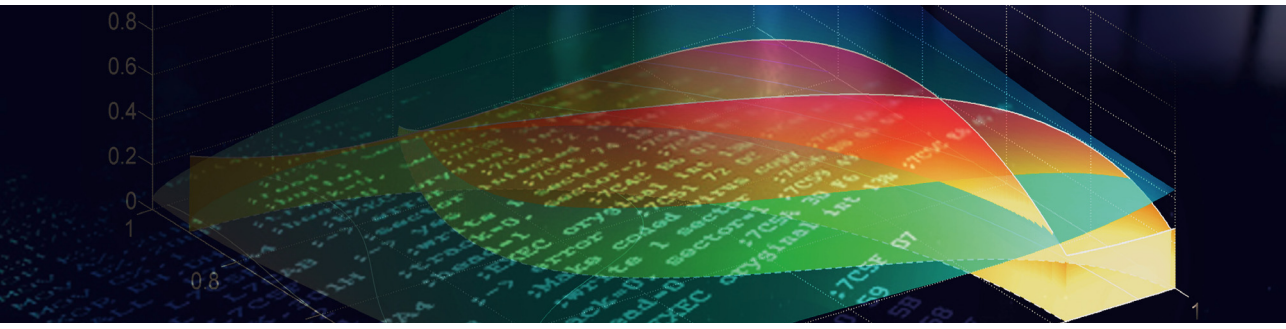

Michael Brückner

# Prediction Games

## Machine Learning in the Presence of an Adversary

Universitätsverlag Potsdam

Michael Brückner

# Prediction Games
## Machine Learning in the Presence of an Adversary

Michael Brückner

# Prediction Games

Machine Learning in the Presence of an Adversary

Universitätsverlag Potsdam

**Bibliografische Information der Deutschen Nationalbibliothek**
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.de/ abrufbar.

**Universitätsverlag Potsdam 2012**
http://verlag.ub.uni-potsdam.de/

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 2533 / Fax: 2292
E-Mail: verlag@uni-potsdam.de

Zugl.: Potsdam, Univ., Diss., 2012
Gutachter:
Prof. Dr. Karsten Borgwardt, Max-Planck-Institut für Intelligente Systeme
Prof. Dr. Andreas Fischer, Institut für Numerische Mathematik, TU Dresden
Prof. Dr. Tobias Scheffer, Institut für Informatik, Universität Potsdam
Datum der Disputation: 5. Juli 2012

This work is licensed under a Creative Commons License:
Attribution - Noncommercial - Share Alike 3.0 Unported
To view a copy of this license visit
http://creativecommons.org/licenses/by-nc-sa/3.0/
Umschlagmotiv: Montage aus einer Grafik des Autors und folgendem Foto:
http://www.flickr.com/photos/hj_barraza/415134620/sizes/z/in/photostream/
Urheber: .hj barraza, CC-License: Attribution - Share Alike 2.0

Online veröffentlicht auf dem Publikationsserver der Universität Potsdam:
URL http://pub.ub.uni-potsdam.de/volltexte/2012/6037/
URN urn:nbn:de:kobv:517-opus-60375
http://nbn-resolving.org/urn:nbn:de:kobv:517-opus-60375

Zugleich gedruckt erschienen im Universitätsverlag Potsdam:
ISBN 978-3-86956-203-2

# Zusammenfassung

Eine der Aufgabenstellungen des Maschinellen Lernens ist die Konstruktion von *Vorhersagemodellen* basierend auf gegebenen *Trainingsdaten*. Ein solches Modell beschreibt den Zusammenhang zwischen einem *Eingabedatum*, wie beispielsweise einer E-Mail, und einer *Zielgröße* – zum Beispiel, ob die E-Mail durch den Empfänger als erwünscht oder unerwünscht empfunden wird. Dabei ist entscheidend, dass ein gelerntes Vorhersagemodell auch die Zielgrößen zuvor unbeobachteter *Testdaten* korrekt vorhersagt.

Die Mehrzahl existierender Lernverfahren wurde unter der Annahme entwickelt, dass Trainings- und Testdaten derselben Wahrscheinlichkeitsverteilung unterliegen. Insbesondere in Fällen in welchen zukünftige Daten von der Wahl des Vorhersagemodells abhängen, ist diese Annahme jedoch verletzt. Ein Beispiel hierfür ist das automatische Filtern von Spam-E-Mails durch E-Mail-Anbieter. Diese konstruieren Spam-Filter basierend auf zuvor empfangenen E-Mails. Die Spam-Sender verändern daraufhin den Inhalt und die Gestaltung der zukünftigen Spam-E-Mails mit dem Ziel, dass diese durch die Filter möglichst nicht erkannt werden.

Bisherige Arbeiten zu diesem Thema beschränken sich auf das Lernen *robuster* Vorhersagemodelle welche unempfindlich gegenüber geringen Veränderungen des datengenerierenden Prozesses sind. Die Modelle werden dabei unter der *Worst-Case-Annahme* konstruiert, dass diese Veränderungen einen maximal negativen Effekt auf die Vorhersagequalität des Modells haben. Diese Modellierung beschreibt die tatsächliche Wechselwirkung zwischen der Modellbildung und der Generierung zukünftiger Daten nur ungenügend. Aus diesem Grund führen wir in dieser Arbeit das Konzept der *Prädiktionsspiele* ein. Die Modellbildung wird dabei als mathematisches Spiel zwischen einer lernenden und einer datengenerierenden Instanz beschrieben. Die spieltheoretische Modellierung ermöglicht es uns, die Interaktion der beiden Parteien exakt zu beschreiben. Dies umfasst die jeweils verfolgten Ziele, ihre Handlungsmöglichkeiten, ihr Wissen über einander und die zeitliche Reihenfolge, in der sie agieren.

Insbesondere die Reihenfolge der Spielzüge hat einen entscheidenden Einfluss auf die spieltheoretisch optimale Lösung. Wir betrachten zunächst den Fall gleichzeitig agierender Spieler, in welchem sowohl der Lerner als auch der Datengenerierer keine Kenntnis über die Aktion des jeweils anderen Spielers haben. Wir leiten hinreichende Bedingungen her, unter welchen dieses Spiel eine Lösung in Form eines eindeutigen *Nash*-Gleichgewichts besitzt. Im Anschluss diskutieren wir zwei verschiedene Verfahren zur effizienten Berechnung dieses Gleichgewichts. Als zweites betrachten wir den Fall eines *Stackelberg*-Duopols. In diesem Prädiktionsspiel wählt der Lerner zunächst das Vorhersagemodell, woraufhin der Datengenerierer in voller Kenntnis des Modells reagiert. Wir leiten ein relaxiertes

Optimierungsproblem zur Bestimmung des Stackelberg-Gleichgewichts her und stellen ein mögliches Lösungsverfahren vor. Darüber hinaus diskutieren wir, inwieweit das Stackelberg-Modell bestehende robuste Lernverfahren verallgemeinert. Abschließend untersuchen wir einen Lerner, der auf die Aktion des Datengenerierers, d.h. der Wahl der Testdaten, reagiert. In diesem Fall sind die Testdaten dem Lerner zum Zeitpunkt der Modellbildung bekannt und können in den Lernprozess einfließen. Allerdings unterliegen die Trainings- und Testdaten nicht notwendigerweise der gleichen Verteilung. Wir leiten daher ein neues integriertes sowie ein zweistufiges Lernverfahren her, welche diese *Verteilungsverschiebung* bei der Modellbildung berücksichtigen.

In mehreren Fallstudien zur Klassifikation von Spam-E-Mails untersuchen wir alle hergeleiteten, sowie existierende Verfahren empirisch. Wir zeigen, dass die hergeleiteten spieltheoretisch-motivierten Lernverfahren in Summe signifikant bessere Spam-Filter erzeugen als alle betrachteten Referenzverfahren.

# Abstract

In many applications one is faced with the problem of inferring some functional relation between input and output variables from given data. Consider, for instance, the task of email spam filtering where one seeks to find a model which automatically assigns new, previously unseen emails to class spam or non-spam. Building such a predictive model based on observed training inputs (*e.g.,* emails) with corresponding outputs (*e.g.,* spam labels) is a major goal of machine learning.

Many learning methods assume that these *training data* are governed by the same distribution as the *test data* which the predictive model will be exposed to at application time. That assumption is violated when the test data are generated in response to the presence of a predictive model. This becomes apparent, for instance, in the above example of email spam filtering. Here, email service providers employ spam filters and spam senders engineer campaign templates such as to achieve a high rate of successful deliveries despite any filters.

Most of the existing work casts such situations as learning *robust* models which are unsusceptible against small changes of the data generation process. The models are constructed under the worst-case assumption that these changes are performed such to produce the highest possible adverse effect on the performance of the predictive model. However, this approach is not capable to realistically model the true dependency between the model-building process and the process of generating future data. We therefore establish the concept of *prediction games*: We model the interaction between a *learner*, who builds the predictive model, and a *data generator*, who controls the process of data generation, as an one-shot game. The game-theoretic framework enables us to explicitly model the players' interests, their possible actions, their level of knowledge about each other, and the order at which they decide for an action.

We model the players' interests as minimizing their *own* cost function which both depend on both players' actions. The learner's action is to choose the model parameters and the data generator's action is to perturbate the training data which reflects the modification of the data generation process with respect to the past data.

We extensively study three instances of prediction games which differ regarding the order in which the players decide for their action. We first assume that both player choose their actions simultaneously, that is, without the knowledge of their opponent's decision. We identify conditions under which this *Nash prediction game* has a meaningful solution, that is, a unique Nash equilibrium, and derive algorithms that find the equilibrial prediction model. As a second case, we consider a data generator who is potentially fully informed about the move of the learner. This setting establishes a Stackelberg competition. We derive a relaxed optimization criterion to determine the solution of this game and show that this

*Stackelberg prediction game* generalizes existing prediction models. Finally, we study the setting where the learner observes the data generator's action, that is, the (unlabeled) test data, before building the predictive model. As the test data and the training data may be governed by differing probability distributions, this scenario reduces to learning under *covariate shift*. We derive a new integrated as well as a two-stage method to account for this data set shift.

In case studies on email spam filtering we empirically explore properties of all derived models as well as several existing baseline methods. We show that spam filters resulting from the Nash prediction game as well as the Stackelberg prediction game in the majority of cases outperform other existing baseline methods.

# Acknowledgements

I am pleased to take the opportunity to thank many people who supported me in my studies and in writing this thesis.

First of all, I would like to thank my advisor Tobias Scheffer for giving me the opportunity to pursue my PhD in his research group. This thesis would not have been possible without his invaluable support and guidance. I am grateful to STRATO AG, in particular to the chief information officer René Wienholtz, for financing my studies through a joint research project on email spam filtering with the University of Potsdam. This cooperation gave me the chance not only to work on an exciting research topic, but also to translate my findings into practice.

I would like to thank Christian Kanzow for pointing me to a shortcoming in one of my publications and for assisting me in developing a revised version. I really enjoyed our fruitful email discussions. I wish to thank Steffen Bickel, Christoph Sawade, and Niels Landwehr for the many productive talks, suggestions and ideas, and especially for consistently reviewing my writings and for proofreading this thesis. It was a great pleasure working together with Ulf Brefeld, Uwe Dick, Laura Dietz, Isabel Drost, Peter Haider, Paul Prasse, Arvid Terzibaschian, and Thomas Vanck. They all contributed to a very friendly and stimulating research atmosphere. Especially our annual winter workshops were always great fun. Furthermore, I would like to thank the STRATO developer team including Arne Jansen, Michael Kockelkorn, Jan Schmidt, and Barnim Dzwillo for being patient with me when I was implementing my algorithms.

Finally, I would like to thank my parents who were always supporting and encouraging me.

# Contents


**1 Introduction** **1**
- 1.1 Motivating Examples . . . 2
- 1.2 An Arms Race Between Learner and Data Generator . . . 4
- 1.3 Contributions . . . 5
- 1.4 Own Previously Published Work . . . 8
- 1.5 Outline . . . 10

**2 Learning Predictive Models** **11**
- 2.1 Risk Minimization . . . 12
- 2.2 Learning in the Bayesian Framework . . . 14
- 2.3 Linear Decision Functions . . . 17
- 2.4 Loss Functions . . . 19
- 2.5 Regularization . . . 22
- 2.6 Feature Representation and Kernels . . . 23
- 2.7 Parameter Estimation . . . 26

**3 The Prediction Game** **29**
- 3.1 An Introduction to Game Theory . . . 29
  - 3.1.1 Basic Terms and Definitions . . . 30
  - 3.1.2 Solution Concepts . . . 32
- 3.2 Adversarial Prediction Problems . . . 35
  - 3.2.1 Number of Players . . . 35
  - 3.2.2 Number of Repetitions . . . 36
- 3.3 Modeling the Prediction Game . . . 37
- 3.4 Classes of Prediction Games . . . 39

**4 Nash Prediction Games** **41**
- 4.1 Nash Solution to Prediction Games . . . 41
  - 4.1.1 Existence of a Nash Equilibrium . . . 42
  - 4.1.2 Uniqueness of the Nash Equilibrium . . . 43
- 4.2 Finding the Unique Nash Equilibrium . . . 48
  - 4.2.1 An Inexact Linesearch Approach . . . 48
  - 4.2.2 A Modified Extragradient Approach . . . 50
- 4.3 Applying Kernels . . . 50
- 4.4 Instances of the Nash Prediction Game . . . 52
  - 4.4.1 Nash Logistic Regression . . . 53

4.4.2 Nash Support Vector Machine . . . 54
4.5 Related Work . . . 55
4.6 Empirical Evaluation . . . 56
4.6.1 Convergence . . . 57
4.6.2 Regularization Parameters . . . 58
4.6.3 Evaluation for Nash-Playing Adversary . . . 60
4.6.4 A Case Study on Email Spam Filtering . . . 61
4.6.5 Efficiency versus Effectiveness . . . 63
4.6.6 Nash-Equilibrial Transformation . . . 63

**5 Stackelberg Prediction Games** **67**
5.1 Stackelberg Solution to Prediction Games . . . 67
5.2 An SQP Method for Stackelberg Prediction Games . . . 71
5.3 Applying Kernels . . . 72
5.4 Instances of the Stackelberg Prediction Game . . . 73
5.4.1 Worst-Case Loss . . . 73
5.4.2 Linear Loss . . . 74
5.4.3 Logistic Loss . . . 75
5.5 Related Work . . . 76
5.6 Empirical Evaluation . . . 77
5.6.1 A Case Study on Email Spam Filtering . . . 77
5.6.2 Efficiency versus Effectiveness . . . 79
5.6.3 Transformation . . . 79

**6 Covariate Shift** **81**
6.1 Learning under Covariate Shift . . . 82
6.1.1 MAP Estimation under Covariate Shift . . . 83
6.1.2 An Integrated Model . . . 86
6.2 Logistic Regression Importance Estimation . . . 88
6.3 A Two-Stage Approximation . . . 91
6.4 Applying Kernels . . . 92
6.5 Related Work . . . 93
6.6 Empirical Evaluation . . . 94
6.6.1 A Case Study on Email Spam Filtering . . . 94
6.6.2 Inspection of the Resampling Weights . . . 96

**7 Conclusions** **99**

**Bibliography** **105**

**Appendix** **111**

**Notation** **119**

# 1 Introduction

In this thesis we consider the problem of identifying the relation between an *object* and the *target* attribute from some given set of object-target pairs called *training data*. This relationship, *i.e.,* a mapping from objects to targets, is referred to as a *prediction model*. It can be used to estimate the value of the target attribute for previously unseen objects. Identifying such a prediction model from a finite amount of data is generally a non-trivial problem as the model is required to generalize to new objects. To this end, the given training data are typically assumed to be a representative sample of data points which one can expect at application time, that is, the *test data*.

For several applications, however, this assumption is violated and therefore many common theoretical guarantees and bounds on the models' predictive performance are no longer valid. We address a special case where this assumption is violated, called the *adversarial prediction problem*. In this setting, the future test data depend on the chosen prediction model and, consequently, on the training data. Classical learning methods, which disregard this dependency, are not expected to identify satisfying prediction models.

The adversarial prediction problem establishes a game, where one party—called the *learner*—infers a predictive model from the training data. A second party—called the *data generator*—controls the data generation process that is used to produce future test data. Both parties interact with each other. The data generator responds to the learner's chosen prediction model by changing the data generation process. In contrast, the learner adapts the prediction model to explain the relation between objects and targets generated by this new data generation process.

Such situations frequently occur in practice. Consider, for instance, the following scenarios: In computer and network security, scripts that control attacks are engineered with botnet and intrusion detection systems in mind. Credit card fraudsters adapt their unauthorized use of credit cards—in particular, amounts charged per transactions and per day and the type of businesses that amounts are charged from—such as not to trigger alerting mechanisms, employed by credit card companies. Email spam senders design message templates that are instantiated by nodes of botnets, where those templates are specifically designed to produce a low spam score with filtering strategies, that are expected to be employed by the email service provider.

In all of these applications, the learner and the data generator are aware of each other, and factor the possible actions of their opponent into their decisions. To find the optimal action of the learner, that is, the prediction model which is most robust with respect to the expected action of the data generator, we model this interaction as a mathematical two-player game.

To the best of our knowledge, we are the first who systematically study the adversarial prediction problem using concepts from game theory. We formulate this learning problem as a game, which we call a *prediction game*. We continue prior research on robust learning algorithms, which are mainly based on the worst-case assumption that the data generator desires to impose the highest possible costs on the learner. The worst-case assumption amounts to a zero-sum game in which the cost of one player is the gain of the other. However, several applications motivate problem settings in which the goals of the learner and the data generator, while still conflicting, are not necessarily entirely antagonistic. For instance, a fraudster's goal of maximizing the profit made from exploiting phished account information is not the inverse of an email service provider's goal of achieving a high spam recognition rate at close-to-zero false positives. When modeling such settings as a zero-sum game, one often makes overly pessimistic assumptions about the data generator's behavior and may not necessarily obtain an optimal outcome. To this end, we abstain from the worst-case assumption and derive prediction games where either player is expected to follow their own interests which may or may not be antagonistic.

To begin with, we present some typical applications in Section 1.1 which motivate the study of adversarial prediction problems. In Section 1.2, we briefly discuss common properties of adversarial prediction problems and informally introduce the concept of prediction games to model such problems. We summarize the main contributions of this thesis in Section 1.3 and list own, previously published work in Section 1.4. Finally, Section 1.5 provides an overview on the remaining chapters of this thesis.

## 1.1 Motivating Examples

In a variety of applications, data at application time are partially generated by an adversary whose interests are in conflict with those of the learner. Consider, for instance, the following example scenarios.

#### Drug and Therapy Selection

Estimating the prospect of success of a therapy, for example, a combination of antiviral agents, is important to design new drugs and medications. The goal of a pharmacologist, *i.e.,* the learner, is to derive a prediction model which estimates the success of a specific therapy. However, viruses and bacteria, *i.e.,* the adversaries, frequently mutate and become resistant against drugs. The likelihood of becoming resistant depends on the specific DNA of the pathogenic agent, the patients' medication histories, and the chosen pharmaceutical, which depends on the outcome of the predictive model. To predict the success of a potential therapy the evolutional response of the pathogenic agent has to be considered when learning the model.

**Defending against Denial-of-Service Attacks**

Users of online services, such as web hosting, online file storage, community portals, or bulletin boards, generate data when requesting the service. An attacker aims at blocking the service as long as possible by sending large amounts of potentially computationally expensive requests, for instance, repeated requests of a specific CGI script, which may slow down the server. The task of the service provider is to distinguish between legitimate and those abusive requests, to defend against denial-of-service (DoS) attacks. The main challenge for the learner is to identify abnormal usage patterns while the adversary constantly changes the attack vector.

**Intrusion Detection**

In computer networks, firewalls, intrusion detection systems, and system integrity checkers are used to prevent unwanted access by intruders. Similar as for denial-of-service attacks, the prevention system must differentiate between legitimate and illegitimate access attempts. Thereby, a typical intrusion scenario might contain the following steps: First the attacker gathers information on the target, for example, crawling the website for email addresses, performing a whois lookup, port scans, ping sweeps, and checking versions of running applications and services. Thereafter, the intruder may try out different attacks based on the expected vulnerabilities and misconfigurations. Finally, the intruder gets access to the system, installs their own backdoor, and covers up their track. The goal in intrusion detection is to identify such access patterns in order to protect the system from being compromised. However, the intruders are expected to vary their strategies to remain undetected.

**Preventing Credit Card Fraud**

Customers of banks paying with credit card generate large amounts of transaction data. To prevent abuse, *e.g.,* by stolen cards, the financial institutes employ fraud detection systems. These systems try to identify unusual transaction patterns, such as repeatedly withdrawing money from locations being far away from each other. In order to avoid detection, criminals will try to carry out transactions in such a way, that existing detection systems are not triggered; for instance, by modifying their strategy of drawing money.

**Email Spam Filtering**

Email service providers (ESP) are faced with the problem of filtering unsolicited messages. This is important not only to prevent their customers from spam, but also to ensure the availability of the service. Each email, which is accepted for delivery by the ESP, yields processing and storage costs. Since these resources are limited, receiving too many mainly unwanted emails acts as a denial-of-service attack. Moreover, outgoing spam emails may cause the sending server being blacklisted by other

ESPs, which partially terminates the service, too. Hence, for inbound and, especially, outbound emails, content-based filtering techniques are employed by ESPs. Unfortunately, spam senders respond to these filters by constantly modifying the spam emails and the spam-generating software, respectively.

Throughout the thesis we consider the last application of email spam filtering as a running example. We are exclusively interested in identifying an optimal action for the learner, *i.e.,* an adversary-aware prediction model, and do not explicitly solve for an optimal action of the data generator, that is, a data generation model that minimizes the data generator's costs.

## 1.2 An Arms Race Between Learner and Data Generator

In this section we briefly introduce the concept of *prediction games* to model the conflicting interests of the learner and the data generator as a mathematical game. We briefly discuss our modeling decisions and give a short overview of distinct instances of this type of game. We use a rather informal language to explain the principal ideas.

An adversarial interaction between a *learner* and a *data generator* can be considered a game in which one player controls the predictive model whereas the other player controls the process of data generation. Generally, this interaction between both parties is an ongoing process where the data generator produces some data sample, whereupon the learner builds a predictive model, whereupon the data generator changes the data generation process and so on. In each round of the game both players decide for an *action*, *e.g.,* the parameters of the predictive model and the data model, respectively. The outcome of such an interaction can be quantified by *costs* or gains for both players. Under varying assumptions on the players, the way they interact, and their cost functions several games can be distinguished.

To give a brief overview about these games, we first consider the case where we are faced with a setting where both players choose their actions in order to minimize their costs over a (possibly infinite) period of time. Here, both players decide for their actions with respect to the expected outcome of the current and all subsequent repetitions of the game. This setting allows all players to adapt to their opponent's way of playing and, of course, they have to consider this when making their decisions. The main drawback of this game model is not just the complexity of finding a game solution, but also the problem of evaluation which would require to play against a real data generator for several rounds.

In contrast, we may suppose that the ongoing interactions between the learner and the data generator results from playing a new one-shot game in each round. In this case, the players may observe the previous actions of their opponent, but are not assumed to minimize their costs over several rounds by adapting to the opponents' style of playing. In this thesis, we focus on this conceptually simpler model where the players only consider the current round. We call this kind of game a (one-shot) *prediction game*.

Another important element of a game is the order at which the players interact. We differentiate between the following three cases.

**Both players make their decisions simultaneously**

In the first setting we consider simultaneously acting players, that is, both players have to make their decision before observing the opponent's move. In this case, the players have to make an estimate of what the other player is expected to do. This guess is purely based on previously observed actions and the knowledge about the game, *e.g.,* the players' cost functions and their possible moves. We call such games *Nash prediction games* and study them in Chapter 4.

**The learner moves first and the data generator reacts on the learner's action**

In *Stackelberg prediction games*, the learner is supposed to act first by choosing the prediction model. Thereafter, the data generator observes the learner's move and responds by deciding for their own action. As for Nash prediction games, the learner has to consider the opponent's expected move when making their decision. In contrast, the data generator acts in full knowledge of the learner's decision and only has to minimize their costs for the learner's fixed action. In the special case where both players' cost functions are antagonistic, this game reduces to a zero-sum game which is typically used to model a worst-case scenario. This and the more general Stackelberg prediction game are discussed in Chapter 5.

**The learner reacts on the data generator's move**

In this setting, the data generator moves first and produces a sample of data instances which the learner partially observes before making their decision. The term "partially" means that the learner has full access to the instances chosen by the data generator, but not to the corresponding target labels. The task of the learner is now to construct a predictive model based on the data generator's action—the unlabeled data, which the predictive model is exposed to—and prior observations. As the underlying probability distribution of the new sample is potentially different to that of the previously observed data, this game reduces to learning under *covariate shift* which we discuss in Chapter 6.

All of these scenarios assume a different level of knowledge of the players which leads to different games. We formalize and study these games in the subsequent chapters.

## 1.3 Contributions

In this thesis we cast the adversarial prediction problem as a mathematical game and study three game settings which differ with respect to the order at which the players make their decision.

The main contributions of this thesis are as follows:

- We establish the concept of *prediction games*. We formulate the adversarial prediction problem as an one-shot game of complete information by characterizing the players—the learner and the data generator. Therefore, we specify the players' interests, their possible actions, and the way they interact with each other.

  Prediction games enhance existing approaches on learning models which are robust against adversarial changes of the data generation process. Prediction games do *not* rely on the worst-case assumption that the data generator aims to impose the highest possible damage on the learner. Instead, the players' interests are expressed by minimizing their own expected prediction costs. These are approximated by regularized empirical cost functions which depend on both players' actions. The learner's actions is to choose the parameters of the prediction model and the data generator's action is modeled as a perturbation of the training data.

  To the best of our knowledge, games with data-dependent cost functions have not been studied before—neither in the mathematical discipline *game theory*, nor in the area of *machine learning*. This class of games enables us to realistically model the interaction of the players and their knowledge about each other. For instance, the order in which the player decide for their action establishes three differing game settings. We study these three settings, characterize the corresponding game solutions, and present algorithms to solve for the solutions.

- A situation where the learner and the data generator have to decide for an action before observing their opponent's move establishes a static game. Existing work focuses on a minimax strategy where one assumes a maximal malicious data generator. This is generally a too pessimistic assumption. We therefore introduce the *Nash prediction game* to find a Nash-optimal strategy for the learner.

  Playing such a Nash-optimal strategy is reasonable if the Nash equilibrium is unique and both players act rational in the sense of minimizing their *own* costs. We therefore derive easily verifiable conditions under which a unique Nash equilibrium is known to exist.

  We derive two instances of the Nash prediction game which base on logistic regression and the support vector machine (SVM). For the second instance, we derive a newly, twice-continuously differentiable loss function by embedding a trigonometric function into the perceptron loss. We show that for both instances, the Nash equilibrium is unique for sufficiently large regularization parameters.

  We propose two distinct algorithms to solve for the solution of the Nash prediction game and illustrate how to employ kernel functions. We show that the proposed instances of the Nash prediction game significantly outperform their *i.i.d.* baselines as well as a worst-case assuming baseline for the problem of classifying future emails based on training data from the past.

- The more conservative case, where the data generator observes the learner's move before deciding for their own action, has not been addressed in the literature before. This setting establishes a Stackelberg competition that we model as *Stackelberg prediction game*.

  A Stackelberg competition can be expressed as a bilevel optimization problem which is NP-hard in general. We relax the original problem and derive a single-stage optimization problem. We state sufficient conditions under which this minimization problem can be locally solved, for instance, by an SQP solver. By making use of the representer theorem, we show that the proposed optimization criterion can be directly kernelized.

  We reveal that the Stackelberg model generalizes existing prediction models such as the SVM with uneven margins and the SVM for invariances. We evaluate spam filters resulting from three instances of the Stackelberg prediction game on several spam-filtering data sets. We show that the Stackelberg filters outperform other baselines in most of the cases.

- Finally, the learner may act in full knowledge of the data generator's action. From perspective of the learner, this game scenario reduces to learning under *covariate shift*.

  We derive a MAP estimator under covariate shift which is based on an unbiased likelihood function. This model equals a regular MAP estimator with resampled training instances. We show that these resampling weights equal the testing-to-training density ratio and illustrate how to infer these weights from the data without estimating the densities of the training and test data separately.

