. ISBN: 978-1-351-05150-7.

[16] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.

[17] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. ISBN: 978-1-107-05713-5.

[18] Dirk-André Deckert. Mathematical statistics and applications of machine learning, 2021. Available at: https://dirk-deckert-lmu.gitlab.io/ss21-msaml/.

[19] René L. Schilling. *Measures, Integrals and Martingales*. Cambridge University Press, 2005.

[20] Shai Ben David, Tyler Lu, Teresa Luu, and David Pal. Impossibility theorems for domain adaptation. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 129–136, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. Available at: https://proceedings.mlr.press/v9/david10a.html/.

[21] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, nov 1984.

[22] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, New York, 1998.

[23] Michael M. Wolf. Mathematical foundations of supervised learning, October 2023. (growing lecture notes), available at https://mediatum.ub.tum.de/doc/1723378/1723378.pdf.

[24] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from Data: A Short Course*. AMLbook, 2012. ISBN: 978-1-60049-006-4.

[25] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[26] Ievgen Redko, Amaury Habrard, Emilie Morvant, Marc Sebban, and Younès Bennani. *Advances in Domain Adaptation Theory*. ISTE Press Ltd and Elsevier Ltd, 2019. ISBN: 978-1-78548-236-6.

[27] Hal Daumé III. Frustratingly easy domain adaptation. In Annie Zaenen and Antal van den Bosch, editors, *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[28] Ajitesh Kumar. Logistic regression in machine learning, 2024. Available at: `https://vitalflux.com/python-train-model-logistic-regression/`.

[29] Hal Daumé III, Abhishek Kumar, and Avishek Saha. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 53–59, 2010.

[30] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Subspace alignment for domain adaptation, 2014.

[31] Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

[32] Karsten M. Borgwardt, Arthur Gretton, Malte J. Rasch, Hans-Peter Kriegel, Bernhard Scholkopf, and Alex Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22 14:e49–57, 2006.

[33] Selen Uguroglu and Jaime G. Carbonell. Feature selection for transfer learning. In *ECML/PKDD*, 2011.

[34] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation, 2015.

[35] Pengcheng Wu and Thomas G. Dietterich. Improving svm accuracy by training on auxiliary data sources. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 110, New York, NY, USA, 2004. Association for Computing Machinery.

[36] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Proceedings of the 20th International Conference*

on Neural Information Processing Systems, NIPS'07, page 1433–1440, Red Hook, NY, USA, 2007. Curran Associates Inc.

[37] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms, 2023.

[38] Takafumi Kanamori, Shohei Hido, and Masashi Sugiyama. A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, 10(48):1391–1445, 2009.

[39] F. B. Nemeth, N. Leopold-Kerschbaumer, D. Debreceni, F. Fleischmann, K. Borbely, D. Mazurencu-Marinescu-Pele, M. Zigman, and K. V. Kepesidis. Bridging spectral gaps: Cross-device model generalization in blood-based infrared spectroscopy. 2024. Submitted manuscript.

[40] Gabriela Csurka, Timothy M. Hospedales, Mathieu Salzmann, and Tatiana Tommasi. *Visual Domain Adaptation in the Deep Learning Era*. Springer Nature Switzerland, 2022. ISBN: 978-3-031-79170-3.

[41] Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*, volume 31 of *Stochastic Modelling and Applied Probability*. Springer, 1996.

[42] Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.

[43] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael I. Jordan. Bridging theory and algorithm for domain adaptation. In *International Conference on Machine Learning*, 2019.

[44] Y. Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *ArXiv*, abs/0902.3430, 2009.

Affidavit

Hereby, I declare that I have written the present work independently and have used no sources or tools other than those specified in the work.

Munich, [12.12.2024]

David Mihai Mazurencu-Marinescu-Pele