  We derive an optimization criterion to jointly estimate the parameters of the prediction model as well as of the model which computes the resampling weights. This complements the predominant sequential approach of first estimating the covariate shift between the training and the test data, and then learning a predictive model based on a resampled version of the training data.

  To solve the integrated optimization problem, we propose an inexact linesearch method and state the gradient of the objective. We show that the objective is generally non-convex in all parameters and, therefore, derive a convex two-stage approximation. This method is conceptually simple and can be used to, firstly, find the resampling weights and, subsequently, learn almost any standard predictive model.

  We empirically explore methods for learning under covariate shift including the newly derived methods as well as existing baseline methods. We observe that, for the task of filtering email spam, all covariate shift-compensating methods do hardly outperform the *i.i.d.* baselines. We empirically verify potential reasons for this negative result and conclude that for spam filter, certain requirements on the use of covariate shift models may not be met.

## 1.4 Own Previously Published Work

This thesis builds on several previously published articles. In this section, we provide a list of these publications and detail on own contributions.

[14] Brückner, M., Bickel, S., Scheffer, T.: Optimal spamming: Solving a family of adversarial classification games. In: Proceedings of NIPS Workshop on Machine Learning in Adversarial Environments for Computer Security (2007)

In this work, I cast the adversarial prediction problem as a game between a learner and an adversary. I discuss potential game solutions depending on the players' cost functions and their level of rationality. Chapter 3 generalizes the problem formulation of this publication.

[17] Brückner, M., Scheffer, T.: Nash equilibria of static prediction games. In: Advances in Neural Information Processing Systems (NIPS). MIT Press (2009)

The paper resumes the problem setting of our previous publication [14]. I discuss static one-shot prediction games where the players are not required to have entirely antagonistic interests. I derived and implemented a method to solve for the solution of this game, that is, a Nash equilibrium, and conducted several experiments on email spam filtering.

[16] Brückner, M., Kanzow, C., Scheffer, T.: Static prediction games for adversarial learning problems. Journal of Machine Learning Research (JMLR) 13, 2589–2626 (2012)

Chapter 4 builds on that publication which extends our prior work on Nash prediction games [17]. Christian Kanzow and I derive sufficient conditions for the existence of a unique solution, which is an essential requirement to apply static prediction games. Complementing [17], I derived a second instance of the Nash prediction game, called Nash support vector machine which is based on a newly proposed loss function. I empirically studied the convergence, performance, and runtime behavior of the new as well as existing baseline methods.

[18] Brückner, M., Scheffer, T.: Stackelberg games for adversarial prediction problems. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), San Diego, CA, USA. ACM Press (2011)

In this paper, I extend the Nash model [14, 17] to dynamic prediction games where the players act non-simultaneously. I formalize the problem as a bilevel optimization problem and derive a relaxed single-stage minimization problem, which can be solved with standard tools. I implemented the proposed method and conducted experiments on several real-world spam data sets. The findings of this work are presented in Chapter 5.

[7] Bickel, S., Brückner, M., Scheffer, T.: Discriminative learning for differing training and test distributions. In: Proceedings of International Conference on Machine Learning (ICML). ACM Press, Oregon, USA (2007)

The paper presents a method for discriminative learning under covariate shift based on resampling the training instances. Steffen Bickel, Tobias Scheffer, and I derived an integrated model to jointly estimate models to predict the target concept as well as the resampling weights. I derived and implemented the Newton gradient descent method for the integrated logistic regression and the kernelization of the method. Steffen Bickel conducted the experiments.

[8] Bickel, S., Brückner, M., Scheffer, T.: Dataset Shift in Machine Learning, chap. Discriminative learning under covariate shift with a single optimization problem, pp. 161–178. MIT Press (2009)

This article extends our previous work on covariate shift [7] to arbitrary loss functions—especially the family of exponential loss functions—and provides further experiments. Steffen Bickel generalized the integrated model of [7], implemented the extended model, and conducted the new experiments. I derived sufficient conditions for the convexity of the optimization criterion.

[9] Bickel, S., Brückner, M., Scheffer, T.: Discriminative learning under covariate shift. Journal of Machine Learning Research (JMLR) 10, 2137–2155 (2009)

The paper builds on our previous publications on learning under covariate shift [7,8]. Steffen Bickel and I derived an efficient and flexible two-stage approximation of the previously published integrated model and reveal its relation to kernel mean matching. Steffen Bickel implemented this method and conducted new experiments based on a revised parameter tuning process. Chapter 6 presents and partially extends our findings of this publication.

[15] Brückner, M., Haider, P., Scheffer, T.: Highly scalable discriminative spam filtering. In: Proceedings of 15th Text REtrieval Conference (TREC). National Institute of Standards and Technology (NIST) (2006)

Peter Haider, Tobias Scheffer, and I took part in the spam filtering challenge TREC Spam Track 2006. The paper briefly presents the concepts underlying our submitted filter. This includes the decoding, preprocessing, and tokenization of emails as well as the winnow algorithm to infer the filter from millions of training emails. I implemented the learning algorithm and the parsing routines, and conducted the experiments of the first task (periodical offline training) whereas Peter Haider conducted the experiments of the second task (active learning). The presented steps of feature extraction are used in the experiments of this thesis.

## 1.5 Outline

This thesis relies on concepts from machine learning as well as game theory. To this end, Chapter 2 gives a brief introduction into some aspects of learning theory to build prediction models from data. In Chapter 3, we then present principles of game theory. Later in the chapter, the presented concepts of machine learning and game theory are combined to model the adversarial prediction problem as a game between a learner and a data generator. We identify three distinct game settings, depending on the order at which the players make their decision. The first case, where both player decide simultaneously, is called Nash prediction game. We study this setting in Chapter 4. Chapter 5 covers the Stackelberg prediction game which addresses the case where the data generator acts after the learner and decides on their action in full knowledge on the learner's move. The final case, where the data generator moves first and the learner reacts, is discussed in Chapter 6. In this setting, the game between learner and data generator reduces to learning under covariate shift. Chapter 7 concludes this thesis.

# 2 Learning Predictive Models

This thesis centers around the derivation and analysis of methods to build adversary-aware prediction models. To this end, this chapter gives a brief introduction into some aspects of learning theory to build prediction models from data.

In practice one is often confronted with the problem of estimating some unknown *target* attribute of a given *object*, for instance, whether an email (the object) belongs to class spam or not (the target attribute). An object $x \in \mathcal{X}$ is an element of the *input space* $\mathcal{X}$ which could be any data domain such as documents, stock prices, images, database records, or measurements. The target attribute $y \in \mathcal{Y}$ is typically a categorical or an one-dimensional numerical value of *output space* $\mathcal{Y}$.

The task of predicting the target attribute of some given object can be solved by *supervised learning*. Typically, in the supervised learning problem, one firstly determines a relationship $h : \mathcal{X} \to \mathcal{Y}$ between objects and targets based on a given sample of $n$ objects $\boldsymbol{x} := (x_1, \ldots, x_n) \in \mathcal{X}^n$ with corresponding targets $\boldsymbol{y} := (y_1, \ldots, y_n) \in \mathcal{Y}^n$. In a second step, one applies this mapping $h$ to new, previously unseen objects $\dot{x}_j \in \mathcal{X}$, in order to predict the corresponding (unobserved) target attribute $\dot{y}_j \in \mathcal{Y}$ for $j = 1, \ldots, l$. The desired mapping $h$ is called a *prediction model*, the observed object-target pairs $D = \{(x_i, y_i)\}_{i=1}^n$ are referred as *training data*, and the new object-target pairs form a set of *test data* $\dot{D} = \{(\dot{x}_j, \dot{y}_j)\}_{j=1}^l$. To separate between training and test variables, we use a dot which indicates whether an object or target is observed in the first step at training time or in the second step at application time.

The main challenge in supervised learning is to infer a prediction model from the given data. To this end, an object-target pair $(x, y)$ is typically considered to be the result of some unknown stochastic data generation process. That is, $(x, y)$ is a realization of the corresponding joint *random variable* $\boldsymbol{XY}$ where any object $x \in \mathcal{X}$ and target $y \in \mathcal{Y}$ can be an assignment of the single random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively. The probability of a realization $(x, y)$ is denoted by $p_{\boldsymbol{XY}}(x, y)$ where $p_{\boldsymbol{XY}}$ is some unknown *probability density function* (PDF) with corresponding *cumulative distribution function* (CDF) $P_{\boldsymbol{XY}}$.

A common assumption in supervised learning is that the training object-target pairs $(x_i, y_i)$ are independent and identically distributed (*i.i.d.*) for $i = 1, \ldots, n$. This means, that all training objects with corresponding targets are drawn statistically independently of each other from the same underlying *training distribution* $P_{\boldsymbol{XY}}$. Likewise, all test object-target pairs $(\dot{x}_j, \dot{y}_j)$ are considered to be *i.i.d.* according to some *test distribution* which is typically assumed to equal the training distribution. Hence, the given training data establish a *representative sample* of the training distribution and, under the above assumption, of the test distribution, too.

A straightforward (generative) approach to supervised learning is to estimate the density of data distribution $P_{\boldsymbol{XY}}$ from this sample and to compute the probability of $x$ being assigned to target $y$ by using the definition of the conditional probability,

$$p_{\boldsymbol{Y}|\boldsymbol{X}=x}(y) = \frac{p_{\boldsymbol{XY}}(x,y)}{p_{\boldsymbol{X}}(x)} = \frac{p_{\boldsymbol{XY}}(x,y)}{\int_{\mathcal{Y}} p_{\boldsymbol{XY}}(x,y')\,\mathrm{d}y'}. \tag{2.1}$$

In this case, an optimal prediction is to map $x$ to target $y^*$ with largest probability $p_{\boldsymbol{Y}|\boldsymbol{X}=x}(y^*)$, that is,

$$h^*(x) := \operatorname*{argmax}_{y\in\mathcal{Y}} p_{\boldsymbol{Y}|\boldsymbol{X}=x}(y). \tag{2.2}$$

However, estimating the density $p_{\boldsymbol{XY}}$ from a finite sample $D$ is a non-trivial problem. To see this, consider the empirical estimate of $p_{\boldsymbol{XY}}$,

$$p_D(x,y) := \frac{1}{n}\left|\{i \in \{1,...,n\} \mid x_i = x,\ y_i = y\}\right|, \tag{2.3}$$

and the corresponding cumulative distribution function,

$$P_D(x,y) := \frac{1}{n}\sum_{i=1}^{n}\left|\{i \in \{1,...,n\} \mid x_i \leq x,\ y_i \leq y\}\right|.$$

Even though, this estimate is consistent in the sense that $P_D$ uniformly converges to $P_{\boldsymbol{XY}}$ if the sample size gets larger (see, for instance, Theorem 19.1 in [73]), the induced prediction model would be meaningless, as $p_D$ assigns zero probability to all unseen object-target pairs.

In the following two sections, we will discuss strategies to solve this problem, namely the principle of *risk minimization* and the *Bayesian* approach to learning. We present the concept of generalized linear decision functions for several learning problems in Section 2.3. In Section 2.4 and 2.5 we state commonly used loss functions and regularizers for risk minimization, and in Section 2.6 we introduce the concept of kernel functions. Finally, we present an exemplary algorithm to estimate the parameters of a predictive model in Section 2.7.

## 2.1 Risk Minimization

A common approach to learning is to reformulate the learning problem by introducing a discriminant *decision function* $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. This function models the relationship between $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ by

$$h(x) := \operatorname*{argmax}_{y\in\mathcal{Y}} f(x,y) \tag{2.4}$$

where $h(x)$ is an estimate of $y$ for a given $x$. In addition, an *error functional* $\Delta : F \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$ is introduced to measure the disagreement between label $y$ and prediction $h(x)$. The error functional enables us to assess the quality of a single decision function $f$ and

its corresponding hypothesis $h$, respectively, by the expected *generalization error* or risk $\theta[f, p_{\boldsymbol{XY}}]$ given by

$$\theta[f, p_{\boldsymbol{XY}}] := \mathrm{E}_{\boldsymbol{XY}}[\Delta[f, x, y]] = \int_{\mathcal{X}\times\mathcal{Y}} \Delta[f, x, y]\, p_{\boldsymbol{XY}}(x, y)\, \mathrm{d}(x, y). \tag{2.5}$$

In case of classification, where $\mathcal{Y}$ is finite, when choosing the zero-one error

$$\Delta^{0/1}[f, x, y] := \begin{cases} 0 & \text{if } y = \operatorname{argmax}_{y'\in\mathcal{Y}} f(x, y') \\ 1 & \text{otherwise} \end{cases} \tag{2.6}$$

the generalization error equals the probability of mispredicting $y$.

In risk minimization one seeks to choose one particular decision function $\hat{f}$ from the *decision function space* $F$ which minimizes the generalization error, that is,

$$\hat{f} := \operatorname*{argmin}_{f\in F} \theta[f, p_{\boldsymbol{XY}}].$$

Unfortunately, evaluating $\theta[f, p_{\boldsymbol{XY}}]$ again requires the knowledge of $p_{\boldsymbol{XY}}$ and, consequently, the learning task amounts to identifying a proper estimator of this risk based on the observed training sample $D$ and to minimize it with respect to $f \in F$. An unbiased estimate of $\theta[f, p_{\boldsymbol{XY}}]$ is obtained by replacing $p_{\boldsymbol{XY}}$ in (2.5) by its empirical estimate (2.3) which leads to the *empirical error* $\theta[f, p_D]$ with

$$\theta[f, p_D] := \frac{1}{n}\sum_{i=1}^{n} \Delta[f, x_i, y_i] \tag{2.7}$$

and $\theta[f, p_D] \xrightarrow{\text{a.s.}} \theta[f, p_{\boldsymbol{XY}}]$. Minimizing this quantity with respect to $f \in F$ generally poses an *ill*-posed optimization problem, that is, $\theta[f, p_D]$ is not continuous in $f$ in general and a minimizer $f^*$ of the empirical error might not be unique, or worse, might not exist. This renders the problem to be potentially NP-hard or even impossible to solve. Moreover, the empirical error $\theta[f, p_D]$ is likely to underestimate the generalization error $\theta[f, p_{\boldsymbol{XY}}]$ in practice.

Regularization theory (*cf.* Section 2.5) addresses these problems by introducing a regularization term $\Omega : F \to \mathbb{R}^+$, that is,

$$\hat{\theta}[f, p_D] := \theta[f, p_D] + \rho\Omega[f] = \frac{1}{n}\sum_{i=1}^{n} \Delta[f, x_i, y_i] + \rho\Omega[f], \tag{2.8}$$

and constraining the error functional $\Delta$ so that $\hat{\theta}[f, p_D]$ is continuous in $f$ for any fixed $D$. The *regularizer* $\Omega[f]$ is chosen so that it implicitly restricts the decision function space $F$ to a compact set. Here, the predefined parameter $\rho$ controls the amount of regularization, *i.e.,* a larger value of $\rho$ implies a more restrictive decision function space. For $\rho > 0$, a

minimizer

$$\hat{f}_{\text{ERM}} := \operatorname*{argmin}_{f \in F} \hat{\theta}[f, p_D] \tag{2.9}$$

of the *regularized empirical error* can be obtained efficiently for an appropriately chosen decision function space $F$, error functional $\Delta$, and regularizer $\Omega$. Before we discuss these components in detail, we shall introduce the Bayesian approach to learning.

## 2.2 Learning in the Bayesian Framework

The Bayesian approach differs from risk minimization such that we are directly addressing the probability of target $y \in \mathcal{Y}$, given any object $x \in \mathcal{X}$. We assume that there exist a true relationship $h^* : \mathcal{X} \to \mathcal{Y}$ between the objects and the targets, that is, the object-target pairs are drawn *i.i.d.* from some unknown distribution

$$P_{\boldsymbol{XY}|\boldsymbol{H}=h^*}(x, y) = P_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h^*}(y) P_{\boldsymbol{X}=x}(x).$$

The goal of Bayesian learning is *not* necessarily to identify $h^*$, but to estimate the density of $P_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h^*}(y)$ from the available data. Directly modeling this quantity enables us not only to predict the most likely target value of a given object $x$, but also to state the confidence of the prediction.

The underlying relationship between objects and targets is unknown, however, it is represented in the training objects $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{X}^n$ and the corresponding targets $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathcal{Y}^n$. To estimate the density $p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h^*}(y)$ we consider the conditional $p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{Y}^n=\boldsymbol{y}}$. By applying the theorem of total probability, it can be expanded as in (2.10), and (2.11) follows from the chain rule. Equation 2.12 exploits the facts that, because of the *i.i.d.* assumption, target $y$ is independent of the other training instances given the relationship $h$ between the objects and targets, and that $h$ is independent of object $x$ as long as $y$ is unobserved.

$$\begin{aligned}
p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{Y}^n=\boldsymbol{y}}(y) &= \int_H p_{\boldsymbol{YH}|\boldsymbol{X}=x,\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{Y}^n=\boldsymbol{y}}(y, h)\,\mathrm{d}h && (2.10)\\
&= \int_H p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{Y}^n=\boldsymbol{y},\boldsymbol{H}=h}(y) p_{\boldsymbol{H}|\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{Y}^n=\boldsymbol{y}}(h)\,\mathrm{d}h && (2.11)\\
&= \int_H p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h}(y) p_{\boldsymbol{H}|\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{Y}^n=\boldsymbol{y}}(h)\,\mathrm{d}h && (2.12)
\end{aligned}$$

If we had access to an infinite sample of object-target pairs, only the true relationship $h^*$ would still be consistent with the data so that $p_{\boldsymbol{H}|\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{Y}^n=\boldsymbol{y}}$ would become a single-point distribution which is one for $\boldsymbol{H} = h^*$ and zero otherwise. Hence, under the assumption that $h^*$ is in the set of considered models $H$, the above conditional $p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{Y}^n=\boldsymbol{y}}$ is a consistent estimate of $p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h^*}(y)$ in the sense that $p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{Y}^n=\boldsymbol{y}} \overset{n\to\infty}{=} p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h^*}$.

The first term $p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h}(y)$ in (2.12) specifies the probability of observing target $y \in \mathcal{Y}$ for a given object $x \in \mathcal{X}$ and any fixed model $h \in H$. The second term $p_{\boldsymbol{H}|\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{Y}^n=\boldsymbol{y}}(h)$ states the a-posteriori probability of $h$, that is, the probability of model $h$, after having

observed the training data. According to Bayes' rule, the posterior of a model can be stated in terms of its label *likelihood*, *prior*, and *evidence*, that is,

$$p_{\boldsymbol{H}|\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{Y}^n=\boldsymbol{y}}(h) = \frac{\overbrace{p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{H}=h}(\boldsymbol{y})}^{\text{label likelihood of } h}\ \overbrace{p_{\boldsymbol{H}}(h)}^{\text{prior of } h}}{\underbrace{\int_H p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{H}=h'}(\boldsymbol{y})\,\mathrm{d}h'}_{\text{evidence of } H}}. \tag{2.13}$$

The label likelihood of $h$ states how plausible it is to observe target tuple $\boldsymbol{y} = (y_1, \ldots, y_n)$ if we are given a tuple of objects $\boldsymbol{x} = (x_1, \ldots, x_n)$. Here, the underlying relationship between an object $x$ and target $y$ is expressed by model $h$, that is, this term covers all information about $h$ which can be obtained from the training data. In contrast, the prior of $h$ contains all information on the true relationship before having observed the training data. This term expresses the prior belief or knowledge of $h$, *e.g.,* an uniform prior presumes that all mappings $h$ in *hypothesis space* $H$ are equally plausible whereas all $h \notin H$ are impossible and have zero a-priori probability. Finally, the evidence measures the complexity of hypothesis space $H$, that is, the flexibility of models $h$ in $H$. This term is constant for any fixed hypothesis space, so that (2.12) resolves to

$$p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{Y}^n=\boldsymbol{y}}(y) \quad \propto \quad \int_H p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h}(y) p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{H}=h}(\boldsymbol{y}) p_{\boldsymbol{H}}(h)\,\mathrm{d}h. \tag{2.14}$$

This decomposition leads to a *discriminative* predictive model, since we are only modeling the conditional $p_{\boldsymbol{Y}|\boldsymbol{X}=x}$, that is, we solely derive a functional relation which explains the discriminating properties of objects $x \in \mathcal{X}$. In contrast, a *generative* model involves an estimate of the complete data density $p_{\boldsymbol{XY}}$ so that, after the training process, we could generate new object-target pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$. This intrinsically model-based approach additionally assess how well $h$ models the distribution of input instances $x$. As this is irrelevant for the task of correctly predicting $y$ at hand, we focus on discriminative models in this thesis.

For several learning problems, for instance, linear regression, all terms of the integral in (2.14) can be expressed by density functions of a specific functional form so that the above integral can be solved analytically. However, in the absence of an analytical solution, this integral poses a crucial problem. Even though, there exist various strategies to solve integration numerically, this would have to be performed for each new object $x$ and target $y$ which is generally intractable.

One approach to avoid the difficulty of numerical integration is to assume that the posterior of $h$ concentrates its probability mass around the true relationship $h^*$. In this case, $p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{H}=h}(\boldsymbol{y}) p_{\boldsymbol{H}}(h)$ can be approximated by a single-point distribution which is one if $\boldsymbol{H}$ equals some point estimate $\hat{h}$ and zero otherwise, that is,

$$\int_H p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h}(y) p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{H}=h}(\boldsymbol{y}) p_{\boldsymbol{H}}(h)\,\mathrm{d}h \;\approx\; p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=\hat{h}}(y). \tag{2.15}$$

If we again assume that $h^*$ is in $H$, this approximation is consistent in the sense that (2.15) becomes an equation and $\hat{h}$ equals $h^*$ for $n \to \infty$. By this approximation, one splits the prediction task into a *training* phase to find point estimate $\hat{h}$ and an *inference* phase where the target of a given object is predicted using $p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=\hat{h}}(y)$.

Common point estimates are the Bayes point and the mean, median, or mode of the posterior $p_{\boldsymbol{H}|\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{Y}^n=\boldsymbol{y}}(h) \propto p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{H}=h}(\boldsymbol{y})p_{\boldsymbol{H}}(h)$. The posterior mode is referred as *maximum-a-posteriori* (MAP) estimate or, in case of an uniform prior, as *maximum-likelihood* (ML) estimate. All of these estimates are consistent, that is, they converge to the true relationship $h^*$ as the training sample gets larger, they are asymptotically unbiased, and they are efficient. In practice, however, their behavior may differ depending on the shape of the posterior. For a discussion on Bayesian point estimates see, for instance, Chapter 10 of [73].

Bayesian learning is conceptually different from risk minimization, however, there exists a strong link between both approaches. Let us, for instance, consider the MAP estimate of the decision function,

$$\hat{f}_{\mathrm{MAP}} := \operatorname*{argmax}_{f \in F} p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{F}=f}(\boldsymbol{y})p_{\boldsymbol{F}}(f), \tag{2.16}$$

with corresponding MAP hypothesis (*cf.* Equation 2.4)

$$h_{\mathrm{MAP}}(x) := \operatorname*{argmax}_{y \in \mathcal{Y}} \hat{f}_{\mathrm{MAP}}(x, y).$$

In addition, let us assume a label likelihood and prior of the exponential family,

$$p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{F}=f}(y) := \frac{\exp(-\frac{1}{n}\Delta^{0/1}[f, x, y])}{\int_{\mathcal{Y}} \exp(-\frac{1}{n}\Delta^{0/1}[f, x, y'])\,\mathrm{d}y'} \tag{2.17}$$

$$p_{\boldsymbol{F}}(f) := \frac{\exp(-\rho\Omega[f])}{\int_H \exp(-\rho\Omega[f'])\,\mathrm{d}f'} \tag{2.18}$$

where $\Delta^{0/1}$ and $\Omega$ are defined as in the previous section. Then, the following remark shows the equivalence of Bayesian learning and risk minimization.

**Remark 2.1.** Let the label likelihood and prior of decision function $f$ be defined as in (2.17) and (2.18) with hypothesis $h(x) := \operatorname{argmax}_{y \in \mathcal{Y}} f(x, y)$, then the MAP decision function $\hat{f}_{\mathrm{MAP}}$ equals the regularized empirical error estimate $\hat{f}_{\mathrm{ERM}}$: Starting with (2.19), Equation 2.20 follows since the logarithmic function is strictly monotonically increasing and, consequently, the maximizing argument $\hat{f}_{\mathrm{MAP}}$ of the posterior is also a minimizing argument of the negative log-posterior and vice versa. Equation 2.21 exploits the fact that the training pairs $(x_i, y_i)$ are drawn *i.i.d.*, so that the label likelihood of $f$ factorizes into $n$ per-instance label likelihoods $p_{\boldsymbol{Y}|\boldsymbol{X}=x_i,\boldsymbol{F}=f}(y_i)$. Inserting (2.17) and (2.18) gives (2.22). Finally, (2.23) holds as the logarithmized denominators of the label likelihood and the prior are constant for any fixed training sample and the zero-one error defined in (2.6).

$$\hat{f}_{\text{MAP}} = \operatorname*{argmax}_{f \in F} p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{F}=f}(\boldsymbol{y}) p_{\boldsymbol{F}}(f) \tag{2.19}$$

$$= \operatorname*{argmin}_{f \in F} -\log p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x},\boldsymbol{F}=f}(\boldsymbol{y}) - \log p_{\boldsymbol{F}}(f) \tag{2.20}$$

$$= \operatorname*{argmin}_{f \in F} -\log \prod_{i=1}^{n} p_{\boldsymbol{Y}|\boldsymbol{X}=x_i,\boldsymbol{F}=f}(y_i) - \log p_{\boldsymbol{F}}(f) \tag{2.21}$$

$$= \operatorname*{argmin}_{f \in F} \sum_{i=1}^{n} -\log \frac{\exp(-\frac{1}{n}\Delta^{0/1}[f, x_i, y_i])}{\int_{\mathcal{Y}} \exp(-\frac{1}{n}\Delta^{0/1}[f, x_i, y']) \,\mathrm{d}y'} - \log \frac{\exp(-\rho\Omega[f])}{\int_F \exp(-\rho\Omega[f']) \,\mathrm{d}f'} \tag{2.22}$$

$$= \operatorname*{argmin}_{f \in F} \sum_{i=1}^{n} \frac{1}{n}\Delta^{0/1}[f, x_i, y_i] + \rho\Omega[f] \;=\; \hat{f}_{\text{ERM}} \tag{2.23}$$

As the objective in (2.23) equals the regularized empirical error $\hat{\theta}[f, p_D]$, the corresponding minimizer $\hat{f}_{\text{ERM}}$ equals the MAP estimate $\hat{f}_{\text{MAP}}$ as well. ◇

Having presented the principle ideas on how to reformulate the learning problem from a Bayesian perspective, we now turn back to the more intuitive framework of risk minimization to study the single components of prediction models. We start with the decision function space $F$.

## 2.3 Linear Decision Functions

A common class of models $f$ are *linear decision functions* which rely on inner products in some *feature space*, also referred as Hilbert space $\mathcal{H}$. We consider a finite-dimensional feature space $\mathcal{H} := \mathbb{R}^m$ and assume the existence of an $m$-dimensional numeric representation $\boldsymbol{\phi}(x) := [\phi_1(x), \ldots, \phi_m(x)]^\mathsf{T}$ of each object $x \in \mathcal{X}$. This vector-valued function $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{H}$ is called a *feature mapping* and $\phi_i(x) \in \mathbb{R}$ for $i = 1, \ldots, n$ are referred as the *features* of $x$. In case of vectorial objects $x \in \mathcal{X} \subseteq \mathbb{R}^m$, the feature mapping may reduce to the identity mapping $\boldsymbol{\phi}(x) := x$. Considering only linear mappings $f$ seems overly restrictive. However, as feature mapping $\boldsymbol{\phi}$ can be an arbitrary function, decision function $f$ is generally non-linear in the input space. Indeed, any smooth function in the input space can be expressed by a linear mapping $f$ in an appropriate feature space $\mathcal{H}$ induced by mapping $\boldsymbol{\phi}$. In Section 2.6, we discuss how to construct such powerful feature mappings for objects $x \in \mathcal{X}$ solely based on a similarity measure over the input space $\mathcal{X}$.

Based on a given feature mapping $\boldsymbol{\phi}$, one can define a decision function space $F$ for each specific learning task. For instance, in *binary classification* where $\mathcal{Y} := \{-1, +1\}$, the decision function is typically defined by $f(x, y) := yg(x)$ where $g(x) := \mathbf{w}^\mathsf{T}\boldsymbol{\phi}(x)$. Consequently, the resulting hypothesis (2.4) resolves to

$$h(x) := \operatorname*{argmax}_{y' \in \mathcal{Y}} f(x, y') = \operatorname*{argmax}_{y' \in \mathcal{Y}} y\mathbf{w}^\mathsf{T}\boldsymbol{\phi}(x) = \operatorname{sign} \mathbf{w}^\mathsf{T}\boldsymbol{\phi}(x)$$

with corresponding decision function space $F := \{(x, y) \mapsto y\mathbf{w}^\mathsf{T}\boldsymbol{\phi}(x) \mid \mathbf{w} \in \mathcal{W}\}$ and *parameter space* $\mathcal{W} \subseteq \mathbb{R}^m$.

Similarly, for *multi-class classification* where $\mathcal{Y} := \{1, \ldots, k\}$ for some finite $k \in \mathbb{N}$, the decision function can be defined by a combination of $k$ linear mappings $g_j(x) := \mathbf{w}_j^\mathsf{T}\boldsymbol{\phi}(x)$ for $j = 1, \ldots, k$, that is, $f(x, y) := g_y(x) = \mathbf{w}_y^\mathsf{T}\boldsymbol{\phi}(x)$ with corresponding hypothesis

$$h(x) := \operatorname*{argmax}_{y' \in \mathcal{Y}} f(x, y') = \operatorname*{argmax}_{y' \in \mathcal{Y}} \mathbf{w}_{y'}^\mathsf{T}\boldsymbol{\phi}(x)$$

and decision function space $F := \{(x, y) \mapsto \mathbf{w}_y^\mathsf{T}\boldsymbol{\phi}(x) \mid \mathbf{w}_j \in \mathcal{W} \text{ for } j = 1, \ldots, k\}$. In comparison to binary classification, the number of parameters which have to be estimated from the data increases from $m$ to $k \cdot m$.

A final example is *linear regression* where $\mathcal{Y} := \mathbb{R}$. Here, the result of mapping $g(x)$ is typically used as the prediction of $y$. This can be expressed by defining mapping $f(x, y) := -(g(x) - y)^2 = -(\mathbf{w}^\mathsf{T}\boldsymbol{\phi}(x) - y)^2$ so that

$$h(x) := \operatorname*{argmax}_{y' \in \mathcal{Y}} f(x, y') = \operatorname*{argmin}_{y' \in \mathcal{Y}} (\mathbf{w}^\mathsf{T}\boldsymbol{\phi}(x) - y')^2 = \mathbf{w}^\mathsf{T}\boldsymbol{\phi}(x)$$

and $F := \{(x, y) \mapsto -(\mathbf{w}^\mathsf{T}\boldsymbol{\phi}(x) - y)^2 \mid \mathbf{w} \in \mathcal{W}\}$.

The algorithms presented in this thesis focus on binary classification. They may be applicable to other learning task by adapting the decision function space accordingly. We only consider hypotheses of the form[1] $h_\mathbf{w}(x) := \operatorname{sign} g_\mathbf{w}(x)$ with

$$g_\mathbf{w}(x) := \mathbf{w}^\mathsf{T}\boldsymbol{\phi}(x)$$

and $\mathbf{w} \in \mathcal{W}$. For this choice, the error functional is generally considered to be a product of a loss term $\ell(g_\mathbf{w}(x), y)$, which penalizes mispredictions, and an object- and target-specific weighting term $c(x, y)$, which states the importance of correctly predicting the given instance, that is,

$$\Delta[f_\mathbf{w}, x, y] := c(x, y)\ell(g_\mathbf{w}(x), y). \tag{2.24}$$

As the decision function $f_\mathbf{w}$ and the corresponding hypothesis $h_\mathbf{w}$ are uniquely defined by weight vector $\mathbf{w}$, in the following we redefine the (regularized) empirical error and the regularizer to be functions of $\mathbf{w} \in \mathcal{W}$ rather than functionals of decision function $f_\mathbf{w} \in F$ with $f_\mathbf{w}(x, y) = y g_\mathbf{w}(x) = y\mathbf{w}^\mathsf{T}\boldsymbol{\phi}(x)$. In addition, we substitute the error functional $\Delta$ by (2.24) so that (2.8) reduces to

$$\hat{\theta}(\mathbf{w}, D) := \frac{1}{n}\sum_{i=1}^{n} c(x_i, y_i)\ell(g_\mathbf{w}(x_i), y_i) + \rho\Omega(\mathbf{w}). \tag{2.25}$$

Whereas $c(x_i, y_i)$ denote fixed weighting factors which do not depend on the parameter vector $\mathbf{w}$, *loss function* $\ell$ and *regularization function* $\Omega$ are chosen such to obtain a good estimate of the generalization error while restricting the computational complexity of the learning problem. We discuss popular choices of these functions in the following sections.

[1] We use subscript $\mathbf{w}$ to indicate that a function or operator is parameterized by vector $\mathbf{w}$.

## 2.4 Loss Functions

A loss function $\ell : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}^+$ is used to assess the appropriateness of a decision function, that is, the discrepancy between the true class $y$ of $x$ and the decision function-induced prediction $h_{\mathbf{w}}(x)$. In classification, the zero-one loss which is given in the following definition, is the theoretically optimal choice as, in the limit, it results in a hypothesis with minimal misprediction probability.

**Definition 2.1.** *For a binary classifier $h_{\mathbf{w}} : \mathcal{X} \to \{-1, +1\}$ of the form $h_{\mathbf{w}}(x) = \operatorname{sign} g_{\mathbf{w}}(x)$ with $g_{\mathbf{w}}(x) = \mathbf{w}^\mathsf{T}\boldsymbol{\phi}(x)$ and corresponding weight vector $\mathbf{w} \in \mathcal{W}$, the* zero-one loss *is given by*

$$\ell^{0/1}(g_{\mathbf{w}}(x), y) := \begin{cases} 0 & \text{if } y g_{\mathbf{w}}(x) > 0 \\ 1 & \text{if } y g_{\mathbf{w}}(x) \leq 0 \end{cases} .$$

However, minimizing the (regularized) empirical zero-one error poses an NP-hard problem for which reason other loss functions are employed in practice. In the following, we present the most commonly used loss functions and a new loss function for binary classification (*cf.* Figure 2.1). The proofs of the presented propositions can be found in the appendix.

The choice of a particular loss function mainly depends on the learning task and theoretical implications such as the continuity condition in risk minimization or the normalization constraint in the Bayesian framework. In addition, to efficiently compute the minimizer $\mathbf{w}^*$ of the regularized empirical error $\hat{\theta}(\mathbf{w}, D)$ (*cf.* Equation 2.25), the loss function is typically required to be convex and continuously differentiable or even twice-continuously differentiable with respect to $\mathbf{w} \in \mathcal{W}$.

One of the most commonly used loss functions for binary classification is the hinge loss which is a piecewise-linear convex upper bound on the zero-one loss.

**Definition 2.2.** *For a binary classifier $h_{\mathbf{w}} : \mathcal{X} \to \{-1, +1\}$ of the form $h_{\mathbf{w}}(x) = \operatorname{sign} g_{\mathbf{w}}(x)$ with $g_{\mathbf{w}}(x) = \mathbf{w}^\mathsf{T}\boldsymbol{\phi}(x)$ and corresponding weight vector $\mathbf{w} \in \mathcal{W}$, the* hinge loss *is given by*

$$\ell^{\mathrm{h}}(g_{\mathbf{w}}(x), y) := \max(0, 1 - y g_{\mathbf{w}}(x)) = \begin{cases} 0 & \text{if } y g_{\mathbf{w}}(x) > 1 \\ 1 - y g_{\mathbf{w}}(x) & \text{if } y g_{\mathbf{w}}(x) \leq 1 \end{cases} .$$

The hinge loss linearly penalizes not only misclassification where $f_{\mathbf{w}}(x, y) < 0$ with $f_{\mathbf{w}}(x, y) = y g_{\mathbf{w}}(x)$, but also correct predictions if the corresponding decision value $f_{\mathbf{w}}(x, y)$ is less than 1, that is, if the functional *margin* between object $x$ and the decision surface $G_{\mathbf{w}} := \{\boldsymbol{\phi}(x) \in \mathbb{R}^m \mid \mathbf{w}^\mathsf{T}\boldsymbol{\phi}(x) = 0\}$ is small. Minimizing $\hat{\theta}(\mathbf{w}, D)$ together with the hinge loss function yields a large margin classifier referred as *support vector machine* (SVM). The hinge loss is not continuously differentiable in $\mathbf{w} \in \mathcal{W}$ so that direct gradient-based minimization is not applicable[2] and one often considers the dual problem which is a quadratic program. Alternatively, some variants of the SVM employ the squared hinge loss which is once-continuously differentiable. Though, one main drawback of the squared hinge loss is

[2] In practice, the SVM objective can be efficiently minimized in the primal using subgradient or bundle methods which do not rely on the differentiability condition.

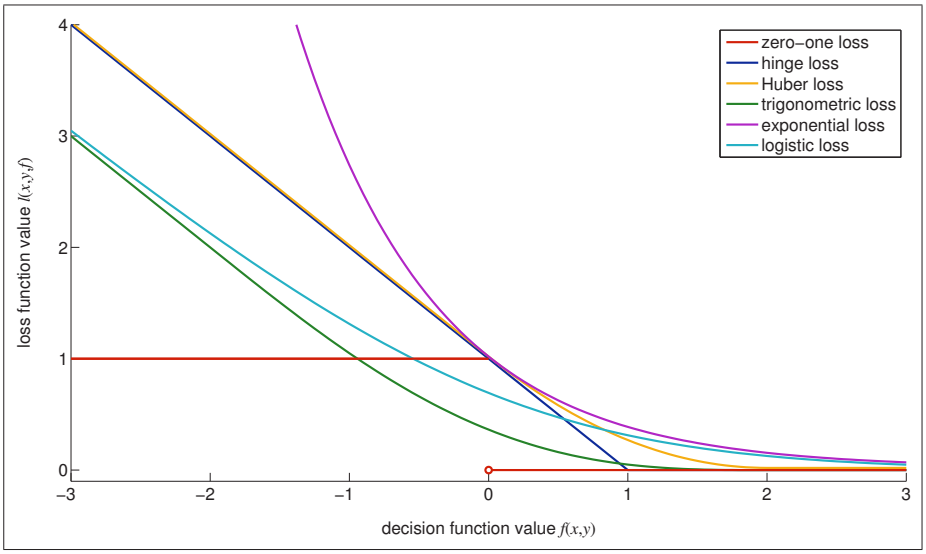


**Figure 2.1:** Loss functions for binary classification with $\delta = 1$ for the Huber loss and $\delta = 2$ for the trigonometric loss.

its sensitivity to outliers in the data, as misclassified objects are penalized quadratically. Another approach is to redefine the hinge loss by embedding a differentiable function to avoid discontinuities. An example of such a loss function is Huber's classification loss [19].

**Definition 2.3.** *For a binary classifier* $h_{\mathbf{w}} : \mathcal{X} \to \{-1, +1\}$ *of the form* $h_{\mathbf{w}}(x) = \operatorname{sign} g_{\mathbf{w}}(x)$ *with* $g_{\mathbf{w}}(x) = \mathbf{w}^{\mathsf{T}}\boldsymbol{\phi}(x)$*, weight vector* $\mathbf{w} \in \mathcal{W}$*, and any fixed smoothness parameter* $\delta > 0$*, the* Huber loss *is given by*

$$\ell^{\mathrm{H}}(g_{\mathbf{w}}(x), y) := \begin{cases} 0 & \text{if} \quad yg_{\mathbf{w}}(x) > 1 + \delta \\ \frac{1}{4\delta}(1 + \delta - yg_{\mathbf{w}}(x))^2 & \text{if} \quad |1 - yg_{\mathbf{w}}(x)| \leq \delta \\ 1 - yg_{\mathbf{w}}(x) & \text{if} \quad yg_{\mathbf{w}}(x) < 1 - \delta \end{cases} .$$

The Huber loss combines the relative robustness of the hinge loss against outliers and the continuous differentiability of a second-order polynomial.

**Proposition 2.2.** *For any fixed* $(x, y) \in \mathcal{X} \times \mathcal{Y}$*, the Huber loss* $\ell^{\mathrm{H}}(g_{\mathbf{w}}(x), y)$ *specified in Definition 2.3, is convex and once-continuously differentiable (but not twice-continuously differentiable) with respect to* $g_{\mathbf{w}}(x)$ *and, consequently, in* $\mathbf{w} \in \mathcal{W}$*.*

According to the proposition, the Huber loss is only once-continuously differentiable, which might be insufficient in some settings. Therefore, we propose a twice-continuously differentiable loss function which, similar to the Huber loss, linearly penalizes serious misclassifications.

**Definition 2.4.** *For a binary classifier* $h_{\mathbf{w}} : \mathcal{X} \to \{-1, +1\}$ *of the form* $h_{\mathbf{w}}(x) = \operatorname{sign} g_{\mathbf{w}}(x)$ *with* $g_{\mathbf{w}}(x) = \mathbf{w}^{\mathsf{T}}\boldsymbol{\phi}(x)$*, weight vector* $\mathbf{w} \in \mathcal{W}$*, and any fixed smoothness parameter* $\delta > 0$*, the* trigonometric loss *is given by*

$$\ell^{\mathrm{t}}(g_{\mathbf{w}}(x), y) := \begin{cases} 0 & \text{if} \quad yg_{\mathbf{w}}(x) > \delta \\ \frac{1}{2}(\delta - yg_{\mathbf{w}}(x)) - \frac{\delta}{\pi}\cos\left(\frac{\pi}{2\delta} yg_{\mathbf{w}}(x)\right) & \text{if} \quad |yg_{\mathbf{w}}(x)| \leq \delta \\ -yg_{\mathbf{w}}(x) & \text{if} \quad yg_{\mathbf{w}}(x) < -\delta \end{cases}.$$

Analogous to the Huber loss, where a polynomial is embedded into the hinge loss, the trigonometric loss combines the perceptron loss $\ell^{\mathrm{p}}(g_{\mathbf{w}}(x), y) = \max(0, -yg_{\mathbf{w}}(x))$ with a trigonometric function. This trigonometrical embedding yields a twice-continuously differentiable function.

**Proposition 2.3.** *For any fixed* $(x, y) \in \mathcal{X} \times \mathcal{Y}$*, the trigonometric loss* $\ell^{\mathrm{t}}(g_{\mathbf{w}}(x), y)$ *specified in Definition 2.4 is convex and twice-continuously differentiable with respect to* $g_{\mathbf{w}}(x)$ *and, consequently, in* $\mathbf{w} \in \mathcal{W}$*.*

If not stated otherwise, we set $\delta := 1$ for the trigonometric loss. Another convex loss function is the exponential loss, which is the underlying loss of boosting methods [33].

**Definition 2.5.** *For a binary classifier* $h_{\mathbf{w}} : \mathcal{X} \to \{-1, +1\}$ *of the form* $h_{\mathbf{w}}(x) = \operatorname{sign} g_{\mathbf{w}}(x)$ *with* $g_{\mathbf{w}}(x) = \mathbf{w}^{\mathsf{T}}\boldsymbol{\phi}(x)$ *and weight vector* $\mathbf{w} \in \mathcal{W}$*, the* exponential loss *is given by*

$$\ell^{\mathrm{e}}(g_{\mathbf{w}}(x), y) := \exp(-yg_{\mathbf{w}}(x)).$$

The exponential loss is a smooth function, that is, an infinitely-differentiable function. But similar as for the squared hinge loss, the exponential loss is highly sensitive to outliers. A more robust smooth loss function is the logistic loss which is employed in *logistic regression*.

**Definition 2.6.** *For a binary classifier* $h_{\mathbf{w}} : \mathcal{X} \to \{-1, +1\}$ *of the form* $h_{\mathbf{w}}(x) = \operatorname{sign} g_{\mathbf{w}}(x)$ *with* $g_{\mathbf{w}}(x) = \mathbf{w}^{\mathsf{T}}\boldsymbol{\phi}(x)$ *and corresponding weight vector* $\mathbf{w} \in \mathcal{W}$*, the* logistic loss *is given by*

$$\ell^{\mathrm{l}}(g_{\mathbf{w}}(x), y) := \log\left(1 + \exp(-yg_{\mathbf{w}}(x))\right).$$

**Proposition 2.4.** *For any fixed* $(x, y) \in \mathcal{X} \times \mathcal{Y}$*, the logistic loss* $\ell^{\mathrm{l}}(g_{\mathbf{w}}(x), y)$ *specified in Definition 2.6 is strictly convex and infinitely-continuously differentiable with respect to* $g_{\mathbf{w}}(x)$ *and, consequently, in* $\mathbf{w} \in \mathcal{W}$*.*

In contrast to all previously discussed loss functions, the logistic loss has a probabilistic interpretation, too. Let us assume, that the partial likelihood of decision function $f_{\mathbf{w}}(x, y) = yg_{\mathbf{w}}(x)$ with weight vector $\mathbf{w}$ has the form of the logistic function, that is,

$$p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{W}=\mathbf{w}}(+1) := \frac{1}{1 + \exp(-g_{\mathbf{w}}(x))},$$

so that

$$p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{W}=\mathbf{w}}(-1) = 1 - p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{W}=\mathbf{w}}(+1) = \frac{\exp(-g_{\mathbf{w}}(x))}{1+\exp(-g_{\mathbf{w}}(x))} = \frac{1}{1+\exp(g_{\mathbf{w}}(x))},$$

and consequently,

$$p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{W}=\mathbf{w}}(y) = \frac{1}{1+\exp(-yg_{\mathbf{w}}(x))}.$$

Equation 2.17 specifies the relation between the model likelihood and the loss function which together with the above expression yields

$$\exp(-\ell(g_{\mathbf{w}}(x), y)) = \frac{1}{1+\exp(-yg_{\mathbf{w}}(x))}.$$

Solving this equation for mapping $\ell(g_{\mathbf{w}}(x), y)$ gives the logistic loss as stated in Definition 2.6. Since maximizing the a-posteriori probability is equivalent to minimizing the regularized empirical error, the obtained models equal as well. An appealing consequence of this observation is that the decision values $f_{\mathbf{w}}(x, y)$ can be directly used to compute the conditional probability $\frac{1}{1+\exp(-f_{\mathbf{w}}(x,y))}$.

## 2.5 Regularization

One motivation for regularization is to transfer an *ill*-posed optimization problem into a feasible one, that is, to reformulate the problem so that the objective is continuous in its arguments and a solution exists. Recall, that we aim at minimizing the empirical estimate $\theta[f_{\mathbf{w}}, p_D]$ (*cf.* Equation 2.7) of the generalization error with respect to $f_{\mathbf{w}}$. The continuity condition of this function can be ensured by choosing a continuous loss function. In addition, we assume that the loss function and, consequently, the empirical error is convex in $\mathbf{w}$.

We now aim at restricting the set of possible solutions to a compact set, since then at least one solution is known to exist. An obvious way to ensure this condition is to restrain the set of feasible weight vectors, *e.g.,* by defining $\mathcal{W} := \{\mathbf{w} \in \mathbb{R}^m \mid \Omega(\mathbf{w}) \leq \rho\}$ for some convex regularization function $\Omega : \mathbb{R}^m \to \mathbb{R}$ and a fixed regularization parameter $\rho \in \mathbb{R}$ with $-\infty < \inf_{\mathbf{w}} \Omega(\mathbf{w}) < \rho < \infty$. However, as this establishes a constrained optimization problem, a more common approach is to implicitly restrict the feasible set. This is obtained by augmenting the objective by a regularization term, so that it becomes uniformly strongly convex, that is, $\hat{\theta}(\mathbf{w}, D) := \theta[f_{\mathbf{w}}, p_D] + \rho\Omega(\mathbf{w})$ (*cf.* Equation 2.25). This results in an unconstrained optimization problem with $\mathcal{W} := \mathbb{R}^m$. As the first summand $\theta[f_{\mathbf{w}}, p_D]$ is convex, a sufficient condition for $\hat{\theta}(\mathbf{w}, D)$ to be strongly convex is to employ a strongly convex regularizer $\Omega(\mathbf{w})$, controlled by some fixed $\rho \in \mathbb{R}$ with $0 < \rho < \infty$.

The most commonly used method to regularize $\mathbf{w}$ or $f_{\mathbf{w}}$, respectively, is Tikhonov regularization which implicitly restricts the $L^2$-norm of the gradient of function $f_{\mathbf{w}}$ with respect to $\boldsymbol{\phi}(x)$,

$$\|\nabla_{\boldsymbol{\phi}(x)} f_{\mathbf{w}}\|_2^2 := \int_{\mathcal{X}} \|\nabla_{\boldsymbol{\phi}(x)} f_{\mathbf{w}}(x, y)\|_2^2 \, \mathrm{d}x.$$

The Tikhonov regularizer favors smooth functions whereas complex, heavily fluctuating functions are strongly penalized. Since decision function $f_{\mathbf{w}}(x, y)$ is linear in $\boldsymbol{\phi}(x)$, the $L^2$-norm of $\nabla_{\boldsymbol{\phi}(x)} f_{\mathbf{w}}$ corresponds to the $l^2$-norm of weight vector $\mathbf{w}$. Hence, we set

$$\Omega(\mathbf{w}) := \frac{1}{2}\|\mathbf{w}\|_2^2. \tag{2.26}$$

A more detailed explanation on Tikhonov regularization in the context of machine learning can be found, for instance, in [51] and in Chapter 5.5.5 of [11].

Note, that the regularizer in (2.26) has a probabilistic meaning. As discussed before, in the Bayesian framework the regularizer corresponds to a negative log-prior on $\mathbf{w} \in \mathbb{R}^m$. If we choose a Gaussian prior centered at zero and the covariance matrix set to the identity matrix,

$$\Omega(\mathbf{w}) := -\log \mathcal{N}\left(\mathbf{w} \,|\, \mathbf{0}, \mathbf{I}_m\right),$$

then the regularizer resolves to

$$\Omega(\mathbf{w}) = \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w} + \frac{m}{2}\log 2\pi,$$

which equals the regularizer in Equation 2.26 up to a constant term.

We follow the approach of implicitly restricting the feasible set of weight vectors and, if not otherwise stated, restrain ourselves to regularizers of the form in (2.26). However, for most of the presented methods, we only require the regularizer to be uniformly strongly convex in $\mathbf{w} \in \mathcal{W}$.

## 2.6 Feature Representation and Kernels

So far, we assumed the knowledge of some predefined numeric representation $\boldsymbol{\phi}(x) := [\phi_1(x), \ldots, \phi_m(x)]^\mathsf{T} \in \mathbb{R}^m$ of objects $x \in \mathcal{X}$. However, in many applications such representations might be hard to identify. For instance, objects such as text documents, images, biological molecules, or audio sequences can be hardly mapped into a vector space without running the risk of loosing relevant information. Moreover, constructing such mappings generally requires much domain knowledge and engineering effort. In this section we discuss strategies to overcome these difficulties by the use of kernel functions, *i.e.,* the inner product in some implicitly defined feature space (see, for instance, Part III of [64] for further reading).

Consider the decision function $f_{\mathbf{w}}$ for binary classification where $f_{\mathbf{w}}(x, y) := y\mathbf{w}^\mathsf{T}\boldsymbol{\phi}(x)$. The corresponding hyperplane $G_{\mathbf{w}} := \left\{\boldsymbol{\phi}(x) \in \mathbb{R}^m \mid \mathbf{w}^\mathsf{T}\boldsymbol{\phi}(x) = 0\right\}$ is assumed to separate at least one training object $\boldsymbol{\phi}(x_i)$ form the others and, consequently, lives in the subspace spanned by all mapped training instances $\boldsymbol{\phi}(x_i)$ for $i = 1, ..., n$. Formally, the *representer theorem* (see, for instance, [63]) justifies this consideration. It states, that the normal vector $\mathbf{w}$ of any separating hyperplane can be expressed as a linear combination of the separated

points, that is, there exists $\boldsymbol{\omega} \in \mathbb{R}^n$ so that

$$\mathbf{w} = \sum_{i=1}^{n} \omega_i \boldsymbol{\phi}(x_i). \tag{2.27}$$

The $n$-dimensional vector $\boldsymbol{\omega}$ of weights $\omega_i$, which are not necessarily uniquely defined by (2.27), is called a *dual* weight vector of $\mathbf{w} \in \mathbb{R}^m$. This formulation enables us to substitute $\mathbf{w}$ so that

$$g_{\mathbf{w}}(x) = \mathbf{w}^\mathsf{T} \boldsymbol{\phi}(x) = \sum_{i=1}^{n} \omega_i \boldsymbol{\phi}(x_i)^\mathsf{T} \boldsymbol{\phi}(x) = \sum_{i=1}^{n} \omega_i k(x_i, x) = g_{\boldsymbol{\omega}}(x), \tag{2.28}$$

where now $f_{\boldsymbol{\omega}}(x, y) := y g_{\boldsymbol{\omega}}(x)$ is parameterized by $\boldsymbol{\omega} \in \mathbb{R}^n$ and training instances $x_i$, $i = 1, ..., n$. Mapping $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with $k(x, x') = \boldsymbol{\phi}(x)^\mathsf{T} \boldsymbol{\phi}(x')$ is referred as *kernel function* which is a similarity measure between any two objects $x, x' \in \mathcal{X}$. Likewise, we can define the $l^2$-norm regularizer as a function of the dual weight vector $\boldsymbol{\omega}$,

$$\Omega(\mathbf{w}) = \frac{1}{2} \mathbf{w}^\mathsf{T} \mathbf{w} = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \omega_i \omega_j k(x_i, x_j) = \frac{1}{2} \boldsymbol{\omega}^\mathsf{T} \mathbf{K} \boldsymbol{\omega} = \Omega(\boldsymbol{\omega}), \tag{2.29}$$

where $\mathbf{K} \in \mathbb{R}^{n \times n}$ is the (positive semi-definite) *kernel matrix* with $K_{ij} := k(x_i, x_j)$, $\forall i, j \in \{1, ..., n\}$.

Hence, instead of minimizing the regularized empirical error over the primal weight vector $\mathbf{w} \in \mathbb{R}^m$ for $n$ given mapped training object-class pairs $(\boldsymbol{\phi}(x_i), y_i)$, we can directly employ a kernel function without the need of an explicitly defined feature mapping. However, for any valid kernel $k$, there still has to exist a feature mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{H}$. This function is assumed to map objects from the input space $\mathcal{X}$ into a potentially infinite-dimensional vector space (*i.e.,* a Hilbert space) $\mathcal{H}$, so that the kernel can be stated as an inner product in this space, that is, there exists $\boldsymbol{\phi}$ so that

$$k(x, x') = \left\langle x, x' \right\rangle_{\mathcal{H}} = \left\langle \boldsymbol{\phi}(x), \boldsymbol{\phi}(x') \right\rangle.$$

According to Mercer's theorem (*cf.* for instance, Theorem 2.10 in [64]), this condition is met if $k$ is a positive semi-definite function, *i.e.,*

$$\int_{\mathcal{X}} \int_{\mathcal{X}} k(x, x') u(x) u(x') \, \mathrm{d}x \, \mathrm{d}x' \geq 0$$

for all square-integrable functions $u : \mathcal{X} \to \mathbb{R}$. For a finite sample of $n$ instances $x_i$, this requirement reduces to

$$\sum_{i=1}^{n} \sum_{j=1}^{n} k(x_i, x_j) u_i u_j = \mathbf{u}^\mathsf{T} \mathbf{K} \mathbf{u} \geq 0$$

for all vectors $\mathbf{u} \in \mathbb{R}^n$, that is, kernel matrix $\mathbf{K}$ must be positive semi-definite. That is, each mapping $k$ which, for any given finite sample of objects, yields a positive semi-definite kernel matrix, is a valid kernel function referred as *Mercer kernel*.

An appealing consequence is that we can state mapped instances $\boldsymbol{\phi}(x_i) \in \mathbb{R}^n$ explicitly solely based on a given symmetric positive semi-definite similarity matrix (or kernel matrix) $\mathbf{K} \in \mathbb{R}^{n \times n}$ of objects $x_i \in \mathcal{X}$, $i = 1, \ldots, n$. Consider therefore the eigen-decomposition of this matrix,

$$\mathbf{K} = \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{V}^\mathsf{T},$$

where $\boldsymbol{V}$ is the column matrix of eigenvectors of kernel matrix $\mathbf{K}$ and $\boldsymbol{\Lambda}$ is a diagonal matrix whose diagonal elements are the corresponding non-negative eigenvalues. Let us define

$$\boldsymbol{\phi}_{\text{PCA}} : x \mapsto \boldsymbol{\Lambda}^{\frac{1}{2}^+} \boldsymbol{V}^\mathsf{T} \left[k(x_1, x), \ldots, k(x_n, x)\right]^\mathsf{T}, \tag{2.30}$$

where $\boldsymbol{\Lambda}^{\frac{1}{2}^+}$ denotes the Moore-Penrose pseudo-inverse of the square root of $\boldsymbol{\Lambda}$ with $\boldsymbol{\Lambda} = \boldsymbol{\Lambda}^{\frac{1}{2}} \boldsymbol{\Lambda}^{\frac{1}{2}}$. This mapping establishes an $n$-dimensional vectorial representation of objects $x_i$, that can be constructed for every valid kernel function and finite sample of objects. It is referred as *kernel PCA mapping* or Mercer mapping.

**Remark 2.5.** Notice, that for any positive semi-definite kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and fixed training instances $x_1, \ldots, x_n \in \mathcal{X}$, the kernel PCA mapping is a uniquely defined real function with $\boldsymbol{\phi}_{\text{PCA}} : \mathcal{X} \to \mathbb{R}^n$ so that $k(x_i, x_j) = \boldsymbol{\phi}_{\text{PCA}}(x_i)^\mathsf{T} \boldsymbol{\phi}_{\text{PCA}}(x_j)$ for any $i, j \in \{1, \ldots, n\}$: We first show that $\boldsymbol{\phi}_{\text{PCA}}$ is a real mapping from the input space $\mathcal{X}$ to the Euclidean space $\mathbb{R}^n$. As $x \mapsto [k(x_1, x), \ldots, k(x_n, x)]^\mathsf{T}$ is a real vector-valued function and $\boldsymbol{V}$ is a real $n \times n$ matrix, it remains to show that the pseudo-inverse of $\boldsymbol{\Lambda}^{\frac{1}{2}}$ is real as well. Since the kernel function is positive semi-definite, all eigenvalues $\lambda_i$ of $\mathbf{K}$ are non-negative, and hence, $\boldsymbol{\Lambda}^{\frac{1}{2}}$ is a diagonal matrix with real diagonal entries $\sqrt{\lambda_i}$ for $i = 1, \ldots, n$. The pseudo-inverse of this matrix is the uniquely defined diagonal matrix $\boldsymbol{\Lambda}^{\frac{1}{2}^+}$ with real non-negative diagonal entries $\frac{1}{\sqrt{\lambda_i}}$ if $\lambda_i > 0$ and zero otherwise. This proves the first claim.

The kernel PCA mapping also satisfies $k(x_i, x_j) = \boldsymbol{\phi}_{\text{PCA}}(x_i)^\mathsf{T} \boldsymbol{\phi}_{\text{PCA}}(x_j)$ for any pair of training instances $x_i$ and $x_j$, since for all $i = 1, \ldots, n$,

$$\begin{aligned}
\boldsymbol{\phi}_{PCA}(x_i) &= \boldsymbol{\Lambda}^{\frac{1}{2}^+} \boldsymbol{V}^\mathsf{T} \left[k(x_1, x_i), \ldots, k(x_n, x_i)\right]^\mathsf{T} \\
&= \boldsymbol{\Lambda}^{\frac{1}{2}^+} \boldsymbol{V}^\mathsf{T} \mathbf{K} \mathbf{e}_i \\
&= \boldsymbol{\Lambda}^{\frac{1}{2}^+} \boldsymbol{V}^\mathsf{T} \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{V}^\mathsf{T} \mathbf{e}_i \\
&= \boldsymbol{\Lambda}^{\frac{1}{2}^+} \boldsymbol{\Lambda} \boldsymbol{V}^\mathsf{T} \mathbf{e}_i
\end{aligned}$$

where $\mathbf{e}_i \in \{0, 1\}^n$ is the $i$-th unit vector. Hence,

$$\begin{aligned}
\boldsymbol{\phi}_{PCA}(x_i)^\mathsf{T} \boldsymbol{\phi}_{PCA}(x_j) &= \mathbf{e}_i^\mathsf{T} \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{\Lambda}^{\frac{1}{2}^+} \boldsymbol{\Lambda}^{\frac{1}{2}^+} \boldsymbol{\Lambda} \boldsymbol{V}^\mathsf{T} \mathbf{e}_j \\
&= \mathbf{e}_i^\mathsf{T} \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{\Lambda}^+ \boldsymbol{\Lambda} \boldsymbol{V}^\mathsf{T} \mathbf{e}_j \\
&= \mathbf{e}_i^\mathsf{T} \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{V}^\mathsf{T} \mathbf{e}_j \\
&= \mathbf{e}_i^\mathsf{T} \mathbf{K} \mathbf{e}_j = \mathbf{K}_{ij} = k(x_i, x_j)
\end{aligned}$$

which proves the second claim. ◇

To classify a new instance $x \in \mathcal{X}$ for a given parameter vector $\mathbf{w}$, we may first map $x$ into the PCA mapping-induced feature space and apply the linear classifier $h_{\mathbf{w}}(x) = \operatorname{sign} g_{\mathbf{w}}(x)$ with $g_{\mathbf{w}}(x) = \mathbf{w}^\mathsf{T} \boldsymbol{\phi}_{\mathrm{PCA}}(x)$. Alternatively, we can derive a dual representation of $\mathbf{w}$ so that $\mathbf{w} = \sum_{i=1}^n \omega_i \boldsymbol{\phi}_{\mathrm{PCA}}(x_i)$, and consequently $g_{\mathbf{w}}(x) = g_{\boldsymbol{\omega}}(x) = \sum_{i=1}^n \omega_i k(x_i, x)$, where $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_n]^\mathsf{T}$ is a not necessarily uniquely defined dual weight vector of $\mathbf{w}$. Therefore, we have to identify a solution $\boldsymbol{\omega}$ of the linear system

$$\mathbf{w} = \boldsymbol{\Lambda}^{\frac{1}{2}^+} \boldsymbol{V}^\mathsf{T} \mathbf{K} \boldsymbol{\omega}. \tag{2.31}$$

A direct calculation shows that

$$\boldsymbol{\omega} := \boldsymbol{V} \boldsymbol{\Lambda}^{\frac{1}{2}^+} \mathbf{w} \tag{2.32}$$

is a solution of (2.31) provided that either all elements $\lambda_i$ of the diagonal matrix $\boldsymbol{\Lambda}$ are positive or that $\lambda_i = 0$ implies that the same component of the vector $\mathbf{w}$ is also equal to zero (in which case the solution is non-unique). In fact, inserting (2.32) in (2.31) then gives

$$\boldsymbol{\Lambda}^{\frac{1}{2}^+} \boldsymbol{V}^\mathsf{T} \mathbf{K} \boldsymbol{\omega} = \boldsymbol{\Lambda}^{\frac{1}{2}^+} \boldsymbol{V}^\mathsf{T} \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{V}^\mathsf{T} \boldsymbol{V} \boldsymbol{\Lambda}^{\frac{1}{2}^+} \mathbf{w} = \boldsymbol{\Lambda}^{\frac{1}{2}^+} \boldsymbol{\Lambda}^{\frac{1}{2}} \boldsymbol{\Lambda}^{\frac{1}{2}} \boldsymbol{\Lambda}^{\frac{1}{2}^+} \mathbf{w} = \mathbf{w}.$$

The advantage of this second approach is that classifying a new instances $x \in \mathcal{X}$ requires the computation of the scalar product $\sum_{i=1}^n \omega_i k(x_i, x)$ rather than a matrix multiplication when mapping $x$ into the kernel PCA mapping-induced feature space (*cf.* Equation 2.30).

In conclusion, even if the model components cannot be expressed in terms of inner products such as in (2.28) and (2.29), one can still apply kernel functions by constructing an explicit representation such as defined by the kernel PCA mapping. Likewise, if we are not given an explicit kernel function but a symmetric positive semi-definite similarity matrix, we can still construct a vectorial representation of the data to learn a predictive model.

## 2.7 Parameter Estimation

In the previous sections we have discussed the required components to determine the regularized empirical error in (2.25) of a particular weight vector $\mathbf{w} \in \mathbb{R}^m$ or a dual weight vector $\boldsymbol{\omega} \in \mathbb{R}^n$, respectively, for a given data set $D = \{(x_i, y_i)\}_{i=1}^n$. We now turn to the problem of estimating an optimal parameter vector which minimizes this error,

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w} \in \boldsymbol{W}} \hat{\theta}(\mathbf{w}, D) = \operatorname*{argmin}_{\mathbf{w} \in \boldsymbol{W}} \frac{1}{n} \sum_{i=1}^n c(x_i, y_i) \ell(g_{\mathbf{w}}(x_i), y_i) + \rho \Omega(\mathbf{w}).$$

This process is referred to as the *training* phase.

As discussed before, the regularized empirical error is generally designed so that the resulting minimization problem is convex and unrestricted, *i.e.,* $\mathcal{W} = \mathbb{R}^m$. There exists a large variety of methods to solve such optimization problems, where most of them are descent methods such as the method of steepest descent, conjugate gradient, (quasi) Newton, cutting plane, and interior point (*cf.* for instance, [13]). For one of the most recent distributed optimization methods see [12].

In this section, we exemplarily present the stochastic gradient method also referred as *Robbins-Monro* algorithm [68]. Even if this method is a first-order method which generally attains a low convergence rate, it is very common for regularized empirical error minimization as its memory requirement is in general much smaller than that of a second-order or full gradient-based approach. In addition, stochastic gradient methods can be easily parallelized (see, for instance, [77]) which makes them especially applicable to learn from very large amounts of data.

The principal idea of iterative first-order methods is to construct a sequence of parameter vectors $\mathbf{w}^{(0)}, \mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(k)}$ which converges to a vector $\mathbf{w}^*$ satisfying the first-order optimality condition. For unrestricted optimization problems, this condition says that the gradient of $\hat{\theta}(\mathbf{w}, D)$ attains zero at $\mathbf{w}^*$,

$$\left.\frac{\partial \hat{\theta}(\mathbf{w}, D)}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}^*} = \frac{1}{n}\sum_{i=1}^{n}\left(c(x_i, y_i)\ell'(g_{\mathbf{w}^*}(x_i), y_i)\boldsymbol{\phi}(x_i) + \rho\Omega'(\mathbf{w}^*)\right) = \mathbf{0}, \tag{2.33}$$

where $\ell'(z, y)$ denotes the first derivative of $\ell(z, y)$ with respect to $z$ and $\Omega'(\mathbf{w})$ denotes the gradient of $\Omega(\mathbf{w})$. Finding the root of the continuous function $\frac{\partial \hat{\theta}(\mathbf{w}, D)}{\partial \mathbf{w}}$, which is defined as expectation on an underlying probability space with the empirical density $p_D(x, y)$ (*cf.* Equation 2.3),

$$\mathrm{E}_D\left[c(x, y)\ell'(g_{\mathbf{w}^*}(x), y)\boldsymbol{\phi}(x) + \rho\Omega'(\mathbf{w}^*)\right],$$

establishes a stochastic optimization problem. Corollary 2.6, which immediately follows from the *Robbins-Monro* theorem (see, *e.g.,* Theorem 1.4.26 in [28]), states a recursively defined sequence which converges almost surely to a solution of the problem in (2.33).

**Corollary 2.6.** *Let the regularized empirical error $\hat{\theta}(\mathbf{w}, D)$ be defined as in Equation 2.25 for given training data $D = \{(x_i, y_i)\}_{i=1}^n$. Suppose $\hat{\theta}(\mathbf{w}, D)$ is uniformly strongly convex and continuously differentiable in $\mathbf{w}$ and suppose that inequality*

$$\left\|c(x_i, y_i)\ell'(g_{\mathbf{w}}(x_i), y_i)\boldsymbol{\phi}(x_i) + \rho\Omega'(\mathbf{w})\right\|_2^2 \leq \varsigma\left(1 + \|\mathbf{w}\|_2^2\right) \tag{2.34}$$

*holds for all $i \in \{1, \ldots, n\}$, $\mathbf{w} \in \mathcal{W}$, and some constant $\varsigma \in \mathbb{R}$. Then, for any $\alpha > 0$, $\mathbf{w}^{(0)} \in \mathcal{W}$, and $(x^{(k)}, y^{(k)})$ randomly drawn from $D$, sequence*

$$\mathbf{w}^{(k+1)} := \mathbf{w}^{(k)} - \frac{\alpha}{k}\left(c(x^{(k)}, y^{(k)})\ell'(g_{\mathbf{w}^{(k)}}(x^{(k)}), y^{(k)})\boldsymbol{\phi}(x^{(k)}) + \rho\Omega'(\mathbf{w}^{(k)})\right) \tag{2.35}$$

*converges almost surely to the minimizer $\mathbf{w}^*$ of $\hat{\theta}(\mathbf{w}, D)$.*

The condition in (2.34) says, that for any fixed $(x, y) \in \mathcal{X} \times \mathcal{Y}$, neither $\ell'(g_{\mathbf{w}^*}(x), y)$ nor $\Omega'(\mathbf{w})$ grow faster than a linear function of $\|\mathbf{w}\|_2$. Notice, that except for the exponential loss, all discussed loss functions (*cf.* Section 2.4) and, for instance, the quadratic regularizer in (2.26), satisfy this requirement.

According to Robbins and Siegmund [58], Corollary 2.6 can also be extended to the case of constrained optimization where $\mathcal{W} \subset \mathbb{R}^m$ is some non-empty convex set. This is, for

instance, attained by replacing $\mathbf{w}^{(k+1)}$ in (2.35) by its $L^2$-projection $\Pi_{\mathcal{W}}(\mathbf{w}^{(k+1)})$ into the feasible set $\mathcal{W}$, where

$$\Pi_A(\mathbf{a}) := \operatorname*{argmin}_{\mathbf{a}' \in A} \left\|\mathbf{a} - \mathbf{a}'\right\|_2^2. \tag{2.36}$$

In the special case, where $\mathcal{W} = \{\mathbf{w} \in \mathbb{R}^m \mid \|\mathbf{w}\|_2 \leq r\}$ is the closed $\ell^2$-ball of radius $r > 0$ and $\mathbf{w}^{(k+1)} \notin \mathcal{W}$, this projection reduces to rescaling vector $\mathbf{w}^{(k+1)}$ to length $r$.

Finally, we state the pseudo-code of the induced algorithm for regularized *empirical risk minimization* (ERM). Note that, for instance, the perceptron [61] and Pegasos [65] are specific instances of this algorithm.

**Algorithm 1** SGD-ERM: Stochastic Gradient Descent for Empirical Risk Minimization

**Require:** Training data $D = \{(x_i, y_i)\}_{i=1}^n$, regularization parameter $\rho > 0$, and learning rate $\alpha > 0$.

1: Select initial $\mathbf{w}^{(0)} \in \mathcal{W}$ and set $k := 0$.
2: **repeat**
3: Draw $(x, y)$ randomly from $D$.
4: Set $\mathbf{d}^{(k)} := -c(x, y)\ell'(g_{\mathbf{w}^{(k)}}(x), y)\boldsymbol{\phi}(x) - \rho\Omega'(\mathbf{w}^{(k)})$.
5: Set $\mathbf{w}^{(k+1)} := \Pi_{\mathcal{W}}\left(\mathbf{w}^{(k)} + \frac{\alpha}{k}\mathbf{d}^{(k)}\right)$.
6: Set $k := k + 1$.
7: **until** $\|\mathbf{w}^{(k)} - \mathbf{w}^{(k-1)}\|_2^2 \leq \epsilon$.

# 3 The Prediction Game

In the previous chapter we presented the principal ideas to build regular predictive models based on a given sample of object-target pairs. The goal was to find a mapping $h_{\mathbf{w}}$ which generalizes to new unseen data $\dot{D} = \{(\dot{x}_j, \dot{y}_j)\}_{j=1}^{l}$ so that it correctly predicts the target $\dot{y}_j \in \mathcal{Y}$ of the test objects $\dot{x}_j \in \mathcal{X}$. So far, we have assumed that the training data and test data are drawn *i.i.d.* from the same underlying probability space, that is, training sample $D$ and test sample $\dot{D}$ are governed by the same distribution.

However, in adversarial prediction problems, such as the introductive examples in Section 1.1, this assumption is often violated. Here, the object-target pairs of the training data are governed by an unknown distribution $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}$, where $\gamma$ is the corresponding data generation model, for instance, the parameter values of a spam template. The test data distribution $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$, induced by data generation model $\dot{\gamma}$, may significantly differ from the training data distribution, since $\gamma$ and $\dot{\gamma}$ are not assumed to equal anymore. This discrepancy between the data generation models is assumed to be, at least partially, caused in response to the predictive model. Such situations where the data generation process is to some extend influenced by the outcome of the learning process involve a conflict of interest between the *learner*, who builds the prediction model $h_{\mathbf{w}}$, and the *data generator*, who controls $\dot{\gamma}$.

In this chapter we present the conceptual extension of the classical learning setting to the problem of learning adversary-aware predictive models. To this end, Section 3.1 starts with a short introduction to game theory which provides techniques to study situations of potential conflict between two or more individuals called *games*. We close this section with a discussion on well known game solution concepts and continue with inspecting the players' interaction in the adversarial learning setting in Section 3.2. In Section 3.3 we formalize this interaction as a game theoretical model and discuss its components, which are the players of the game, their possible actions, and their cost functions. Finally we introduce three classes of prediction games in Section 3.4 which we will study in detail in the subsequent chapters.

## 3.1 An Introduction to Game Theory

In this section we give a brief overview on game theoretic terms and models used throughout the thesis, and introduce the well known solution concepts for static and dynamic non-cooperative games.

### 3.1.1 Basic Terms and Definitions

The participants of games, which could be individuals or groups of individuals, are referred as *players*. Each player's plan of action is called their *strategy* and the set of actions available to a player is referred as their *action set*. Formally, a strategy is a probabilistic assignment of a particular action to the observed state of the game. If each player chooses their action deterministically, then this identical mapping is called a *pure strategy*, that is, the probability of choosing this single action is one whereas all other actions are played with zero probability. In this case, the terms strategy and action are often used interchangeably. The concept of *mixed strategy* is more general in the way that each player chooses a probability density over the whole space of possible actions. When playing a mixed strategy, the player's move is to randomly draw an action from the induced probability distribution.

The *outcome* of a game is one particular combination of actions and the implied *costs* (or payoffs) for the players. Each player is aware that this outcome is affected by their and the others chosen actions and that, consequently, their chosen action has an impact on the other players' decisions. In a *non-cooperative* game, the players are not allowed to communicate while making their decisions. Hence, the players have only limited information about the other players' strategies and they have to make an assumption on what the other players might do. To make reasonable conjectures, the players are assumed to act *rationally* in the sense of securing their highest possible payoff. Under this assumption, the combination of each player's best plan of action can be characterized as *equilibrium* strategies. If all players commit to one such combination, neither player has an incentive to change their plan of actions solely. Hence, these game equilibria are combinations of strategies that rational players are predicted to choose when they interact.

Games are characterized in terms of rules which describe how each player's behavior affects their and the other players' payoffs. Amongst others, the rules cover information on the participants of the game, their knowledge about each other, their possible actions, their payoffs, and the order in which they choose their actions. By these rules, the following classes of games can be distinguished.

#### Static and Dynamic Games

Games in which all players act simultaneously are referred as *static games*, Nash games, or single-stage games (for example, rock-paper-scissors). By contrast, games where players act in some predefined order and may make more than one move during the game are called *dynamic games*, sequential games, or multi-stage games (for example, poker). A dynamic game decouples into several stages where at each stage a subset of the players simultaneously decide for their action, in the simplest case, exactly one player per stage. If, in dynamic games, the move of each player can be observed by the followers, then the players have *perfect information*; otherwise it is a game of *imperfect information*. The special case of a two-player two-stage game of perfect information is called a *Stackelberg game*.

#### Constant-Sum Games

In case of two players having diametrically opposed interests, the costs of both players add to a constant for which reason such games are called *constant-sum games* or *zero-sum games*, respectively, if the costs sum up to zero. Zero-sum games are typically used to model worst-case scenarios where the gain of one player is the loss of the other. However, most games are not constant-sum games, as there may be mutually beneficial or mutually harmful outcomes for the players.

#### One-Shot and Repeated Games

Static as well as dynamic games can be played only once, referred as *one-shot games*, or they can be played repeatedly, called *repeated games*. In the latter case, the players aim at maximizing their payoffs over all finitely or infinitely many rounds of the game. Therefore, the players will have to take the impact of their action on the outcome of the current and future rounds into account. Note, that a repeated game is also a dynamic game but not necessarily vice versa. For instance, poker is a dynamic game where several players interact consecutively and each player may make several moves. However, poker is not a repeated game. Only when participating in a poker tournament one plays a repeated (dynamic) game.

#### Bayesian Games

The information of the players about each other (*e.g.,* the players' interests, their possible actions, etc.) are often *incomplete* which implies an additional element of risk. Typically, this uncertainty is characterized by a probability distribution over the outcome of the game where initial beliefs are factored in by priors. Such games are called *Bayesian games* because of the probabilistic analysis inherent in such games. Notice, that games of imperfect information can be transformed into dynamic Bayesian games by using the so called Harsanyi transformation (see, for instance, Section 2.4 in [57]).

#### Discrete and Continuous Games

Games in which players choose from a finite set of actions are called *discrete games*. In contrast, the concept of *continuous games* allows games to have more general action sets of possibly uncountably many actions. In the special case where all players' payoff functions are continuous, the game is called a *continuous kernel game*.

Games can be formalized in different ways. The two most common definitions are the *normal-form* or strategic-form representation and the *extensive-form* representation. The normal form of a game is often used to describe static games whereas the extensive form also allows to capture sequential interactions such as in dynamic games. As we study mainly static games in this thesis, we make use of the normal-form representation.

**Definition 3.1.** *A game in* normal form *consists of*

1. *A finite, totally ordered set $V$ of players $v \in V$, where $v < \bar{v}$ implies that player $v \in V$ acts before player $\bar{v} \in V$ and $v = \bar{v}$ says that both players act simultaneously;*
2. *A non-empty set $A_v$ of possible actions $a_v \in A_v$ for each player $v \in V$;*
3. *A cost function $\theta_v : \times_{\bar{v} \in V} A_{\bar{v}} \to \mathbb{R}$ for each $v \in V$ that specifies their cost at the outcome of the game.*

The definition given above differs to that used in literature insofar that we assumes an ordered set of players to capture Stackelberg games as well; for static games, this ordering amounts to an equality relation.

### 3.1.2 Solution Concepts

A feasible combination of strategies (or actions, when playing pure strategies) is called a *solution* or *strategy profile* of a game. For static games, the most commonly used solution concept is that of an *Nash equilibrium* (NE). An NE describes a steady state of the play in which each player has a correct expectation about the other players' behavior and acts rationally. If all player commit to such a combination of equilibrium strategies, no player gains a benefit by unilaterally selecting a different strategy. However, the concept of NE does not attempt to examine the process by which a steady state is reached for which reason some Nash equilibria may be more plausible then others.

**Definition 3.2.** *A* Nash equilibrium *in pure strategies of a static game with player set $V = \{1, \dots, k\}$, action sets $A_1, \dots, A_k$, and cost functions $\theta_1, \dots, \theta_k$ is a combination of actions $(a_1, \dots, a_k) \in A_1 \times \dots \times A_k$ with the property that for every player $v \in V$ we have*

$$\theta_v(a_1, \dots, a_v, \dots, a_k) \leq \theta_v(a_1, \dots, a'_v, \dots, a_k) \quad \forall\, a'_v \in A_v.$$

The concept of NE can also be defined in terms of the *rational reaction sets* $A_v^*$ of the players $v \in P$. These sets contain all actions that minimize the players' costs for a given combination of actions of all other players.

**Definition 3.3.** *Assume we are given a game with player set $V = \{1, \dots, k\}$, action sets $A_1, \dots, A_k$, and cost functions $\theta_1, \dots, \theta_k$. Then, the* rational reaction set *of player $v \in V$ for a given combination of actions $\mathbf{a}_{-v} := (a_1, \dots, a_{v-1}, a_{v+1}, \dots, a_k)$ is defined by*

$$A_v^*(\mathbf{a}_{-v}) := \left\{ a_v \in A_v \;\middle|\; \theta_v(a_1, \dots, a_v, \dots, a_k) \leq \theta_v(a_1, \dots, a'_v, \dots, a_k)\ \forall\, a'_v \in A_v \right\} \subseteq A_v.$$

By this definition, a combination of actions $(a_1, \dots, a_k) \in A_1 \times \dots \times A_k$ is an NE if, and only if, $a_v \in A_v^*(a_1, \dots, a_{v-1}, a_{v+1}, \dots, a_k)$ for all $v = 1, \dots, k$. Before presenting a visual interpretation of NE with the help of rational reaction sets, we shall discuss a refinement of this equilibrium to dynamic games called *subgame perfect equilibrium* (SPE). Similar to NE, a subgame perfect (Nash) equilibrium is a steady state of a dynamic game where

neither player has an interest in changing their strategy solely. However, the SPE takes the order at which the players make their decisions into account. Formally, a combination of strategies is an SPE, if this strategy profile is an NE in each stage of the game, *i.e.,* an NE in all subgames of the dynamic game. An SPE is commonly identified by backward induction where one first considers the final stage of the game and determines which actions the players of this stage are supposed to choose. All of these action combinations are Nash equilibria with respect to the players of that stage. This set establishes the joint action space of the players in the second to last stage. Then, we determine the set of all NEs with respect to the players of the last and the second to last stage which establishes the joint action set of the players in the third to last stage and so on.

In the following we will illustrate the concept of SPE in pure strategies for Stackelberg games, one of the simplest dynamic games. As mentioned before, a Stackelberg game is a two-stage game of perfect information where in the first stage player $v_1 = 1$, called the *leader*, decides for an action. In the second stage, player $v_2 = 2$, referred as the *follower*, observes the leader's decision and chooses their own action. Following the idea of backward induction, an SPE of a Stackelberg game, a *Stackelberg equilibrium*, can be determined as follows: We consider the last stage and specify the follower's rational reaction set

$$A_2^*(a_1) := \left\{ a_2 \in A_2 \;\middle|\; \theta_2(a_1, a_2) = \min_{a_2' \in A_2} \theta_2(a_1, a_2') \right\},$$

that is, for any fixed action $a_1 \in A_1$ of the leader, set $A_2^*(a_1)$ contains all actions which raise minimal costs to the follower. These actions are of course Nash optimal as there is only one player in the final stage. Considering the second to last stage (*i.e.,* the first stage) of the play, the rational reaction set $A_2^*(a_1)$ now becomes the follower's action set. Hence, an SPE of a Stackelberg game is an NE of the subgame with players $p_1$ and $p_2$ and action spaces $A_1$ and $\bigcup_{a_1' \in A_1} A_2^*(a_1')$, *i.e.,* a Stackelberg equilibrium is a pair of actions which satisfies

$$\begin{aligned} \theta_1(a_1, a_2) &= \min_{a_1' \in A_1} \theta_1(a_1', a_2) \\ \theta_2(a_1, a_2) &= \min_{a_2' \in A_2^*(a_1)} \theta_2(a_1, a_2') \end{aligned}$$

where the latter condition is satisfied for all $a_2 \in A_2^*(a_1)$ by definition. Therefore, an SPE of a Stackelberg game can be specified as follows.

**Definition 3.4.** *A* subgame perfect equilibrium *(SPE) in pure strategies of a Stackelberg game with leader $v_1 = 1$ and follower $v_2 = 2$, action sets $A_1, A_2$, and cost functions $\theta_1, \theta_2$ is a solution $(a_1, a_2) \in A_1 \times A_2$ of the minimization problem*

$$\begin{aligned} \min_{a_1 \in A_1, a_2 \in A_2} \quad & \theta_1(a_1, a_2) \\ \text{s.t.} \quad & \theta_2(a_1, a_2) = \min_{a_2' \in A_2} \theta_2(a_1, a_2') \end{aligned}$$

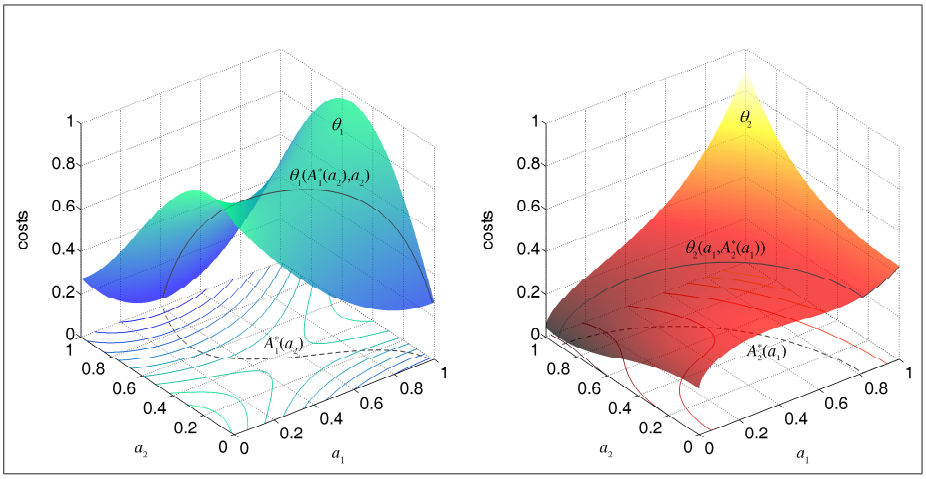


**Figure 3.1:** Exemplary objectives $\theta_1$ and $\theta_2$ of two players 1 (blue) and 2 (red) with one-dimensional action spaces $A_1 = A_2 = [0, 1]$. The dashed dark gray curves indicate the players' optimal response curves, the solid curves show the corresponding (minimal) costs.

To illustrate the differences between NE and SPE, let us consider a continuous kernel game with two players $V = \{1, 2\}$, action sets $A_1 = A_2 = [0, 1]$, and continuous cost functions $\theta_1$ and $\theta_2$ visualized in Figure 3.1. For the given cost functions, each player's rational reaction set $A_v^*(\mathbf{a}_{-v})$ is a singleton for any fixed $\mathbf{a}_{-v}$ so that $\bigcup_{\mathbf{a}_{-v}} A_v^*(\mathbf{a}_{-v})$ reduces to a curve. Hence, a pair of actions $(a_1, a_2) \in A_1 \times A_2$ is an NE, if $a_1 = A_1^*(a_2)$ and $a_2 = A_2^*(a_1)$. This means that the reaction curves intersect at this point. Figure 3.2 (right) visualizes the reaction curves of both players and the intersection points, *i.e.,* the Nash equilibria $\text{NE}_1$, $\text{NE}_2$, and $\text{NE}_3$. If both players act simultaneously, these points are stable states of the game where both players have no incentive in solely deviating from their strategy.

To understand the concept of SPE, let us now assume that player 1 acts *before* player 2. No matter which action $a_1 \in A_1$ player 1 decides on, player 2 chooses an action out of their corresponding rational reaction set $A_2^*(a_1)$, which is a singleton in our example. Player 2's choice raises costs $\theta_1(a_1, A_2^*(a_1))$ to player 1. Player 1 is expected to minimize their costs with respect to $a_1$, that is, the action $a_1 \in A_1$ with minimal costs along player 2's reaction curve. Figure 3.2 shows the subgame perfect equilibrium $\text{SPE}_1$; in the right figure, $\text{SPE}_1$ is a point on player 2's reaction curve $A_2^*(a_1)$ which is tangential to the contour curves of player 1 (blue). Note that SPE as well as NE are not necessarily unique in general. If there are several Nash equilibria, these combinations of actions are generally not equally preferable (for instance, $\text{NE}_2$ imposes higher costs to both players than $\text{NE}_1$). In contrast, if there are several subgame perfect equilibria of a Stackelberg game, each of these points raise the same costs to the leader.

From Figure 3.2 (right) we can also conclude on the existence of NE. In our simple example of two players with continuous cost functions and compact and non-empty action

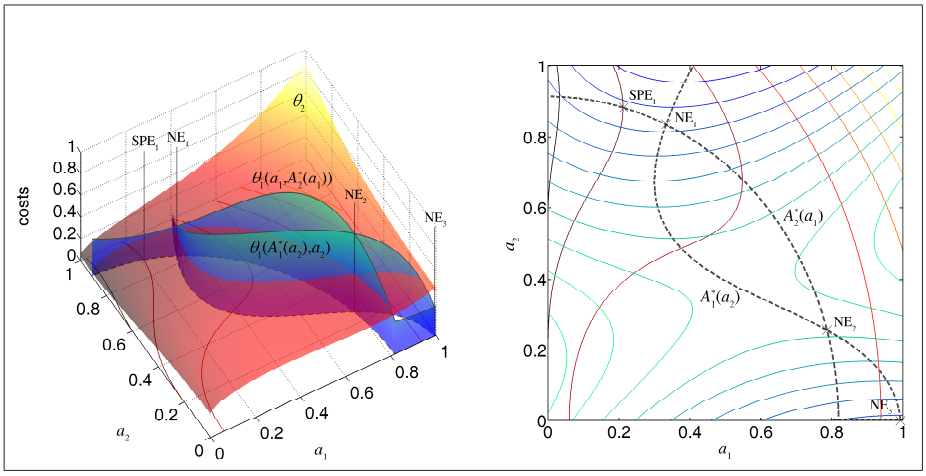


**Figure 3.2:** The left figure shows player 2's objective $\theta_2$ (red surface) as in Figure 3.1 (right) and player 1's costs $\theta_1(A_1^*(a_2), a_2)$ and $\theta_1(a_1, A_2^*(a_1))$, respectively, along the reaction curves of both players. The right figure is a top view of the left plot. The contour curves visualize the players' objectives and the dashed lines their reaction curves. Crosses mark equilibrium points.

spaces, the reaction curves $A_1^*(a_2)$ and $A_2^*(a_1)$ intersect at least once as they are continuous and the image of $A_1^*(a_2)$ is a subset of the domain of $A_2^*(a_1)$ (and vice versa). Loosely speaking, $A_1^*(a_2)$ is a continuous curve that connects the lower and the upper border of the box in Figure 3.2 (right). Likewise, $A_2^*(a_1)$ is a continuous curve from the box's left to right border so that both curves have to intersect. For compact and non-empty action spaces, $A_2^*(a_1)$ is compact and non-empty as well and, hence, there exists at least one minimizing argument of $\theta_1(a_1, A_2^*(a_1))$, *i.e.,* an SPE. We discuss the existence and uniqueness of NE and SPE for prediction games more detailed in Chapters 4.1 and 5.1.

## 3.2 Adversarial Prediction Problems

Before establishing a game theoretical model, we shall study common properties of adversarial prediction problems. We notice that there are basically two types of players involved: The first player type $v_1 = -1$ is called the *learner* and the second $v_2 = +1$ is called the *data generator*. A learner selects a predictive model and a data generator chooses test objects which are presented to the learner. Any combination of prediction model and test data raises certain costs to the players. To detail on these costs and the rules of the game, let us make the following considerations.

### 3.2.1 Number of Players

Prediction games often involve more than two players, that is, the learner and the data generator generally subsume several individuals. Consider therefore our introductory ex-

amples: When fighting against spam emails, network intrusion, and credit card fraud, the data are generally generated by several hundreds, thousands, or even millions of human customers, computer programs, assailants, etc. Likewise, the generated data are often spread over many parties, *i.e.,* learners, such as email service providers, web hosters, and banks. In practice, each of the indefinite number of players is equipped with an own action set and an own cost function. To study the interaction between those parties, we make the following simplifying assumption.

**Assumption 3.1.** *All players of a prediction game interact non-cooperative and act rational with respect to either the learner's action space and cost function or the data generator's action space and cost function, that is, there is no competition among learners or among data generators.*

Based on this assumption, all learners and all data generators can be modeled as two amalgamated players which we call *learner* and *data generator*. In our running example of email spam filtering, we study the competition between recipient and senders, not competition among senders or among filter providers. This is reasonable as there is no conflict between senders who are all interested in the transmission of their emails. This is true for a legitimate email sender, who experiences costs when a legitimate email is erroneously filtered out, just like for an abusive sender who experiences costs when their spam messages are blocked.

### 3.2.2 Number of Repetitions

Beside the number of players, the way the players interact is of importance to describe the interaction. In general, the interaction of the learner and the data generator is an ongoing process: First the data generator chooses a data sample whereupon the learner constructs a prediction model. In response, the data generator changes the data generation model and produces a new sample whereupon the learner adapts their prediction model and so on. As discussed in the previous section, game theory provides us with various game models to anticipate this interaction.

At the first sight, the players' interaction can be considered as a repeated game, since it is an ongoing process. However, this would imply that both players aim at minimizing their costs over a (possibly infinite) period of time at the expense of accepting high costs in a single round. In practice, this seems implausible as, for instance, spam senders typically rent botnets[1] to sent batches of spam emails with the goal of maximizing their profit for each single spam campaign. The same is true for credit card fraudsters who would increase the risk of getting caught when not focusing on each single "round". Likewise, the performance of a spam filter or intrusion detection system must not fall far short just to improve the prediction quality someday. That is why we believe that the ongoing interaction between learner and data generator decouples into a sequence of one-shot games.

[1] A number of mostly home-based computers connect to the Internet that, although their owners are unaware of it, have been compromised by malicious software to send spam emails, spread viruses, perform DoS attacks, etc.

**Assumption 3.2.** *The repeating interactions of the learner and the data generator decouple into a sequence of one-shot prediction games where the players minimize their costs for each round separately.*

Even when considering one-shot games, these games are not independent as, of course, each player is expected to employ all prior knowledge on their opponent including the (partially) observed outcome of the previous game. For instance, the learner observes the true test data of the previous game which may become part of the training data in the next round. However, the players are not expected to conduct an *exploration phase* (such as in repeated games) where the players choose an action just to increase the knowledge about their opponent and, thereby, disregarding their own costs in this round.

## 3.3 Modeling the Prediction Game

In this section we formalize the adversarial prediction problem as an one-shot continuous kernel game of complete information which addresses the previously made considerations. The goal is to (approximately) anticipate the opponents action with the help of the observed training data and potential prior knowledge on the players $V = \{-1, +1\}$.

In the *past*, the data generator $v_1 = +1$ produced a sample $D = \{(x_i, y_i)\}_{i=1}^n$ of $n$ training instances $x_i \in \mathcal{X}$ with corresponding class labels $y_i \in \mathcal{Y} = \{-1, +1\}$ using the unknown data generation model $\gamma$. These object-class pairs are drawn according to a training distribution with density function $p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}(x, y)$. By contrast, *future* object-class pairs, produced by the data generator at application time with generation model $\dot{\gamma}$, are drawn from some test distribution with density $p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$ which may significantly differ from $p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}$.

Recall from the previous chapter, that the task of the learner $v_1 = -1$ is to select the parameter values $\mathbf{w} \in \mathcal{W} \subset \mathbb{R}^m$ of a predictive model $h_{\mathbf{w}}(x) = \operatorname{sign} g_{\mathbf{w}}(x)$ implemented in terms of a linear function $g_{\mathbf{w}} : \mathcal{X} \to \mathbb{R}$ with $g_{\mathbf{w}}(x) = \mathbf{w}^\mathsf{T}\boldsymbol{\phi}(x)$ and feature mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^m$. The learner's theoretical costs equal the *generalization error* (*cf.* Equation 2.5) at application time which is given by

$$\theta_{-1}(\mathbf{w}, p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}) = \int_{\mathcal{X}\times\mathcal{Y}} c_{-1}(x, y)\ell_{-1}(g_{\mathbf{w}}(x), y)\, p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}(x, y)\, \mathrm{d}(x, y).$$

Here, weighting function $c_{-1} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$ and loss function $\ell_{-1} : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}^+$ detail the *weighted loss* $c_{-1}(x, y)\ell_{-1}(g_{\mathbf{w}}(x), y)$ that the learner incurs when the predictive model classifies instance $x$ as $h(x) = \operatorname{sign} g_{\mathbf{w}}(x)$ while the true label is $y$. The positive class- and instance-specific weighting factors $c_{-1}(x, y)$ with $\mathrm{E}_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}[c_{-1}(x, y)] = 1$ specify the importance of minimizing the loss $\ell_{-1}(g_{\mathbf{w}}(x), y)$ for the corresponding object-class pair $(x, y)$. For instance, in spam filtering, the correct classification of non-spam messages can be business-critical for email service providers while failing to detect spam messages runs up processing and storage costs, depending on the size of the message.

The data generator $v_2 = +1$ can modify the data generation process from $\gamma$ to $\dot{\gamma}$. For instance, in practice, spam senders update their campaign templates which are disseminated

to the nodes of botnets. This transformation shifts the training distribution with density $p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}$ to the test distribution with density $p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$. The data generator incurs *transformation costs* by modifying the data generation process which is quantified by $\Omega_{+1}(\gamma, \dot{\gamma})$. This term acts as a regularizer on the transformation and may implicitly constrain the space of possible distribution shifts, depending on the nature of the application that is to be modeled. For instance, the email sender may not be allowed to alter the training distribution for non-spam messages, or to modify the nature of the messages by changing the label from *spam* to *non-spam* or vice versa. Additionally, changing the training distribution for spam messages may run up costs depending on the extent of distortion inflicted on the informational payload. The theoretical costs of the data generator at application time are the sum of the expected prediction costs and the transformation costs,

$$\theta_{+1}(\mathbf{w}, p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}) = \int_{\mathcal{X}\times\mathcal{Y}} c_{+1}(x,y)\ell_{+1}(g_{\mathbf{w}}(x), y)\, p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}(x,y)\,\mathrm{d}(x,y) + \Omega_{+1}(\gamma, \dot{\gamma}),$$

where, in analogy to the learner's costs, $c_{+1}(x,y)\ell_{+1}(g_{\mathbf{w}}(x), y)$ quantifies the weighted loss that the data generator incurs when instance $x$ is labeled as $h(x) = \operatorname{sign} g_{\mathbf{w}}(x)$ while the true class is $y$. The weighting factors $c_{+1}(x,y)$ with $\mathrm{E}_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}[c_{+1}(x,y)] = 1$ express the significance of $(x,y)$ from the perspective of the data generator. In our example scenario, this allows to reflect that costs of correctly or incorrectly classified instances may vary greatly across different physical senders that are aggregated into the amalgamated player.

Since the theoretical costs of both players depend on the unknown data generation models $\gamma$ and $\dot{\gamma}$, they can, for all practical purposes, not be calculated. We can formulate this uncertainty about the true theoretical costs as a Bayesian game by explicitly modeling the distribution $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$, which yields a generative model. However, this would dramatically increase the complexity of the model. Moreover, in classification, generative models are generally less powerful than discriminative predictors. Hence, we abstain from modeling the joint test distribution and focus on regularized, empirical counterparts (*cf.* Equation 2.25) of the theoretical costs based on the training sample $D$. The empirical estimate $\Omega_{+1}(D, \dot{D})$ of the data generator's regularizer $\Omega_{+1}(\gamma, \dot{\gamma})$ penalizes the divergence between training sample $D = \{(x_i, y_i)\}_{i=1}^n$ and a perturbated training sample $\dot{D} = \{(\dot{x}_i, y_i)\}_{i=1}^n$ that would be the outcome of applying the transformation that translates $p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}$ into $p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$ to sample $D$. The learner's cost function, instead of integrating over $p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$, sums over the elements of the perturbated training sample $\dot{D}$.

The empirical costs incurred by the predictive model $h_{\mathbf{w}}(x) = \operatorname{sign} g_{\mathbf{w}}(x)$ with parameter vector $\mathbf{w}$ and the shift from $p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}$ to $p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$ amount to

$$\hat{\theta}_{-1}(\mathbf{w}, \dot{D}) = \sum_{i=1}^{n} c_{-1,i}\ell_{-1}(g_{\mathbf{w}}(\dot{x}_i), y_i) + \rho_{-1}\Omega_{-1}(\mathbf{w}), \tag{3.1}$$

$$\hat{\theta}_{+1}(\mathbf{w}, \dot{D}) = \sum_{i=1}^{n} c_{+1,i}\ell_{+1}(g_{\mathbf{w}}(\dot{x}_i), y_i) + \rho_{+1}\Omega_{+1}(D, \dot{D}), \tag{3.2}$$

where we have replaced the weighting terms $\frac{1}{n}c_v(\dot{x}_i, y_i)$ by constant cost factors $c_{v,i} > 0$

with $\sum_i c_{v,i} = 1$. The trade-off between the empirical loss and the regularizer is controlled by each player's regularization parameter $\rho_v > 0$ for $v \in V = \{-1, +1\}$.

Both players' empirical cost functions can still only be evaluated after the learner has committed to parameters $\mathbf{w}$ and the data generator to a transformation of the data generation process. However this transformation needs only be represented in terms of the effects that it will have on the training sample $D$, that is, $\dot{D}$.

The learner's regularizer $\Omega_{-1}(\mathbf{w})$ in (3.1) accounts for the fact that $\dot{D}$ does not constitute the future test data itself, but is merely a training sample transformed to reflect the test distribution and then used to learn the model parameters $\mathbf{w}$. Future test instances will be generated under $p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$ at application time *after* both players have committed to their actions.

Note, that either player's empirical costs $\hat{\theta}_v$ depend on both players' actions $\mathbf{w} \in \mathcal{W}$ and $\dot{D} \in (\mathcal{X} \times \mathcal{Y})^n$, respectively. Because of the potentially conflicting players' interests, the decision process for $\mathbf{w}$ and $\dot{D}$ becomes a non-cooperative two-player game which we call a *prediction game*.

**Definition 3.5.** *A* prediction game *is a non-cooperative continuous kernel game of complete information with*

1. *Player set* $V := \{-1, +1\}$*, where* $-1$ *denotes the* learner *and* $+1$ *denotes the* data generator*;*
2. *Non-empty, compact, and convex action sets* $A_{-1} := \mathcal{W}$ *and* $A_{+1} := (\mathcal{X} \times \mathcal{Y})^n$*;*
3. *Continuous cost functions* $\hat{\theta}_v(\mathbf{w}, \dot{D})$ *for* $v \in V$ *as defined in* (3.1) *and* (3.2)*.*

Depending of the order at which the players interact, one can identify distinct classes of prediction games. We briefly discuss these classes in the next section.

## 3.4 Classes of Prediction Games

As previously discussed, the order at which the players decide for an action establishes different game settings. In one-shot games one can distinguish between three cases.

First, the learner and the data generator are assumed to make their decisions simultaneously. This setting amounts to a static game in which both players decide for an action before observing their opponents' move. We study this kind of interaction, that is, **Nash prediction games**, in Chapter 4. The assumption of simultaneously acting players can, for example, be motivated by outbound spam filtering. Here, the sender has to make an estimate of the spam filter before sending the emails. When ignoring the filter, the sender's email account is likely to be blocked by the email service provider. Hence, there are little chances for the sender to react on the filter *after* its release. Likewise, the learner has to predict the sender's action in advance, since making a wrong prediction on the class label of outgoing emails may cause the server to be blocked by other email service providers. As

for the sender, there would be no or only little time to react in order to keep the service available.

In the second case we consider a learner who moves *before* the data generator. Here, the learner first chooses a predictive model. Then, the data generator observes the learner's action and reacts by deciding for their own move. To decide on their action, the learner has to anticipate the move of the data generator in advance, which establishes a **Stackelberg prediction game** (*cf.* Chapter 5). A special kind of a Stackelberg game is the zero-sum prediction game in which case both players' interests are antagonistic, *i.e.,* a worst-case setting. Stackelberg prediction games address, for instance, the problem of client-side spam filtering where the spam filter is regularly released by some network security provider. Here, the filter has to be designed so that it anticipates the sender's move within two release cycles whereas the sender can somehow optimally react on a released filter.

Finally, we suppose that the learner has to respond on the data generator, that is, the data generator is expected to move first by producing a sample of (test) data. These data instances, but not the corresponding target attributes, are then observed by the learner who builds a predictive model based on the training data and this unlabeled sample. As the data generator potentially chose their action in response to a previous prediction model, the underlying distribution of the test and the training sample may significantly differ. In this setting, which we discuss in Chapter 6, the data generator's move is fixed *before* the learner has to decide for a predictor, so that the learning problem reduces to learning under **covariate shift**. In the example of email spam filtering, this model reflects the scenario of a server-side inbound spam filter. Here, the learner has to make decisions on a variety of newly incoming emails without or late getting feedback on the class labels by the users. These emails are assumed to be the data generator's move, chosen in response to a previously released spam filter.

In all three cases we only focus on a game-theoretical optimal move of the learner. That is, we do not answer the question what an optimal change of the data generation process would be, as we never model the future data generation process but its effect on the training data.

# 4 Nash Prediction Games

In this chapter we study static prediction games in which both players act simultaneously. We call this class of prediction games *Nash prediction games*.

To identify the learner's optimal action, we establish the concept of Nash equilibrium of prediction games in Section 4.1. A Nash equilibrium is a pair of actions chosen so that no player gains a benefit by unilaterally selecting a different action. If a game has a unique Nash equilibrium and is played by rational players that aim at maximizing their optimization criteria, it is reasonable for each player to assume that the opponent will follow the Nash equilibrium. If one player follows the Nash equilibrium, the optimal move for the other player is to follow this equilibrium as well. If, however, multiple equilibria exist and the players choose their action according to distinct ones, then the resulting combination may be arbitrarily disadvantageous for either player. We therefore study in Section 4.1.2 whether prediction games have a unique Nash equilibrium. In Section 4.2 we derive two algorithms to identify the unique Nash equilibrium, in case it exists. The first algorithm computes the minimax solution of the Nikaido-Isoda function which is by construction a solution of the Nash prediction game, too. In the second algorithm, we reformulate the Nash prediction game into a variational inequality problem which is then solved by a modified extragradient method. In Section 4.3 we study the applicability of general kernel function, and present two instances of Nash prediction games in Section 4.4: Nash logistic regression and the Nash support vector machine. Related work is discussed in Section 4.5. Finally, in Section 4.6, we empirically verify the assumptions made in the modeling process and compare the performance of Nash instances with baseline methods on several email corpora, including a corpus from an email service provider.

## 4.1 Nash Solution to Prediction Games

As discussed in Section 3.1, the outcome of a prediction game (*cf.* Definition 3.5) is one particular combination of actions $(\mathbf{w}^*, \dot{D}^*)$ that runs up costs $\hat{\theta}_v(\mathbf{w}^*, \dot{D}^*)$ for the players. Each player is aware that this outcome is affected by both players' actions and that, consequently, their potential to choose an action can have an impact on the other player's decision.

In this chapter, we model this decision process for $\mathbf{w}^*$ and $\dot{D}^*$ as a static two-player game with complete information, that is, a game where both players commit to an action simultaneously and know their opponent's cost function and action space. In addition, we assume that both players act rationally in the sense of seeking the greatest possible personal advantage. This leads to so called equilibrium strategies which form steady states of the

game in which neither player has an incentive to unilaterally change their plan of actions. In static games, equilibrium strategies are called *Nash equilibria*, which is why we refer to the resulting predictive model as *Nash prediction game* (NPG). In a two-player game, a Nash equilibrium is defined as a pair of actions so that no player can benefit from changing their action solely, that is,

$$\begin{aligned} \hat{\theta}_{-1}(\mathbf{w}^*, \dot{D}^*) &= \min_{\mathbf{w} \in \mathcal{W}} \hat{\theta}_{-1}(\mathbf{w}, \dot{D}^*), \\ \hat{\theta}_{+1}(\mathbf{w}^*, \dot{D}^*) &= \min_{\dot{D} \in (\mathcal{X} \times \mathcal{Y})^n} \hat{\theta}_{+1}(\mathbf{w}^*, \dot{D}), \end{aligned}$$

where $\mathcal{W}$ and $(\mathcal{X} \times \mathcal{Y})^n$ denote the players' action sets.

However, a static prediction game may not have a Nash equilibrium, or it may possess multiple equilibria. If $(\mathbf{w}^*, \dot{D}^*)$ and $(\mathbf{w}', \dot{D}')$ are distinct Nash equilibria and each player decides to act according to a different one of them, then combinations $(\mathbf{w}^*, \dot{D}')$ and $(\mathbf{w}', \dot{D}^*)$ may incur arbitrarily high costs for both players. Hence, one can argue that it is rational for the players to play a Nash equilibrium only when the following assumption is satisfied.

**Assumption 4.1.** *The following statements hold:*

1. *Both players choose their actions simultaneously;*
2. *Both players have full knowledge about both (empirical) cost functions $\hat{\theta}_v(\mathbf{w}, \dot{D})$ defined in* (3.1) *and* (3.2)*, and both action sets $\mathcal{W}$ and $(\mathcal{X} \times \mathcal{Y})^n$;*
3. *Both players act rational with respect to their cost function in the sense of securing their lowest possible costs;*
4. *A unique Nash equilibrium exists.*

Whether Assumptions 4.1.1-4.1.3 are adequate—especially, the assumption of simultaneous actions—strongly depends on the application. For example, in some applications the data generator may unilaterally be able to acquire information about the model parameters $\mathbf{w}$ before committing to $\dot{D}$. Such situations are better modeled as a Stackelberg competition (*cf.* Section 5). On the other hand, when the learner is able to treat any executed action as part of the training data $D$ and update the parameter vector $\mathbf{w}$, the setting is better modeled as repeated executions of a static game with simultaneous actions. The adequateness of Assumption 4.1.4, which we discuss in the following sections, depends on the chosen loss functions, the cost factors, and the regularizers.

### 4.1.1 Existence of a Nash Equilibrium

In this section we will derive sufficient conditions for the existence of a Nash equilibrium. To this end, we first define

$$\begin{aligned} \mathbf{x} &:= \left[\boldsymbol{\phi}(x_1)^\mathsf{T}, \boldsymbol{\phi}(x_2)^\mathsf{T}, \ldots, \boldsymbol{\phi}(x_n)^\mathsf{T}\right]^\mathsf{T} \in \boldsymbol{\phi}(\mathcal{X})^n \subset \mathbb{R}^{m \cdot n}, \\ \dot{\mathbf{x}} &:= \left[\boldsymbol{\phi}(\dot{x}_1)^\mathsf{T}, \boldsymbol{\phi}(\dot{x}_2)^\mathsf{T}, \ldots, \boldsymbol{\phi}(\dot{x}_n)^\mathsf{T}\right]^\mathsf{T} \in \boldsymbol{\phi}(\mathcal{X})^n \subset \mathbb{R}^{m \cdot n}, \end{aligned}$$

as long, concatenated, column vectors induced by feature mapping $\boldsymbol{\phi}$, training sample $D = \{(x_i, y_i)\}_{i=1}^n$, and transformed training sample $\dot{D} = \{(\dot{x}_i, y_i)\}_{i=1}^n$, respectively. For terminological harmony, we refer to vector $\dot{\mathbf{x}}$ as the data generator's action with corresponding action space $\boldsymbol{\phi}(\mathcal{X})^n$, where we assume that the data generator is not allowed to modify the target attribute of an object.

We make the following assumptions on the action spaces and the cost functions which enables us to state the main result on the existence of at least one Nash equilibrium in Lemma 4.1.

**Assumption 4.2.** *The players' cost functions defined in Equations 3.1 and 3.2, and their action spaces $\mathcal{W}$ and $\boldsymbol{\phi}(\mathcal{X})^n$ satisfy the properties:*

1. *Loss functions $\ell_v(z, y)$ with $v \in \{-1, +1\}$ are convex and twice-continuously differentiable with respect to $z \in \mathbb{R}$ for all fixed $y \in \mathcal{Y}$;*
2. *Regularizers $\Omega_v$ are uniformly strongly convex and twice-continuously differentiable with respect to $\mathbf{w} \in \mathcal{W}$ and $\dot{\mathbf{x}} \in \boldsymbol{\phi}(\mathcal{X})^n$, respectively;*
3. *Action spaces $\mathcal{W}$ and $\boldsymbol{\phi}(\mathcal{X})^n$ are non-empty, compact, and convex subsets of finite-dimensional Euclidean spaces $\mathbb{R}^m$ and $\mathbb{R}^{m \cdot n}$, respectively.*

**Lemma 4.1.** *Under Assumption 4.2, at least one equilibrium point $(\mathbf{w}^*, \dot{\mathbf{x}}^*) \in \mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n$ of the Nash prediction game defined by*

$$\begin{array}{l|l} \min\limits_{\mathbf{w}} \quad \hat{\theta}_{-1}(\mathbf{w}, \dot{\mathbf{x}}^*) & \min\limits_{\dot{\mathbf{x}}} \quad \hat{\theta}_{+1}(\mathbf{w}^*, \dot{\mathbf{x}}) \\ \text{s.t.} \quad \mathbf{w} \in \mathcal{W} & \text{s.t.} \quad \dot{\mathbf{x}} \in \boldsymbol{\phi}(\mathcal{X})^n \end{array} \tag{4.1}$$

*exists.*

*Proof.* Each player $v$'s cost function is a sum over $n$ loss terms resulting from loss function $\ell_v$ and regularizer $\Omega_v$. By Assumption 4.2, these loss functions are convex and continuous, and the regularizers are uniformly strongly convex and continuous. Hence, both cost functions $\hat{\theta}_{-1}(\mathbf{w}, \dot{\mathbf{x}})$ and $\hat{\theta}_{+1}(\mathbf{w}, \dot{\mathbf{x}})$ are continuous in all arguments and uniformly strongly convex in $\mathbf{w} \in \mathcal{W}$ and $\dot{\mathbf{x}} \in \boldsymbol{\phi}(\mathcal{X})^n$, respectively. As both action spaces $\mathcal{W}$ and $\boldsymbol{\phi}(\mathcal{X})^n$ are non-empty, compact, and convex subsets of finite-dimensional Euclidean spaces, a Nash equilibrium exists—see Theorem 4.3 of [3]. □

### 4.1.2 Uniqueness of the Nash Equilibrium

We will now derive conditions for the uniqueness of an equilibrium of the Nash prediction game defined in (4.1). We first reformulate the two-player game into an $(n + 1)$-player game. In Lemma 4.2 we then present a sufficient condition for the uniqueness of the Nash equilibrium in this game, and by applying Proposition 4.4 and Lemma 4.5-4.7 we verify whether this condition is met. Finally, we state the main result in Theorem 4.8: The Nash equilibrium is unique under certain properties of the loss functions, the regularizers, and the cost factors which all can be verified easily.

Taking into account the Cartesian product structure of the data generator's action space $\boldsymbol{\phi}(\mathcal{X})^n$, it is not difficult to see that $(\mathbf{w}^*, \dot{\mathbf{x}}^*)$ with $\dot{\mathbf{x}}^* = \left[\dot{\mathbf{x}}_1^{*\mathsf{T}}, \ldots, \dot{\mathbf{x}}_n^{*\mathsf{T}}\right]^{\mathsf{T}}$ and $\dot{\mathbf{x}}_i^* := \boldsymbol{\phi}(\dot{x}_i^*)$ is a solution of the two-player game if, and only if, $(\mathbf{w}^*, \dot{\mathbf{x}}_1^*, \ldots, \dot{\mathbf{x}}_n^*)$ is a Nash equilibrium of the $(n+1)$-player game defined by

$$\begin{array}{l|l|c|l} \min\limits_{\mathbf{w}} \ \hat{\theta}_{-1}(\mathbf{w}, \dot{\mathbf{x}}) & \min\limits_{\dot{\mathbf{x}}_1} \ \hat{\theta}_{+1}(\mathbf{w}, \dot{\mathbf{x}}) & \cdots & \min\limits_{\dot{\mathbf{x}}_n} \ \hat{\theta}_{+1}(\mathbf{w}, \dot{\mathbf{x}}) \\ \text{s.t.} \ \ \mathbf{w} \in \mathcal{W} & \text{s.t.} \ \ \dot{\mathbf{x}}_1 \in \boldsymbol{\phi}(\mathcal{X}) & \cdots & \text{s.t.} \ \ \dot{\mathbf{x}}_n \in \boldsymbol{\phi}(\mathcal{X}) \end{array}, \tag{4.2}$$

which results from (4.1) by repeating $n$ times the cost function $\hat{\theta}_{+1}$ and minimizing this function with respect to $\dot{\mathbf{x}}_i \in \boldsymbol{\phi}(\mathcal{X})$ for $i = 1, \ldots, n$. Then, the *pseudo-gradient* (in the sense of Rosen, [59]) of the game in (4.2) is defined by

$$\mathbf{g}_{\mathbf{r}}(\mathbf{w}, \dot{\mathbf{x}}) := \begin{bmatrix} r_0 \nabla_{\mathbf{w}} \hat{\theta}_{-1}(\mathbf{w}, \dot{\mathbf{x}}) \\ r_1 \nabla_{\dot{\mathbf{x}}_1} \hat{\theta}_{+1}(\mathbf{w}, \dot{\mathbf{x}}) \\ r_2 \nabla_{\dot{\mathbf{x}}_2} \hat{\theta}_{+1}(\mathbf{w}, \dot{\mathbf{x}}) \\ \vdots \\ r_n \nabla_{\dot{\mathbf{x}}_n} \hat{\theta}_{+1}(\mathbf{w}, \dot{\mathbf{x}}) \end{bmatrix} \in \mathbb{R}^{m+m\cdot n}, \tag{4.3}$$

with any fixed vector $\mathbf{r} = [r_0, r_1, \ldots, r_n]^{\mathsf{T}}$ where $r_i > 0$ for $i = 0, \ldots, n$. The derivative of $\mathbf{g}_{\mathbf{r}}$, that is, the *pseudo-Jacobian* of (4.2), is given by

$$\mathbf{J}_{\mathbf{r}}(\mathbf{w}, \dot{\mathbf{x}}) := \mathbf{D} \begin{bmatrix} \nabla^2_{\mathbf{w},\mathbf{w}} \hat{\theta}_{-1}(\mathbf{w}, \dot{\mathbf{x}}) & \nabla^2_{\mathbf{w},\dot{\mathbf{x}}} \hat{\theta}_{-1}(\mathbf{w}, \dot{\mathbf{x}}) \\ \nabla^2_{\dot{\mathbf{x}},\mathbf{w}} \hat{\theta}_{+1}(\mathbf{w}, \dot{\mathbf{x}}) & \nabla^2_{\dot{\mathbf{x}},\dot{\mathbf{x}}} \hat{\theta}_{+1}(\mathbf{w}, \dot{\mathbf{x}}) \end{bmatrix},$$

where

$$\mathbf{D} := \begin{bmatrix} r_0 \mathbf{I}_m & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & r_1 \mathbf{I}_m & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & r_n \mathbf{I}_m \end{bmatrix} \in \mathbb{R}^{(m+m\cdot n)\times(m+m\cdot n)}.$$

Notice, that these derivatives, *i.e.,* pseudo-gradient $\mathbf{g}_{\mathbf{r}}$ and pseudo-Jacobian $\mathbf{J}_{\mathbf{r}}$, exist in view of Assumption 4.2. The above definition of the pseudo-Jacobian enables us to state the following result about the uniqueness of a Nash equilibrium.

**Lemma 4.2.** *Let Assumption 4.2 hold and suppose there exists a fixed vector* $\mathbf{r} = [r_0, r_1, \ldots, r_n]^{\mathsf{T}}$ *with* $r_i > 0$ *for all* $i = 0, 1, \ldots, n$ *so that the corresponding pseudo-Jacobian* $\mathbf{J}_{\mathbf{r}}(\mathbf{w}, \dot{\mathbf{x}})$ *is positive definite for all* $(\mathbf{w}, \dot{\mathbf{x}}) \in \mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n$*. Then, the Nash prediction game in* (4.1) *has a unique equilibrium.*

*Proof.* The existence of a Nash equilibrium follows from Lemma 4.1. To prove the uniqueness, recall from our previous discussion, that the original Nash game in (4.1) has a unique solution if, and only if, the game from (4.2) with one learner and $n$ data generators admits a unique solution. In view of Theorem 2 of [59], the latter attains a unique solution if the pseudo-gradient $\mathbf{g}_{\mathbf{r}}$ is strictly monotone, *i.e.,* for all actions $\mathbf{w}, \mathbf{w}' \in \mathcal{W}$ and $\dot{\mathbf{x}}, \dot{\mathbf{x}}' \in \boldsymbol{\phi}(\mathcal{X})^n$

inequality

$$\left(\mathbf{g}_{\mathbf{r}}(\mathbf{w},\dot{\mathbf{x}}) - \mathbf{g}_{\mathbf{r}}(\mathbf{w}',\dot{\mathbf{x}}')\right)^{\mathsf{T}} \left( \begin{bmatrix} \mathbf{w} \\ \dot{\mathbf{x}} \end{bmatrix} - \begin{bmatrix} \mathbf{w}' \\ \dot{\mathbf{x}}' \end{bmatrix} \right) > 0$$

holds. A sufficient condition for this pseudo-gradient being strictly monotone is the positive definiteness of the pseudo-Jacobian $\mathbf{J}_{\mathbf{r}}$ (see *e.g.,* Theorem 7.11 in [34] or Theorem 6 in [59]). □

To verify if the condition of Lemma 4.2 is satisfied, we analyze the pseudo-Jacobian $\mathbf{J}_{\mathbf{r}}(\mathbf{w},\dot{\mathbf{x}})$. Throughout this thesis, we denote by $\ell'_v(z,y)$ and $\ell''_v(z,y)$ the first and second derivative of the mapping $\ell_v(z,y)$ with respect to $z \in \mathbb{R}$. A direct calculation shows that the first-order partial derivatives are given by

$$\nabla_{\mathbf{w}} \hat{\theta}_{-1}(\mathbf{w},\dot{\mathbf{x}}) = \sum_{i=1}^{n} c_{-1,i} \ell'_{-1}(\dot{\mathbf{x}}_i^{\mathsf{T}}\mathbf{w}, y_i)\dot{\mathbf{x}}_i + \rho_{-1}\nabla_{\mathbf{w}}\Omega_{-1}(\mathbf{w}), \tag{4.4}$$

$$\nabla_{\dot{\mathbf{x}}_i} \hat{\theta}_{+1}(\mathbf{w},\dot{\mathbf{x}}) = c_{+1,i}\ell'_{+1}(\dot{\mathbf{x}}_i^{\mathsf{T}}\mathbf{w}, y_i)\mathbf{w} + \rho_{+1}\nabla_{\dot{\mathbf{x}}_i}\Omega_{+1}(\mathbf{x},\dot{\mathbf{x}}). \tag{4.5}$$

This allows us to calculate the entries of the pseudo-Jacobian:

$$\begin{aligned}
\nabla^2_{\mathbf{w},\mathbf{w}} \hat{\theta}_{-1}(\mathbf{w},\dot{\mathbf{x}}) &= \sum_{i=1}^{n} c_{-1,i}\ell''_{-1}(\dot{\mathbf{x}}_i^{\mathsf{T}}\mathbf{w}, y_i)\dot{\mathbf{x}}_i\dot{\mathbf{x}}_i^{\mathsf{T}} + \rho_{-1}\nabla^2_{\mathbf{w},\mathbf{w}}\Omega_{-1}(\mathbf{w}), \\
\nabla^2_{\mathbf{w},\dot{\mathbf{x}}_i} \hat{\theta}_{-1}(\mathbf{w},\dot{\mathbf{x}}) &= c_{-1,i}\ell''_{-1}(\dot{\mathbf{x}}_i^{\mathsf{T}}\mathbf{w}, y_i)\dot{\mathbf{x}}_i\mathbf{w}^{\mathsf{T}} + c_{-1,i}\ell'_{-1}(\dot{\mathbf{x}}_i^{\mathsf{T}}\mathbf{w}, y_i)\mathbf{I}_m, \\
\nabla^2_{\dot{\mathbf{x}}_i,\mathbf{w}} \hat{\theta}_{+1}(\mathbf{w},\dot{\mathbf{x}}) &= c_{+1,i}\ell''_{+1}(\dot{\mathbf{x}}_i^{\mathsf{T}}\mathbf{w}, y_i)\mathbf{w}\dot{\mathbf{x}}_i^{\mathsf{T}} + c_{+1,i}\ell'_{+1}(\dot{\mathbf{x}}_i^{\mathsf{T}}\mathbf{w}, y_i)\mathbf{I}_m, \\
\nabla^2_{\dot{\mathbf{x}}_i,\dot{\mathbf{x}}_j} \hat{\theta}_{+1}(\mathbf{w},\dot{\mathbf{x}}) &= \begin{cases} c_{+1,i}\ell''_{+1}(\dot{\mathbf{x}}_i^{\mathsf{T}}\mathbf{w}, y_i)\mathbf{w}\mathbf{w}^{\mathsf{T}} + \rho_{+1}\nabla^2_{\dot{\mathbf{x}}_i,\dot{\mathbf{x}}_i}\Omega_{+1}(\mathbf{x},\dot{\mathbf{x}}) & \text{, if } i = j, \\ \rho_{+1}\nabla^2_{\dot{\mathbf{x}}_i,\dot{\mathbf{x}}_j}\Omega_{+1}(\mathbf{x},\dot{\mathbf{x}}) & \text{, if } i \neq j. \end{cases}
\end{aligned}$$

Let us define the matrix

$$\boldsymbol{\Upsilon}(\mathbf{w},\dot{\mathbf{x}}) := \begin{bmatrix} — \dot{\mathbf{x}}_1^{\mathsf{T}} — & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & & \vdots \\ — \dot{\mathbf{x}}_n^{\mathsf{T}} — & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & — \mathbf{w}^{\mathsf{T}} — & & \mathbf{0} \\ \vdots & & \ddots & \\ \mathbf{0} & \mathbf{0} & & — \mathbf{w}^{\mathsf{T}} — \end{bmatrix} \in \mathbb{R}^{2n \times (m + m \cdot n)},$$

and let us denote the smallest eigenvalues of the Hessians of the regularizers on the corresponding action spaces $\mathcal{W}$ and $\boldsymbol{\phi}(\mathcal{X})^n$ by

$$\lambda_{-1} := \inf_{\mathbf{w} \in \mathcal{W}} \lambda_{\min}\left(\nabla^2_{\mathbf{w},\mathbf{w}}\Omega_{-1}(\mathbf{w})\right), \tag{4.6}$$

$$\lambda_{+1} := \inf_{\dot{\mathbf{x}} \in \boldsymbol{\phi}(\mathcal{X})^n} \lambda_{\min}\left(\nabla^2_{\dot{\mathbf{x}},\dot{\mathbf{x}}}\Omega_{+1}(\mathbf{x},\dot{\mathbf{x}})\right), \tag{4.7}$$

where $\lambda_{\min}(\mathbf{A})$ denotes the smallest eigenvalue of the symmetric matrix $\mathbf{A}$.

**Remark 4.3.** Note that the minimum in (4.6) and (4.7) is attained and is strictly positive: The mapping $\lambda_{\min} : \mathcal{M}^{k\times k} \to \mathbb{R}$ is concave on the set of symmetric matrices $\mathcal{M}^{k\times k}$ of dimension $k \times k$ (*cf.* Example 3.10 in [13]), and in particular, it therefore follows that this mapping is continuous. Furthermore, the mappings $u_{-1} : \mathcal{W} \to \mathcal{M}^{m\times m}$ with $u_{-1}(\mathbf{w}) := \nabla^2_{\mathbf{w},\mathbf{w}}\Omega_{-1}(\mathbf{w})$ and $u_{+1} : \boldsymbol{\phi}(\mathcal{X})^n \to \mathcal{M}^{m\cdot n\times m\cdot n}$ with $u_{+1}(\dot{\mathbf{x}}) := \nabla^2_{\dot{\mathbf{x}},\dot{\mathbf{x}}}\Omega_{+1}(\mathbf{x},\dot{\mathbf{x}})$ are continuous (for any fixed $\mathbf{x}$) by Assumption 4.2. Hence, the mappings $\mathbf{w} \mapsto \lambda_{\min}(u_{-1}(\mathbf{w}))$ and $\dot{\mathbf{x}} \mapsto \lambda_{\min}(u_{+1}(\dot{\mathbf{x}}))$ are also continuous since each is precisely the composition $\lambda_{\min} \circ u_v$ of the continuous functions $\lambda_{\min}$ and $u_v$ for $v \in \{-1,+1\}$. Taking into account that a continuous mapping on a non-empty compact set attains its minimum, it follows that there exist elements $\mathbf{w} \in \mathcal{W}$ and $\dot{\mathbf{x}} \in \boldsymbol{\phi}(\mathcal{X})^n$ so that

$$\begin{aligned}\lambda_{-1} &= \lambda_{\min}\left(\nabla^2_{\mathbf{w},\mathbf{w}}\Omega_{-1}(\mathbf{w})\right),\\ \lambda_{+1} &= \lambda_{\min}\left(\nabla^2_{\dot{\mathbf{x}},\dot{\mathbf{x}}}\Omega_{+1}(\mathbf{x},\dot{\mathbf{x}})\right).\end{aligned}$$

Moreover, since the Hessians of the regularizers are positive definite by Assumption 4.2, we see that $\lambda_v > 0$ holds for $v \in \{-1,+1\}$. ◇

Using the definition of $\boldsymbol{\Upsilon}(\mathbf{w},\dot{\mathbf{x}})$, $\lambda_v$, and the abbreviations

$$\begin{aligned}\ell'_{v,i} &:= \ell'_v(\dot{\mathbf{x}}_i^\mathsf{T}\mathbf{w}, y_i) \quad i = 1,\dots,n,\\ \ell''_{v,i} &:= \ell''_v(\dot{\mathbf{x}}_i^\mathsf{T}\mathbf{w}, y_i) \quad i = 1,\dots,n,\\ \mathbf{G}_v &:= \operatorname{diag}(c_{v,1}\ell''_{v,1},\dots,c_{v,n}\ell''_{v,n}) \in \mathbb{R}^{n\times n}\end{aligned}$$

for both players $v \in \{-1,+1\}$, we can summarize the previous discussion.

**Proposition 4.4.** *The pseudo-Jacobian has the representation*

$$\mathbf{J}_\mathbf{r}(\mathbf{w},\dot{\mathbf{x}}) = \mathbf{J}^{(1)}_\mathbf{r}(\mathbf{w},\dot{\mathbf{x}}) + \mathbf{J}^{(2)}_\mathbf{r}(\mathbf{w},\dot{\mathbf{x}}) + \mathbf{J}^{(3)}_\mathbf{r}(\mathbf{w},\dot{\mathbf{x}}) \tag{4.8}$$

*where*

$$\begin{aligned}\mathbf{J}^{(1)}_\mathbf{r}(\mathbf{w},\dot{\mathbf{x}}) &:= \mathbf{D}\boldsymbol{\Upsilon}(\mathbf{w},\dot{\mathbf{x}})^\mathsf{T}\begin{bmatrix}\mathbf{G}_{-1} & \mathbf{G}_{-1}\\ \mathbf{G}_{+1} & \mathbf{G}_{+1}\end{bmatrix}\boldsymbol{\Upsilon}(\mathbf{w},\dot{\mathbf{x}}),\\ \mathbf{J}^{(2)}_\mathbf{r}(\mathbf{w},\dot{\mathbf{x}}) &:= \mathbf{D}\begin{bmatrix}\rho_{-1}\lambda_{-1}\mathbf{I}_m & c_{-1,1}\ell'_{-1,1}\mathbf{I}_m & \cdots & c_{-1,n}\ell'_{-1,n}\mathbf{I}_m\\ c_{+1,1}\ell'_{+1,1}\mathbf{I}_m & \rho_{+1}\lambda_{+1}\mathbf{I}_m & \cdots & \mathbf{0}\\ \vdots & \vdots & \ddots & \vdots\\ c_{+1,n}\ell'_{+1,n}\mathbf{I}_m & \mathbf{0} & \cdots & \rho_{+1}\lambda_{+1}\mathbf{I}_m\end{bmatrix},\\ \mathbf{J}^{(3)}_\mathbf{r}(\mathbf{w},\dot{\mathbf{x}}) &:= \mathbf{D}\begin{bmatrix}\rho_{-1}\nabla^2_{\mathbf{w},\mathbf{w}}\Omega_{-1}(\mathbf{w}) - \rho_{-1}\lambda_{-1}\mathbf{I}_m & \mathbf{0}\\ \mathbf{0} & \rho_{+1}\nabla^2_{\dot{\mathbf{x}},\dot{\mathbf{x}}}\Omega_{+1}(\mathbf{x},\dot{\mathbf{x}}) - \rho_{+1}\lambda_{+1}\mathbf{I}_{m\cdot n}\end{bmatrix}.\end{aligned}$$

Recall, that we want to investigate whether there is some fixed positive vector $\mathbf{r}$ so that $\mathbf{J}_\mathbf{r}(\mathbf{w},\dot{\mathbf{x}})$ is positive definite for each pair of actions $(\mathbf{w},\dot{\mathbf{x}}) \in \mathcal{W}\times\boldsymbol{\phi}(\mathcal{X})^n$. A sufficient condition for the pseudo-Jacobian $\mathbf{J}_\mathbf{r}(\mathbf{w},\dot{\mathbf{x}})$ to be positive definite is that $\mathbf{J}^{(1)}_\mathbf{r}(\mathbf{w},\dot{\mathbf{x}})$, $\mathbf{J}^{(2)}_\mathbf{r}(\mathbf{w},\dot{\mathbf{x}})$,

and $\mathbf{J}_{\mathbf{r}}^{(3)}(\mathbf{w}, \dot{\mathbf{x}})$ are positive semi-definite and at least one of these matrices is positive definite. Before discussing these matrices separately, let us define $r_0 := 1$, $r_i := \frac{c_{-1,i}}{c_{+1,i}} > 0$ for all $i = 1, \ldots, n$, with corresponding matrix

$$\mathbf{D} := \begin{bmatrix} \mathbf{I}_m & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \frac{c_{-1,1}}{c_{+1,1}}\mathbf{I}_m & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \frac{c_{-1,n}}{c_{+1,n}}\mathbf{I}_m \end{bmatrix}, \tag{4.9}$$

and let us make the following assumption on the loss functions $\ell_v$ and the regularizers $\Omega_v$ for $v \in \{-1, +1\}$. Instances of these functions satisfying Assumptions 4.2 and 4.3 will be given in Section 4.4.

**Assumption 4.3.** *For all $\mathbf{w} \in \mathcal{W}$ and $\dot{\mathbf{x}} \in \boldsymbol{\phi}(\mathcal{X})^n$ with $\dot{\mathbf{x}} = \left[\dot{\mathbf{x}}_1^\mathsf{T}, \ldots, \dot{\mathbf{x}}_n^\mathsf{T}\right]^\mathsf{T}$ the following conditions are satisfied:*

1. *Both players' loss functions have the same curvature, that is, the second derivatives of the loss functions are equal for all $y \in \mathcal{Y}$ and $i = 1, \ldots, n$,*

$$\ell''_{-1}(g_{\mathbf{w}}(\dot{\mathbf{x}}_i), y) = \ell''_{+1}(g_{\mathbf{w}}(\dot{\mathbf{x}}_i), y),$$

2. *Both players' regularization parameters satisfy*

$$\rho_{-1}\rho_{+1} > \tau^2 \frac{1}{\lambda_{-1}\lambda_{+1}} \mathbf{c}_{-1}^\mathsf{T}\mathbf{c}_{+1},$$

   *where $\lambda_{-1}$, $\lambda_{+1}$ are the smallest eigenvalues of the Hessians of the regularizers specified in (4.6) and (4.7), $\mathbf{c}_v = [c_{v,1}, c_{v,2}, \ldots, c_{v,n}]^\mathsf{T}$, and*

$$\tau = \sup_{(\mathbf{x},y)\in\boldsymbol{\phi}(\mathcal{X})\times\mathcal{Y}} \frac{1}{2} \left|\ell'_{-1}(g_{\mathbf{w}}(\mathbf{x}), y) + \ell'_{+1}(g_{\mathbf{w}}(\mathbf{x}), y)\right|, \tag{4.10}$$

3. *For all $i = 1, \ldots, n$, either both players have equal instance-specific cost factors, $c_{-1,i} = c_{+1,i}$, or the partial derivative $\nabla_{\dot{\mathbf{x}}_i}\Omega_{+1}(\mathbf{x}, \dot{\mathbf{x}})$ of the data generator's regularizer is independent of $\dot{\mathbf{x}}_j$ for all $j \neq i$.*

Notice, that $\tau$ in Equation 4.10 can be chosen finite as the set $\boldsymbol{\phi}(\mathcal{X}) \times \mathcal{Y}$ is assumed to be compact, and consequently, the values of both continuous mappings $\ell'_{-1}(g_{\mathbf{w}}(\mathbf{x}), y)$ and $\ell'_{+1}(g_{\mathbf{w}}(\mathbf{x}), y)$ are finite for all $(\mathbf{x}, y) \in \boldsymbol{\phi}(\mathcal{X}) \times \mathcal{Y}$.

**Lemma 4.5.** *Let $(\mathbf{w}, \dot{\mathbf{x}}) \in \mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n$ be arbitrarily given. Under Assumptions 4.2 and 4.3, the matrix $\mathbf{J}_{\mathbf{r}}^{(1)}(\mathbf{w}, \dot{\mathbf{x}})$ is symmetric positive semi-definite (but not positive definite) for $\mathbf{D}$ defined as in Equation 4.9.*

**Lemma 4.6.** *Let $(\mathbf{w}, \dot{\mathbf{x}}) \in \mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n$ be arbitrarily given. Under Assumptions 4.2 and 4.3, the matrix $\mathbf{J}_{\mathbf{r}}^{(2)}(\mathbf{w}, \dot{\mathbf{x}})$ is positive definite for $\mathbf{D}$ defined as in Equation 4.9.*

**Lemma 4.7.** *Let* $(\mathbf{w}, \dot{\mathbf{x}}) \in \mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n$ *be arbitrarily given. Under Assumptions 4.2 and 4.3, the matrix* $\mathbf{J}_{\mathbf{r}}^{(3)}(\mathbf{w}, \dot{\mathbf{x}})$ *is positive semi-definite for* $\mathbf{D}$ *defined as in Equation 4.9.*

The previous lemmas, whose proofs can be found in the appendix, guarantee the existence and uniqueness of a Nash equilibrium under the stated assumptions.

**Theorem 4.8.** *Let Assumptions 4.2 and 4.3 hold. Then the Nash prediction game in* (4.1) *has a unique equilibrium.*

*Proof.* The existence of an equilibrium of the Nash prediction game in (4.1) follows from Lemma 4.1. Proposition 4.4 and Lemma 4.5 to 4.7 imply that there is a positive diagonal matrix $\mathbf{D}$ so that $\mathbf{J}_{\mathbf{r}}(\mathbf{w}, \dot{\mathbf{x}})$ is positive definite for all $(\mathbf{w}, \dot{\mathbf{x}}) \in \mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n$. Hence, the uniqueness follows from Lemma 4.2. □

## 4.2 Finding the Unique Nash Equilibrium

According to Theorem 4.8, a unique equilibrium of the Nash prediction game in (4.1) exists for suitable loss functions and regularizers. To find this equilibrium, we derive and study two distinct methods: The first is based on the Nikaido-Isoda function that is constructed so that a minimax solution of this function is an equilibrium of the Nash prediction game and vice versa. This problem is then solved by inexact linesearch. In the second approach, we reformulate the Nash prediction game into a variational inequality problem which is solved by a modified extragradient method.

The data generator's action of transforming the input distribution manifests in a concatenation of transformed training instances $\dot{\mathbf{x}} \in \boldsymbol{\phi}(\mathcal{X})^n$ mapped into the feature space $\dot{\mathbf{x}}_i := \boldsymbol{\phi}(\dot{x}_i)$ for $i = 1, \ldots, n$, and the learner's action is to choose weight vector $\mathbf{w} \in \mathcal{W}$ of classifier $h_{\mathbf{w}}(x) = \operatorname{sign} g_{\mathbf{w}}(x)$ with $g_{\mathbf{w}}(x) = \mathbf{w}^\mathsf{T} \boldsymbol{\phi}(x)$.

### 4.2.1 An Inexact Linesearch Approach

To solve for a Nash equilibrium, we again consider the game from (4.2) with one learner and $n$ data generators. A solution of this game can be identified with the help of the weighted *Nikaido-Isoda* function (Equation 4.11). For any two combinations of actions $(\mathbf{w}, \dot{\mathbf{x}}) \in \mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n$ and $(\mathbf{w}', \dot{\mathbf{x}}') \in \mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n$ with $\dot{\mathbf{x}} = \left[\dot{\mathbf{x}}_1^\mathsf{T}, \ldots, \dot{\mathbf{x}}_n^\mathsf{T}\right]^\mathsf{T}$ and $\dot{\mathbf{x}}' = \left[\dot{\mathbf{x}}_1'^\mathsf{T}, \ldots, \dot{\mathbf{x}}_n'^\mathsf{T}\right]^\mathsf{T}$, this function is the weighted sum of relative cost savings that the $n+1$ players can enjoy by changing from strategy $\mathbf{w}$ to $\mathbf{w}'$ and $\dot{\mathbf{x}}_i$ to $\dot{\mathbf{x}}_i'$, respectively, while the other players continue to play according to $(\mathbf{w}, \dot{\mathbf{x}})$, that is,

$$\vartheta_{\mathbf{r}}(\mathbf{w}, \dot{\mathbf{x}}, \mathbf{w}', \dot{\mathbf{x}}') := r_0 \left(\hat{\theta}_{-1}(\mathbf{w}, \dot{\mathbf{x}}) - \hat{\theta}_{-1}(\mathbf{w}', \dot{\mathbf{x}})\right) + \sum_{i=1}^{n} r_i \left(\hat{\theta}_{+1}(\mathbf{w}, \dot{\mathbf{x}}) - \hat{\theta}_{+1}(\mathbf{w}, \dot{\mathbf{x}}^{(i)})\right), \quad (4.11)$$

where $\dot{\mathbf{x}}^{(i)} := \left[\dot{\mathbf{x}}_1^\mathsf{T}, \ldots, \dot{\mathbf{x}}_i'^\mathsf{T}, \ldots, \dot{\mathbf{x}}_n^\mathsf{T}\right]^\mathsf{T}$. Let us denote the weighted sum of greatest possible cost savings with respect to any given combination of actions $(\mathbf{w}, \dot{\mathbf{x}}) \in \mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n$ by

$$\bar{\vartheta}_{\mathbf{r}}(\mathbf{w}, \dot{\mathbf{x}}) := \max_{(\mathbf{w}', \dot{\mathbf{x}}') \in \mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n} \vartheta_{\mathbf{r}}(\mathbf{w}, \dot{\mathbf{x}}, \mathbf{w}', \dot{\mathbf{x}}'), \quad (4.12)$$

where $(\bar{\mathbf{w}}(\mathbf{w},\dot{\mathbf{x}}),\bar{\mathbf{x}}(\mathbf{w},\dot{\mathbf{x}}))$ denotes the corresponding pair of maximizers. Notice, that the maximum in (4.12) is attained for any $(\mathbf{w},\dot{\mathbf{x}})$, since $\mathcal{W}\times\boldsymbol{\phi}(\mathcal{X})^n$ is assumed to be compact and $\vartheta_{\mathbf{r}}(\mathbf{w},\dot{\mathbf{x}},\mathbf{w}',\dot{\mathbf{x}}')$ is continuous in $(\mathbf{w}',\dot{\mathbf{x}}')$.

By these definitions, a combination $(\mathbf{w}^*,\dot{\mathbf{x}}^*)$ is an equilibrium of the Nash prediction game if, and only if, $\bar{\vartheta}_{\mathbf{r}}(\mathbf{w}^*,\dot{\mathbf{x}}^*)$ is a global minimum of mapping $\bar{\vartheta}_{\mathbf{r}}$ with $\bar{\vartheta}_{\mathbf{r}}(\mathbf{w}^*,\dot{\mathbf{x}}^*)=0$ for any fixed weights $r_i>0$ and $i=0,\ldots,n$ (*cf.* Proposition 2.1(b) of [43]). Equivalently, a Nash equilibrium simultaneously satisfies both equations $\bar{\mathbf{w}}(\mathbf{w}^*,\dot{\mathbf{x}}^*)=\mathbf{w}^*$ and $\bar{\mathbf{x}}(\mathbf{w}^*,\dot{\mathbf{x}}^*)=\dot{\mathbf{x}}^*$.

The significance of this observation is that the equilibrium problem in (4.1) can be reformulated into a minimization problem of the continuous mapping $\bar{\vartheta}_{\mathbf{r}}(\mathbf{w},\dot{\mathbf{x}})$. To solve this minimization problem, we make use of Corollary 3.4 of [43]. We set the weights $r_0:=1$ and $r_i:=\frac{c_{-1,i}}{c_{+1,i}}$ for all $i=1,\ldots,n$ as in (4.9) which ensures the main condition of Corollary 3.4, that is, the positive definiteness of the Jacobian $\mathbf{J}_{\mathbf{r}}(\mathbf{w},\dot{\mathbf{x}})$ in (4.8) (*cf.* proof of Theorem 4.8). According to this corollary, vectors

$$\mathbf{d}_{-1}(\mathbf{w},\dot{\mathbf{x}}):=\bar{\mathbf{w}}(\mathbf{w},\dot{\mathbf{x}})-\mathbf{w}\quad\text{and}\quad\mathbf{d}_{+1}(\mathbf{w},\dot{\mathbf{x}}):=\bar{\mathbf{x}}(\mathbf{w},\dot{\mathbf{x}})-\dot{\mathbf{x}}$$

form a descent direction $\mathbf{d}(\mathbf{w},\dot{\mathbf{x}}):=[\mathbf{d}_{-1}(\mathbf{w},\dot{\mathbf{x}})^{\mathsf{T}},\mathbf{d}_{+1}(\mathbf{w},\dot{\mathbf{x}})^{\mathsf{T}}]^{\mathsf{T}}$ of $\bar{\vartheta}_{\mathbf{r}}(\mathbf{w},\dot{\mathbf{x}})$ at any position $(\mathbf{w},\dot{\mathbf{x}})\in\mathcal{W}\times\boldsymbol{\phi}(\mathcal{X})^n$ (except for the Nash equilibrium where $\mathbf{d}(\mathbf{w}^*,\dot{\mathbf{x}}^*)=\mathbf{0}$), and consequently, there exists $t\in[0,1]$ so that

$$\bar{\vartheta}_{\mathbf{r}}(\mathbf{w}+t\mathbf{d}_{-1}(\mathbf{w},\dot{\mathbf{x}}),\dot{\mathbf{x}}+t\mathbf{d}_{+1}(\mathbf{w},\dot{\mathbf{x}}))<\bar{\vartheta}_{\mathbf{r}}(\mathbf{w},\dot{\mathbf{x}}).$$

Since, $(\mathbf{w},\dot{\mathbf{x}})$ and $(\bar{\mathbf{w}}(\mathbf{w},\dot{\mathbf{x}}),\bar{\mathbf{w}}(\mathbf{w},\dot{\mathbf{x}}))$ are feasible combinations of actions, the convexity of the action spaces ensures that $(\mathbf{w}+t\mathbf{d}_{-1}(\mathbf{w},\dot{\mathbf{x}}),\dot{\mathbf{x}}+t\mathbf{d}_{+1}(\mathbf{w},\dot{\mathbf{x}}))$ is a feasible combination for any $t\in[0,1]$ as well. Algorithm 2 exploits these properties.

**Algorithm 2** ILS: Inexact Linesearch Solver for Nash Prediction Games

**Require:** Cost functions $\hat{\theta}_v$ as defined in (3.1) and (3.2), and action spaces $\mathcal{W}$ and $\boldsymbol{\phi}(\mathcal{X})^n$.

1: Select initial $\mathbf{w}^{(0)}\in\mathcal{W}$, set $\dot{\mathbf{x}}^{(0)}:=\mathbf{x}$, set $k:=0$, and select $\sigma\in(0,1)$ and $\beta\in(0,1)$.
2: Set $r_0:=1$ and $r_i:=\frac{c_{-1,i}}{c_{+1,i}}$ for all $i=1,\ldots,n$.
3: **repeat**
4: Set $\mathbf{d}_{-1}^{(k)}:=\bar{\mathbf{w}}^{(k)}-\mathbf{w}^{(k)}$ where $\bar{\mathbf{w}}^{(k)}:=\operatorname{argmax}_{\mathbf{w}'\in\mathcal{W}}\vartheta_{\mathbf{r}}\left(\mathbf{w}^{(k)},\dot{\mathbf{x}}^{(k)},\mathbf{w}',\dot{\mathbf{x}}^{(k)}\right)$.
5: Set $\mathbf{d}_{+1}^{(k)}:=\bar{\mathbf{x}}^{(k)}-\dot{\mathbf{x}}^{(k)}$ where $\bar{\mathbf{x}}^{(k)}:=\operatorname{argmax}_{\dot{\mathbf{x}}'\in\boldsymbol{\phi}(\mathcal{X})^n}\vartheta_{\mathbf{r}}\left(\mathbf{w}^{(k)},\dot{\mathbf{x}}^{(k)},\mathbf{w}^{(k)},\dot{\mathbf{x}}'\right)$.
6: Find maximal step size $t^{(k)}\in\left\{\beta^l\mid l\in\mathbb{N}\right\}$ with

$$\bar{\vartheta}_{\mathbf{r}}\left(\mathbf{w}^{(k)},\dot{\mathbf{x}}^{(k)}\right)-\bar{\vartheta}_{\mathbf{r}}\left(\mathbf{w}^{(k)}+t^{(k)}\mathbf{d}_{-1}^{(k)},\dot{\mathbf{x}}^{(k)}+t^{(k)}\mathbf{d}_{+1}^{(k)}\right)\geq\sigma\,t^{(k)}\left(\left\|\mathbf{d}_{-1}^{(k)}\right\|_2^2+\left\|\mathbf{d}_{+1}^{(k)}\right\|_2^2\right).$$

7: Set $\mathbf{w}^{(k+1)}:=\mathbf{w}^{(k)}+t^{(k)}\mathbf{d}_{-1}^{(k)}$.
8: Set $\dot{\mathbf{x}}^{(k+1)}:=\dot{\mathbf{x}}^{(k)}+t^{(k)}\mathbf{d}_{+1}^{(k)}$.
9: Set $k:=k+1$.
10: **until** $\left\|\mathbf{w}^{(k)}-\mathbf{w}^{(k-1)}\right\|_2^2+\left\|\dot{\mathbf{x}}^{(k)}-\dot{\mathbf{x}}^{(k-1)}\right\|_2^2\leq\epsilon$.

### 4.2.2 A Modified Extragradient Approach

In Algorithm 2, line 4 and 5, as well as the linesearch in line 6, require to solve a concave maximization problem within each iteration. As this may become computationally demanding, we derive a second approach based on extragradient descent. Therefore, instead of reformulating the equilibrium problem into a minimax problem, we directly address the first-order optimality conditions of each players' minimization problem in (4.2). Under Assumption 4.2, a combination of actions $(\mathbf{w}^*, \dot{\mathbf{x}}^*)$ with $\dot{\mathbf{x}}^* = \left[\dot{\mathbf{x}}_1^{*\mathsf{T}}, \ldots, \dot{\mathbf{x}}_n^{*\mathsf{T}}\right]^\mathsf{T}$ satisfies each player's first-order optimality conditions if, and only if, for all $(\mathbf{w}, \dot{\mathbf{x}}) \in \mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n$ the following inequalities hold.

$$\begin{aligned} \nabla_{\mathbf{w}} \hat{\theta}_{-1}(\mathbf{w}^*, \dot{\mathbf{x}}^*)^\mathsf{T}(\mathbf{w} - \mathbf{w}^*) &\geq 0 \\ \nabla_{\dot{\mathbf{x}}_i} \hat{\theta}_{+1}(\mathbf{w}^*, \dot{\mathbf{x}}^*)^\mathsf{T}(\dot{\mathbf{x}}_i - \dot{\mathbf{x}}_i^*) &\geq 0 \quad \forall\, i = 1, \ldots, n \end{aligned}$$

As the joint action space of all players $\mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n$ is precisely the full Cartesian product of the learner's action set $\mathcal{W}$ and the $n$ data generators' action sets $\boldsymbol{\phi}(\mathcal{X})$, the (weighted) sum of those individual optimality conditions is also a sufficient and necessary optimality condition for the equilibrium problem. Hence, a Nash equilibrium $(\mathbf{w}^*, \dot{\mathbf{x}}^*) \in \mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n$ is a solution of the *variational inequality problem*,

$$\mathbf{g}_\mathbf{r}(\mathbf{w}^*, \dot{\mathbf{x}}^*)^\mathsf{T} \left( \begin{bmatrix} \mathbf{w} \\ \dot{\mathbf{x}} \end{bmatrix} - \begin{bmatrix} \mathbf{w}^* \\ \dot{\mathbf{x}}^* \end{bmatrix} \right) \geq 0 \quad \forall\, (\mathbf{w}, \dot{\mathbf{x}}) \in \mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n \tag{4.13}$$

and vice versa (*cf.* Proposition 7.1 of [40]). The pseudo-gradient $\mathbf{g}_\mathbf{r}$ in (4.13) is defined as in (4.3) with fixed vector $\mathbf{r} = [r_0, r_1, \ldots, r_n]^\mathsf{T}$ where $r_0 := 1$ and $r_i := \frac{c_{-1,i}}{c_{+1,i}}$ for all $i = 1, \ldots, n$ (*cf.* Equation 4.9). Under Assumption 4.3, this choice of $\mathbf{r}$ ensures that the mapping $\mathbf{g}_\mathbf{r}(\mathbf{w}, \dot{\mathbf{x}})$ is continuous and strictly monotone (*cf.* proof of Lemma 4.2 and Theorem 4.8). Hence, the variational inequality problem in (4.13) can be solved by *modified extragradient descent* (see, for instance, Chapter 7.2.3 of [34]).

Algorithm 3 states an iterative extragradient method which—apart from back projection steps—does not require to solve an optimization problem in each iteration. $\Pi_{\mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n}$ denotes the $L^2$-projection operator defined in (2.36). It performs a Euclidean projection of its argument into the joint action space $\mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n$. The proposed algorithm converges to a solution of the variational inequality problem in 4.13, *i.e.,* the unique equilibrium of the Nash prediction game, if Assumptions 4.2 and 4.3 hold (*cf.* Theorem 7.40 of [34]).

## 4.3 Applying Kernels

So far, we assumed the knowledge of feature mapping $\boldsymbol{\phi} : \mathcal{X} \to \boldsymbol{\phi}(\mathcal{X})$ so that we can compute an explicit feature representation $\boldsymbol{\phi}(x_i)$ of the training instances $x_i$ for all $i = 1, \ldots, n$. We now turn to a discussion on the applicability of a general kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which measures the similarity between two instances. As discussed in Section 2.6, kernel function $k$ is assumed to be a positive semi-definite function so that it can be stated in

**Algorithm 3** EDS: Extragradient Descent Solver for Nash Prediction Games

**Require:** Cost functions $\hat{\theta}_v$ as defined in (3.1) and (3.2), and action spaces $\mathcal{W}$ and $\boldsymbol{\phi}(\mathcal{X})^n$.

1: Select initial $\mathbf{w}^{(0)} \in \mathcal{W}$, set $\dot{\mathbf{x}}^{(0)} := \mathbf{x}$, set $k := 0$, and select $\sigma \in (0,1)$ and $\beta \in (0,1)$.

2: Set $r_0 := 1$ and $r_i := \frac{c_{-1,i}}{c_{+1,i}}$ for all $i = 1, \ldots, n$.

3: **repeat**

4: Set $\begin{bmatrix} \mathbf{d}_{-1}^{(k)} \\ \mathbf{d}_{+1}^{(k)} \end{bmatrix} := \Pi_{\mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n} \left( \begin{bmatrix} \mathbf{w}^{(k)} \\ \dot{\mathbf{x}}^{(k)} \end{bmatrix} - \mathbf{g}_{\mathbf{r}} \left( \mathbf{w}^{(k)}, \dot{\mathbf{x}}^{(k)} \right) \right) - \begin{bmatrix} \mathbf{w}^{(k)} \\ \dot{\mathbf{x}}^{(k)} \end{bmatrix}$.

5: Find maximal step size $t^{(k)} \in \left\{ \beta^l \mid l \in \mathbb{N} \right\}$ with

$$-\mathbf{g}_{\mathbf{r}} \left( \mathbf{w}^{(k)} + t^{(k)} \mathbf{d}_{-1}^{(k)}, \dot{\mathbf{x}}^{(k)} + t^{(k)} \mathbf{d}_{+1}^{(k)} \right)^{\mathsf{T}} \begin{bmatrix} \mathbf{d}_{-1}^{(k)} \\ \mathbf{d}_{+1}^{(k)} \end{bmatrix} \geq \sigma \left( \left\| \mathbf{d}_{-1}^{(k)} \right\|_2^2 + \left\| \mathbf{d}_{+1}^{(k)} \right\|_2^2 \right).$$

6: Set $\begin{bmatrix} \bar{\mathbf{w}}^{(k)} \\ \bar{\mathbf{x}}^{(k)} \end{bmatrix} := \begin{bmatrix} \mathbf{w}^{(k)} \\ \dot{\mathbf{x}}^{(k)} \end{bmatrix} + t^{(k)} \begin{bmatrix} \mathbf{d}_{-1}^{(k)} \\ \mathbf{d}_{+1}^{(k)} \end{bmatrix}$.

7: Set step size of extragradient

$$\gamma^{(k)} := -\frac{t^{(k)}}{\left\| \mathbf{g}_{\mathbf{r}} \left( \bar{\mathbf{w}}^{(k)}, \bar{\mathbf{x}}^{(k)} \right) \right\|_2^2} \mathbf{g}_{\mathbf{r}} \left( \bar{\mathbf{w}}^{(k)}, \bar{\mathbf{x}}^{(k)} \right)^{\mathsf{T}} \begin{bmatrix} \mathbf{d}_{-1}^{(k)} \\ \mathbf{d}_{+1}^{(k)} \end{bmatrix}.$$

8: Set $\begin{bmatrix} \mathbf{w}^{(k+1)} \\ \dot{\mathbf{x}}^{(k+1)} \end{bmatrix} := \Pi_{\mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n} \left( \begin{bmatrix} \mathbf{w}^{(k)} \\ \dot{\mathbf{x}}^{(k)} \end{bmatrix} - \gamma^{(k)} \mathbf{g}_{\mathbf{r}} \left( \bar{\mathbf{w}}^{(k)}, \bar{\mathbf{x}}^{(k)} \right) \right)$.

9: Set $k := k + 1$.

10: **until** $\left\| \mathbf{w}^{(k)} - \mathbf{w}^{(k-1)} \right\|_2^2 + \left\| \dot{\mathbf{x}}^{(k)} - \dot{\mathbf{x}}^{(k-1)} \right\|_2^2 \leq \epsilon$.

terms of an inner product in the corresponding reproducing kernel Hilbert space, that is, $\exists \boldsymbol{\phi}$ with $k(x, x') = \boldsymbol{\phi}(x)^{\mathsf{T}} \boldsymbol{\phi}(x')$ for all $x, x' \in \mathcal{X}$.

To derive a kernelized formulation of the Nash prediction game, we assume that the transformed instances lie in the span of the mapped training instances. We restrict the data generator's action space so that the transformed instances $\dot{x}_i$ are mapped into the same subspace of the reproducing kernel Hilbert space as the unmodified training instances $x_i$. Under this restriction, the weight vector $\mathbf{w} \in \mathcal{W}$ and the transformed instances $\boldsymbol{\phi}(\dot{x}_i) \in \boldsymbol{\phi}(\mathcal{X})$ for $i = 1, \ldots, n$ can be expressed as linear combinations of the mapped training instances, *i.e.,* $\exists \alpha_i, \Xi_{ij}$, so that

$$\mathbf{w} = \sum_{i=1}^{n} \omega_i \boldsymbol{\phi}(x_i) \quad \text{and} \quad \boldsymbol{\phi}(\dot{x}_j) = \sum_{i=1}^{n} \Xi_{ij} \boldsymbol{\phi}(x_i) \quad \forall\, j = 1, \ldots, n.$$

Further, let us assume that the action spaces $\mathcal{W}$ and $\boldsymbol{\phi}(\mathcal{X})^n$ can be adequately translated into dual action spaces $\mathcal{A} \subset \mathbb{R}^n$ and $\mathcal{Z} \subset \mathbb{R}^{n \times n}$, which is possible, for instance, if $\mathcal{W}$ and $\boldsymbol{\phi}(\mathcal{X})^n$ are closed $l^2$-balls. Then, a kernelized variant of the Nash prediction game is obtained by inserting the above equations into the players' cost functions in (3.1) and (3.2)

with regularizers in (4.16) and (4.17),

$$\hat{\theta}_{-1}(\boldsymbol{\omega}, \mathbf{B}) = \sum_{i=1}^{n} c_{-1,i}\ell_{-1}(\boldsymbol{\omega}^\mathsf{T}\mathbf{K}\mathbf{B}\mathbf{e}_i, y_i) + \rho_{-1}\frac{1}{2}\boldsymbol{\omega}^\mathsf{T}\mathbf{K}\boldsymbol{\omega}, \tag{4.14}$$

$$\hat{\theta}_{+1}(\boldsymbol{\omega}, \mathbf{B}) = \sum_{i=1}^{n} c_{+1,i}\ell_{+1}(\boldsymbol{\omega}^\mathsf{T}\mathbf{K}\mathbf{B}\mathbf{e}_i, y_i) + \rho_{+1}\frac{1}{2n}\mathrm{tr}\left((\mathbf{B}-\mathbf{I}_n)^\mathsf{T}\mathbf{K}(\mathbf{B}-\mathbf{I}_n)\right), \tag{4.15}$$

where $\mathbf{e}_i \in \{0,1\}^n$ is the $i$-th unit vector, $\boldsymbol{\omega} \in \mathcal{A}$ is the dual weight vector, $\mathbf{B} \in \mathcal{Z}$ is the dual transformed data matrix, and $\mathbf{K} \in \mathbb{R}^{n\times n}$ is the kernel matrix with $K_{ij} := k(x_i, x_j)$. In the dual Nash prediction game with cost functions (4.14) and (4.15), the learner chooses the dual weight vector $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_n]^\mathsf{T}$ and classifies a new instance $x$ by $h_{\boldsymbol{\omega}}(x) = \operatorname{sign} g_{\boldsymbol{\omega}}(x)$ with $g_{\boldsymbol{\omega}}(x) = \sum_{i=1}^{n} \omega_i k(x_i, x)$. In contrast, the data generator chooses the dual transformed data matrix $\mathbf{B}$ which implicitly reflects the change of the training distribution. Their transformation costs are in proportion to the deviation of $\mathbf{B}$ from the identity matrix $\mathbf{I}_n$, and if $\mathbf{B}$ equals $\mathbf{I}_n$, the learner's task reduces to standard kernelized empirical risk minimization. The proposed Algorithms 2 and 3 can be readily applied when replacing $\mathbf{w}$ by $\boldsymbol{\omega}$ and $\dot{\mathbf{x}}_i$ by $\mathbf{B}\mathbf{e}_i$ for all $i = 1, \ldots, n$.

An alternative approach to a kernelization of the Nash prediction game is to first construct an explicit feature representation, such as discussed in Section 2.6 and then to train the Nash model by using Algorithms 2 or 3 together with the constructed feature mapping, for instance, the kernel PCA mapping (*cf.* Equation 2.30). Here, we again assume that the transformed instances $\boldsymbol{\phi}(\dot{x}_i)$ as well as the weight vector $\mathbf{w}$ lie in the span of the mapped training instances $\boldsymbol{\phi}(x)$.

## 4.4 Instances of the Nash Prediction Game

In this section we present two instances of the Nash prediction game and investigate under which conditions those games possess unique Nash equilibria. We start by specifying both players' loss functions and regularizers.

An obvious choice for the loss function of the learner $\ell_{-1}(z, y)$ is the *zero-one loss* given in Definition 2.1. A possible choice for the data generator's loss is $\ell^{0/1}(z, -1)$ which penalizes positive decision values $z$, independently of the class label. This choice accounts for the fact, that the data generator experiences costs when the predictive model blocks an event, that is, assigns an instance to the positive class. For instance, a legitimate email sender experiences costs when a legitimate email is erroneously blocked just like an abusive sender, also amalgamated into the data generator, experiences costs when spam messages are blocked. However, the zero-one loss violates Assumption 4.2 as it is neither convex nor twice-continuously differentiable. In the following sections, we therefore approximate the zero-one loss by the *logistic loss* and the *trigonometric loss* which both satisfy Assumption 4.2.

To regularize the players' actions, recall that $\Omega_{+1}(D, \dot{D})$ is an estimate of the *transformation cost* that the data generator incurs when shifting the training distribution—where the training instances $x_i$ are drawn from—to the test distribution, which is empirically represented by the transformed training instances $\dot{x}_i$. In our analysis, we approximate these costs by the average squared $\ell^2$-distance between $x_i$ and $\dot{x}_i$ in the feature space induced by mapping $\boldsymbol{\phi}$, that is,

$$\Omega_{+1}(D, \dot{D}) := \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \left\| \boldsymbol{\phi}(\dot{x}_i) - \boldsymbol{\phi}(x_i) \right\|_2^2 . \tag{4.16}$$

Geometrically, this choice assumes an adversarial data generator who is allowed to *shift* each single training instance from $\boldsymbol{\phi}(x_i)$ to $\boldsymbol{\phi}(\dot{x}_i) := \boldsymbol{\phi}(x_i) + \mathbf{a}_i$. The corresponding transformation cost induced by the $i$-th instance are in proportion to the squared length of the offset vector $\mathbf{a}_i$, that is, all points of the surface of a hypersphere of radius $r_i := \|\mathbf{a}_i\|_2$ centered at $\boldsymbol{\phi}(x_i)$ raise identical transformation cost $\propto r_i^2$. The amount of transformation that the data generator is allowed to perform is implicitly restricted by the regularization parameter $\rho_{+1}$.

The learner's regularizer $\Omega_{-1}(\mathbf{w})$ penalizes the complexity of the predictive model $h(x) = \operatorname{sign} g_{\mathbf{w}}(x)$. As discussed before in Section 2.5, we consider Tikhonov regularization which reduces to the squared $\ell^2$-norm

$$\Omega_{-1}(\mathbf{w}) := \frac{1}{2} \|\mathbf{w}\|_2^2 \tag{4.17}$$

of weight vector $\mathbf{w}$. The learner's regularization parameter is denoted by $\rho_{-1}$.

### 4.4.1 Nash Logistic Regression

In this section we study the particular instance of the Nash prediction game where each players' loss function rests on the negative logarithm of the logistic function $\sigma(a) := \frac{1}{1+\exp(-a)}$, that is, the *logistic loss* (*cf.* Definition 2.6),

$$\ell^{\mathrm{l}}(z, y) := -\log \sigma(yz) = \log\left(1 + \exp(-yz)\right). \tag{4.18}$$

We consider the regularizers in (4.16) and (4.17), respectively, which give rise to the following definition of the *Nash logistic regression*.

**Definition 4.1.** *The* Nash logistic regression *(NLR) is an instance of the Nash prediction game with non-empty, compact, and convex action spaces $\mathcal{W} \subset \mathbb{R}^m$ and $\boldsymbol{\phi}(\mathcal{X})^n \subset \mathbb{R}^{m \cdot n}$ and cost functions*

$$\begin{aligned} \hat{\theta}^{\mathrm{l}}_{-1}(\mathbf{w}, \dot{\mathbf{x}}) &:= \sum_{i=1}^{n} c_{-1,i} \ell^{\mathrm{l}}(\mathbf{w}^{\mathsf{T}} \dot{\mathbf{x}}_i, y_i) + \rho_{-1} \frac{1}{2} \|\mathbf{w}\|_2^2 \\ \hat{\theta}^{\mathrm{l}}_{+1}(\mathbf{w}, \dot{\mathbf{x}}) &:= \sum_{i=1}^{n} c_{+1,i} \ell^{\mathrm{l}}(\mathbf{w}^{\mathsf{T}} \dot{\mathbf{x}}_i, -1) + \rho_{+1} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \|\dot{\mathbf{x}}_i - \mathbf{x}_i\|_2^2 \end{aligned}$$

*where $\ell^{\mathrm{l}}$ is specified in* (4.18).

In the above definition, column vectors $\mathbf{x} := [\mathbf{x}_1^\mathsf{T}, \ldots, \mathbf{x}_n^\mathsf{T}]^\mathsf{T}$ and $\dot{\mathbf{x}} := [\dot{\mathbf{x}}_1^\mathsf{T}, \ldots, \dot{\mathbf{x}}_n^\mathsf{T}]^\mathsf{T}$ again denote the concatenation of the original and the transformed training instances, respectively, which are mapped into the feature space by $\mathbf{x}_i := \boldsymbol{\phi}(x_i)$ and $\dot{\mathbf{x}}_i := \boldsymbol{\phi}(\dot{x}_i)$.

As in our introductory discussion, the data generator's loss function $\ell_{+1}(z, y) := \ell^{\mathrm{l}}(z, -1)$ penalizes positive decision values independently of the class label $y$. In contrast, instances that pass the classifier, *i.e.,* instances with negative decision values, incur little or almost no costs. By the above definition, the Nash logistic regression obviously satisfies Assumption 4.2, and according to the following corollary, also satisfies Assumption 4.3 for suitable regularization parameters. The proof of the corollary can be found in the appendix.

**Corollary 4.9.** *Let the Nash logistic regression be specified as in Definition 4.1 with positive regularization parameters $\rho_{-1}$ and $\rho_{+1}$ which satisfy*

$$\rho_{-1}\rho_{+1} \geq n\mathbf{c}_{-1}^\mathsf{T}\mathbf{c}_{+1}, \tag{4.19}$$

*then Assumption 4.2 and 4.3 hold, and consequently, the Nash logistic regression possess a unique Nash equilibrium.*

Recall, that the weighting factors $c_{v,i}$ are strictly positive with $\sum_{i=1}^n c_{v,i} = 1$ for both players $v$. In particular, it therefore follows that in the unweighted case where $c_{v,i} = \frac{1}{n}$ for all $i = 1, \ldots, n$ and $v \in V$, a sufficient condition to ensure the existence of a unique Nash equilibrium is to set the learner's regularization parameter to $\rho_{-1} \geq \frac{1}{\rho_{+1}}$.

### 4.4.2 Nash Support Vector Machine

The Nash logistic regression tends to non-sparse solutions. This becomes particularly apparent if the Nash equilibrium $(\mathbf{w}^*, \dot{\mathbf{x}}^*)$ is an interior point of the joined action set $\mathcal{W} \times \boldsymbol{\phi}(\mathcal{X})^n$ in which case the (partial) gradients in (4.4) and (4.5) are zero at $(\mathbf{w}^*, \dot{\mathbf{x}}^*)$. For regularizer (4.17), this implies that $\mathbf{w}^*$ is a linear combination of the transformed instances $\dot{\mathbf{x}}_i$ where all weighting factors are non-zero, since the first derivative of the logistic loss as well as the cost factors $c_{-1,i}$ are non-zero for all $i = 1, \ldots, n$. The *support vector machine* (SVM), which employs the *hinge loss* (*cf.* Definition 2.2), does not suffer from non-sparsity. However, the hinge loss obviously violates Assumption 4.2 as it is not twice-continuously differentiable. Therefore, we make use of the *trigonometric loss* (*cf.* Definition 2.4),

$$\ell^t(g_{\mathbf{w}}(x), y) := \begin{cases} 0 & \text{if} \quad yg_{\mathbf{w}}(x) > \delta \\ \frac{1}{2}(\delta - yg_{\mathbf{w}}(x)) - \frac{\delta}{\pi}\cos\left(\frac{\pi}{2\delta}yg_{\mathbf{w}}(x)\right) & \text{if} \quad |yg_{\mathbf{w}}(x)| \leq \delta \\ -yg_{\mathbf{w}}(x) & \text{if} \quad yg_{\mathbf{w}}(x) < -\delta \end{cases}, \tag{4.20}$$

which is convex and twice-continuously differentiable according to Proposition 2.3. Because of the similarities of the hinge loss and the trigonometric loss, we call the Nash prediction game that is based upon the trigonometric loss *Nash support vector machine* (NSVM) where we again consider the regularizers in (4.16) and (4.17).

**Definition 4.2.** *The* Nash support vector machine *(NSVM) is an instance of the Nash prediction game with non-empty, compact, and convex action spaces $\mathcal{W} \subset \mathbb{R}^m$ and $\boldsymbol{\phi}(\mathcal{X})^n \subset \mathbb{R}^{m \cdot n}$ and cost functions*

$$\begin{aligned} \hat{\theta}^{\mathrm{t}}_{-1}(\mathbf{w}, \dot{\mathbf{x}}) &:= \sum_{i=1}^{n} c_{-1,i} \ell^{\mathrm{t}}(\mathbf{w}^{\mathsf{T}} \dot{\mathbf{x}}_i, y_i) + \rho_{-1} \frac{1}{2} \|\mathbf{w}\|_2^2 \\ \hat{\theta}^{\mathrm{t}}_{+1}(\mathbf{w}, \dot{\mathbf{x}}) &:= \sum_{i=1}^{n} c_{+1,i} \ell^{\mathrm{t}}(\mathbf{w}^{\mathsf{T}} \dot{\mathbf{x}}_i, -1) + \rho_{+1} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \|\dot{\mathbf{x}}_i - \mathbf{x}_i\|_2^2 \end{aligned}$$

*where $\ell^{\mathrm{t}}$ is specified in* (4.20).

The following corollary states sufficient conditions under which the Nash support vector machine satisfies Assumptions 4.2 and 4.3, and consequently has a unique Nash equilibrium.

**Corollary 4.10.** *Let the Nash support vector machine be specified as in Definition 4.2 with positive regularization parameters $\rho_{-1}$ and $\rho_{+1}$ which satisfy*

$$\rho_{-1} \rho_{+1} > n \mathbf{c}_{-1}^{\mathsf{T}} \mathbf{c}_{+1}, \tag{4.21}$$

*then Assumptions 4.2 and 4.3 hold, and consequently, the Nash support vector machine has a unique Nash equilibrium.*

The proof of the corollary can be found in the appendix. The only difference between the uniqueness conditions for NLR and NSVM, is that the above bound is strict for the NSVM.

## 4.5 Related Work

In this chapter, we formulate the problem of learning an adversary-aware predictive model from data as a static two-player game. So far, mainly the special case of a zero-sum prediction game has been investigated. This setting is closely related to a Stackelberg game for which reason we discuss methods which rest upon zero-sum games in Section 5.5. The only work [53] which explicitly considers a Nash-optimal model-building focuses on antagonistic loss functions. As in this case, the Nash game reduces to a minimax problem, the authors' proposed model indeed reflects a worst-case setting which is identical to the problem of the Invar-SVM [71] (*cf.* Section 5.5).

Few authors study related problems such as repeated games in the context of learning adversary-aware models. Androutsopoulos et al. [2], for instance, address the problem where the data instances as well as the predictive model, except for some threshold parameter, are fixed. The interacting players are the user of the predictive model who decides for the threshold, and the data generator who has to decide whether to choose a positive or a negative instance. This model is discussed in the context of email spam filtering where the spam senders control the ratio of spam and the recipients decide whether to read or to disregard a message purely based on the prediction of the spam filter. The authors

derive the Nash equilibrium for this game, *i.e.,* the spam ratio and threshold value which is Nash-optimal for both players.

The solution concept of Nash has been applied to many other areas such as power markets [22], wireless network design [20], network security [62], and telecommunication [1]. All of these applications require efficient methods (see, *e.g.,* [30, 31, 42, 43]) to solve the (generalized) Nash equilibrium problem. A detailed survey on such methods can be found, for instance, in [32].

## 4.6 Empirical Evaluation

The goal of this section is to explore the relative strengths and weaknesses of the discussed instances of the Nash prediction game and existing baseline methods in the context of email spam filtering. We compare the regularized empirical risk minimizers *logistic regression* (LR) and the *support vector machine* (SVM), the worst-case solution *SVM for invariances* with feature removal (Invar-SVM, *cf.* [71]), and the proposed Nash prediction games *Nash logistic regression* (NLR) and the *Nash support vector machine* (NSVM).

We use four corpora of chronologically sorted emails detailed in Table 4.1: The first data set contains emails of an email service provider (ESP) collected between 2007 and 2010. The second (Mailinglist) is a collection of emails from publicly available IT-related mailing lists augmented by spam emails from Bruce Guenter's spam trap[1] of the same time period. The third corpus (Private) contains newsletters as well as spam and non-spam emails of the authors. The last corpus is the NIST TREC 2007 spam corpus [23]. The rate of spam emails in the data sets varies between 65% and 77%.

**Table 4.1:** Data sets used in the experiments.

| *data set* | *instances* | *features* | *delivery period* |
|---|---|---|---|
| ESP | 169,612 | 541,713 | 01/06/2007 - 27/04/2010 |
| Mailinglist | 128,117 | 266,378 | 01/04/1999 - 31/05/2006 |
| Private | 108,178 | 582,100 | 01/08/2005 - 31/03/2010 |
| TREC 2007 | 75,496 | 214,839 | 04/08/2007 - 07/06/2007 |

The feature mapping $\boldsymbol{\phi}(x)$ is defined as follows: Email $x \in \mathcal{X}$ is first tokenized with the X-tokenizer [15, 67] and converted into the $m$-dimensional binary bag-of-word vector $\mathbf{x} := [0, 1]^m$. The value of $m$ is determined by the number of distinct terms in the data set where we have removed all terms which occur only once. For each experiment and each repetition, we then construct the PCA mapping (2.30) with respect to the corresponding $n$ training emails using the linear kernel $k(\mathbf{x}, \mathbf{x}') := \mathbf{x}^\top \mathbf{x}'$, resulting in $n$-dimensional training instances $\boldsymbol{\phi}_{\text{PCA}}(x_i) \in \mathbb{R}^n$ for $i = 1, \ldots, n$.

[1] `http://untroubled.org/spam/`

To ensure the convexity as well as the compactness requirement in Assumption 4.2, we notionally restrict the players' action sets by $\phi(\mathcal{X}) := \{\phi_{\text{PCA}}(x) \in \mathbb{R}^n \mid \|\phi_{\text{PCA}}(x)\|_2^2 \leq \kappa\}$ and $\mathcal{W} := \{\mathbf{w} \in \mathbb{R}^n \mid \|\mathbf{w}\|_2^2 \leq \kappa\}$ for some fixed constant $\kappa$. Note, that by choosing an arbitrarily large $\kappa$, the players' action sets become effectively unbounded.

For both proposed algorithms, ILS and EDS, we set $\sigma := 0.001$, $\beta := 0.2$, and $\epsilon := 10^{-14}$. The algorithms are stopped if $l$ exceeds 30 in line 6 of ILS and line 5 of EDS, respectively. In this case, no convergence is achieved. In all experiments, we use the F-measure—that is, the harmonic mean of precision and recall with positive class *spam*—as evaluation measure and tune all parameters with respect to likelihood. The particular protocol and results of each experiment are detailed in the following sections.

### 4.6.1 Convergence

Corollaries 4.9 (for Nash logistic regression) and 4.10 (for the Nash support vector machine) specify necessary conditions on the regularization parameters $\rho_{-1}$ and $\rho_{+1}$ under which a unique Nash equilibrium exists. When this is the case, both the ILS and EDS algorithms will converge on that Nash equilibrium. In the first set of experiments, we study whether repeated restarts of the algorithm converge on the same equilibrium when the bounds in Equations 4.19 and 4.21 are satisfied, and when they are violated to increasingly large degrees. To satisfy the condition $\sum_i c_{v,i} = 1$, we set $c_{v,i} := \frac{1}{n}$ for $v \in V = \{-1, +1\}$ and $i = 1, \ldots, n$. For these cost factors and for $\rho_{-1} > \frac{1}{\rho_{+1}}$, both bounds (Equations 4.19 and 4.21) are satisfied.

For each value of $\rho_{-1}$ and $\rho_{+1}$ and each of 10 repetitions, we randomly draw 400 emails from the data set and run EDS with a randomly chosen initial solution $(\mathbf{w}^{(0)}, \dot{\mathbf{x}}^{(0)})$ until convergence. We run ILS on the same training set. In each repetition we randomly choose a distinct initial solution, and after each iteration $k$ we compute the Euclidean distance between the EDS solution and the current ILS iterate $\mathbf{w}^{(k)}$. Figure 4.1 reports on these average Euclidean distances between distinctly initialized runs. The blue curves ($\rho_{-1} = 2\frac{1}{\rho_{+1}}$) satisfy Equations 4.19 and 4.21, the yellow curves ($\rho_{-1} = \frac{1}{\rho_{+1}}$) lie exactly on the boundary, and all other curves violate the bounds. Dotted lines show the Euclidean distance between the Nash equilibrium and the solution of logistic regression.

Our findings are as follows. Logistic regression and the regular SVM never coincide with the Nash equilibrium—the Euclidean distances lie in the range between $10^{-2}$ and 2. ILS and EDS always converge to identical equilibria when (4.19) and (4.21) are satisfied (blue and yellow curves). The Euclidean distances lie at the threshold of numerical computing accuracy. When Equations 4.19 and 4.21 are violated by a factor up to 4 (turquoise and red curves), all repetitions still converge on the same equilibrium, indicating that the equilibrium is either still unique or a secondary equilibrium is unlikely to be found. When the bounds are violated by a factor of 8 or 16 (green and purple curves), then some repetitions of the learning algorithms do not converge or start to converge to distinct equilibria. In the latter case, learner and data generator may attain distinct equilibria and may experience an arbitrarily poor outcome when playing a Nash equilibrium.

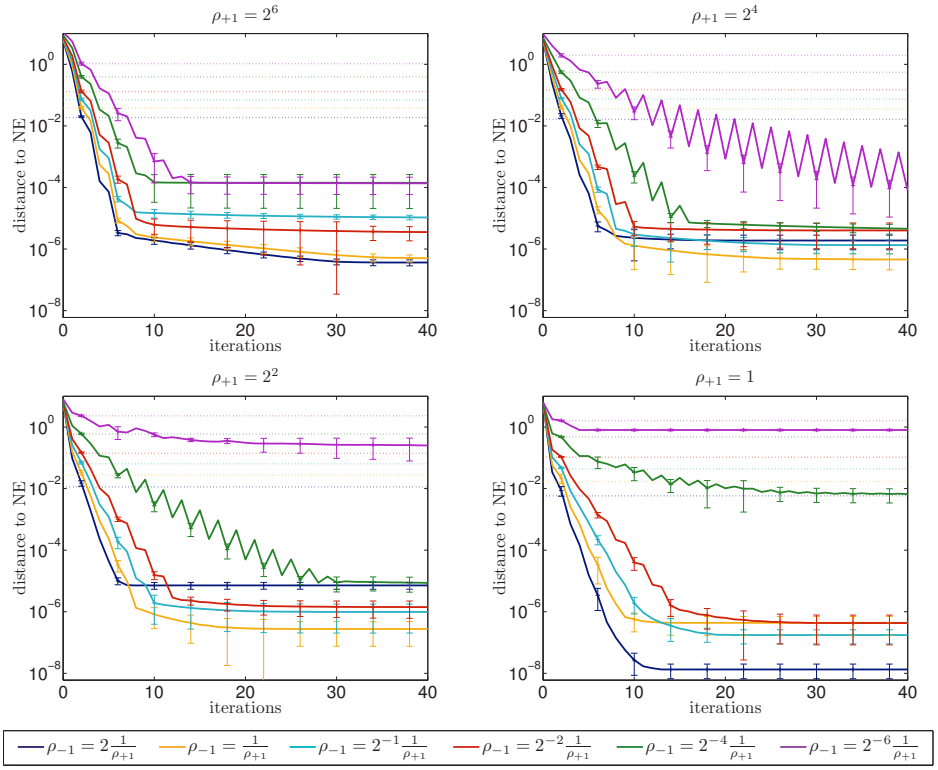


**Figure 4.1:** Average Euclidean distance (solid lines) between the EDS solution and the ILS solution at iteration $k = 0, \ldots, 40$ for Nash logistic regression on the ESP corpus. The dotted lines show the distance between the EDS solution and the solution of logistic regression. Error bars indicate standard deviation.

### 4.6.2 Regularization Parameters

The regularization parameters $\rho_v$ of the players $v \in V = \{-1, +1\}$ play a major role in the prediction game. The learner's regularizer determines the generalization ability of the predictive model and the data generator's regularizer controls the amount of change in the data generation process. In order to tune these parameters, one would need to have access to labeled data that are governed by the transformed input distribution. In our second experiment, we will explore to which extend those parameters can be estimated using a portion of the newest training data. Intuitively, the latest training data may be more similar to the test data than older training data.

We split the data set into three parts: The 2,000 oldest emails constitute the training portion, we use the next 2,000 emails as hold-out portion on which the parameters are tuned, and the remaining emails are used as test set. We randomly draw 200 spam and

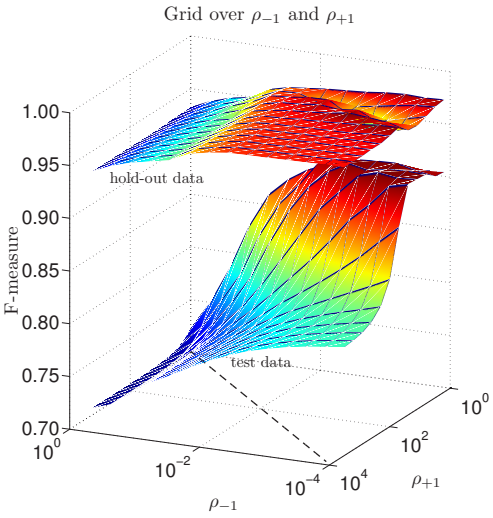


**Figure 4.2a:** Performance of NLR on the hold-out and the test data with respect to regularization parameters.

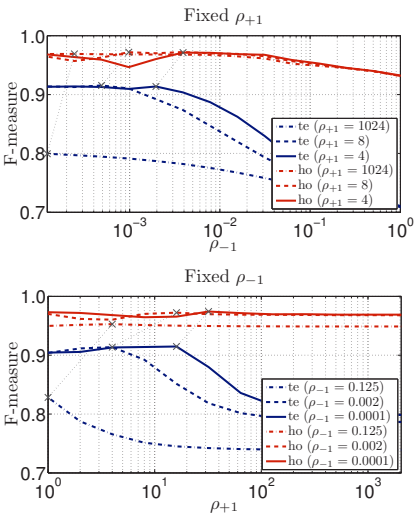


**Figure 4.2b:** Performance of NLR on the hold-out data (ho) and the test data (te) for fixed values of $\rho_v$.

200 non-spam messages from the training portion and draw another subset of 400 emails from the hold-out portion. Both NPG instances are trained on the 400 training emails and evaluated against all emails of the test portion. To tune the parameters, we conduct a grid search maximizing the likelihood on the 400 hold-out emails. We repeat this experiment 10 times for all four data sets and report on the found parameters as well as the "optimal" reference parameters according to the maximal value of F-measure on the test set. Those optimal regularization parameters are not used in later experiments. The intuition of the experiment is that the data generation process has already been changed between the oldest and the latest emails. This change may cause a distribution shift which is reflected in the hold-out portion. We expect that one can tune each players' regularization parameter by tuning with respect to this hold-out set.

In Figure 4.2a we plot the performance of the Nash logistic regression (NLR) on the hold-out and the test data against the regularization parameters $\rho_{-1}$ and $\rho_{+1}$. The dashed line visualizes the bound in (4.19) on the regularization parameters for which NLR is guaranteed to possess a unique Nash equilibrium. Figure 4.2b shows sectional views of the plot in Figure 4.2a along the $\rho_{-1}$-axis (upper diagram) and the $\rho_{+1}$-axis (lower diagram) for several values of $\rho_{+1}$ and $\rho_{-1}$, respectively. As expected, the effect of the regularization parameters on the test data is much stronger than on the hold-out data.

It turns out that the data generator's $\rho_{+1}$ has almost no impact on the value of F-measure on the hold-out data set (see lower diagram of Figure 4.2b). Hence, we conclude that estimating $\rho_{+1}$ without access to labeled data from the test distribution or additional

knowledge about the data generator is difficult for this application. The latest training data are still too different from the test data. In all remaining experiments and for all data sets we set $\rho_{+1} = 8$ for NLR and $\rho_{+1} = 2$ for NSVM, since for those choices the Nash models performed generally best on the hold-out set for a large variety of values of $\rho_{-1}$. For the Invar-SVM the regularization of the data generator's transformation is controlled explicitly by the number $K$ of modifiable attributes per positive instance. We conducted the same experiment for the Invar-SVM resulting in an optimal value of $K = 25$, *i.e.,* the data generator is allowed to remove up to 25 tokens of each spam email of the training data.

From the upper diagram of Figure 4.2b we see that estimating $\rho_{-1}$ for any fixed $\rho_{+1}$ seems possible. Even if we slightly overestimate the learner's optimal regularization parameter—to compensate for the distributional difference between the transformed training sample and the marginal shifted hold-out set—the determined value of $\rho_{-1}$ is close to the optimum for all four data sets.

### 4.6.3 Evaluation for Nash-Playing Adversary

We evaluate both, a regular classifier trained under the *i.i.d.* assumption, and the Nash-equilibrial models against an adversary who does not transform the input distribution and an adversary who executes the Nash-equilibrial transformation on the input distribution. Since we cannot be certain that actual spam senders play a Nash equilibrium, we use the following semi-artificial setting.

The learner observes a sample of 200 spam and 200 non-spam emails drawn from the training portion of the data and estimates the Nash-optimal prediction model with parameters $\dot{\mathbf{w}}$. The trivial baseline solution of regularized empirical risk minimization (*i.i.d.* baseline) is denoted by $\mathbf{w}$. The data generator observes a *distinct* sample $D$ of 200 spam and 200 non-spam messages, also drawn from the training portion, and computes their Nash-optimal response $\dot{D}$.

We again set $c_{v,i} := \frac{1}{n}$ for $v \in \{-1, +1\}$ and $i = 1, \ldots, n$ and study the following four scenarios:

- $(\mathbf{w}, D)$ : Both players ignore the presence of an opponent; that is, the learner employs a regular classifier and the sender does not change the data generation process.

- $(\mathbf{w}, \dot{D})$ : The learner ignores the presence of an active data generator who changes the data generation process so that $D$ evolves to $\dot{D}$ by playing a Nash strategy.

- $(\dot{\mathbf{w}}, D)$ : The learner expects a rational data generator and chooses a Nash-equilibrial prediction model. However, the data generator does not change the input distribution.

- $(\dot{\mathbf{w}}, \dot{D})$ : Both players are aware of the opponent and play a Nash-equilibrial action to secure lowest costs.

We repeat this experiment 100 times for all four data sets. Table 4.2 reports on the average values of F-measure over all repetitions and both NPG instances and corresponding baselines. The numbers in boldface indicate significant differences ($\alpha = 0.05$) between the F-measure values of $g_{\mathbf{w}}$ and $g_{\dot{\mathbf{w}}}$ for fixed sample $D$ and $\dot{D}$, respectively.

**Table 4.2:** Nash equilibrium and *i.i.d.* classifier against passive and Nash-optimal data generator.

| | ESP | | | Mailinglist | | | Private | | | TREC 2007 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NLR vs. LR | | $\mathbf{w}$ | $\dot{\mathbf{w}}$ | | $\mathbf{w}$ | $\dot{\mathbf{w}}$ | | $\mathbf{w}$ | $\dot{\mathbf{w}}$ | | $\mathbf{w}$ | $\dot{\mathbf{w}}$ |
| | $D$ | **0.957** | 0.924 | $D$ | **0.987** | 0.984 | $D$ | **0.961** | 0.944 | $D$ | **0.980** | **0.979** |
| | $\dot{D}$ | 0.912 | **0.925** | $\dot{D}$ | 0.958 | **0.976** | $\dot{D}$ | 0.903 | **0.912** | $\dot{D}$ | 0.955 | **0.961** |
| | ESP | | | Mailinglist | | | Private | | | TREC 2007 | | |
| NSVM vs. SVM | | $\mathbf{w}$ | $\dot{\mathbf{w}}$ | | $\mathbf{w}$ | $\dot{\mathbf{w}}$ | | $\mathbf{w}$ | $\dot{\mathbf{w}}$ | | $\mathbf{w}$ | $\dot{\mathbf{w}}$ |
| | $D$ | **0.955** | 0.939 | $D$ | **0.987** | 0.985 | $D$ | **0.961** | 0.957 | $D$ | 0.979 | **0.981** |
| | $\dot{D}$ | 0.928 | **0.939** | $\dot{D}$ | 0.961 | **0.976** | $\dot{D}$ | 0.932 | **0.936** | $\dot{D}$ | 0.960 | **0.968** |

As expected, when the data generator does not alter the input distribution, the regularized empirical risk minimization baselines, logistic regression and the SVM, are generally best. However, the performance of those baselines drops substantially when the data generator plays the Nash-equilibrial action $\dot{D}$. The Nash-optimal prediction models are more robust against this transformation of the input distribution and significantly outperform the reference methods for all four data sets.

### 4.6.4 A Case Study on Email Spam Filtering

To study the performance of the Nash prediction models and the baselines for email spam filtering, we evaluate all methods *into the future* by processing the test set in chronological order. The test portion of each data set is split into 20 chronologically sorted disjoint subsets. We average the value of F-measure on each of those subsets over the 20 models (trained on different samples drawn from the training portion) for each method.

Figure 4.3 shows that, for all data sets, the NPG instances outperform logistic regression and the SVM that do not explicitly factor the adversary into the optimization criterion. Especially for the ESP corpus, the Nash logistic regression (NLR) and the Nash support vector machine (NSVM) are superior. On the TREC 2007 data set, the methods behave comparably with a slight advantage for the Nash support vector machine. The period over which the TREC 2007 data have been collected is very short. We believe that the training and test instances are governed by nearly identical distributions. Consequently, for this data set, the game-theoretic models do not gain a significant advantage over logistic regression and the SVM that assume *i.i.d.* samples.

Table 4.3 shows aggregated results over all four data sets. For each point in each of the diagrams of Figure 4.3, we conduct a pairwise comparison of all methods based on a paired

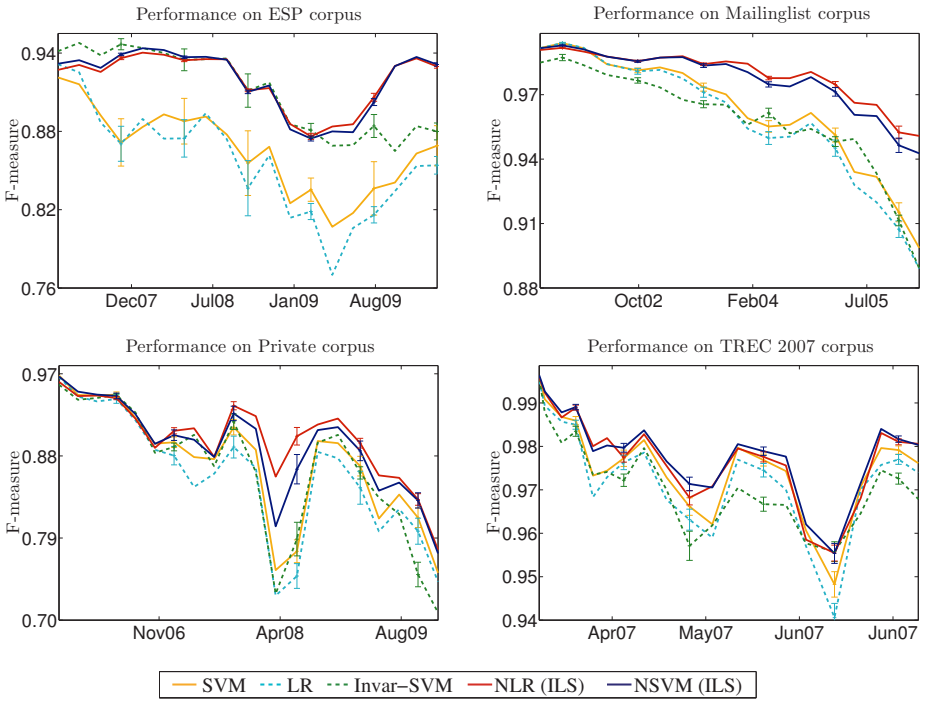


**Figure 4.3:** Value of F-measure of predictive models. Error bars indicate standard errors.

$t$-test at a confidence level of $1 - \alpha$ with $\alpha = 0.05$. When a difference is significant, we count this as a win for the method that achieves a higher value of F-measure. Each line of Table 4.3 details the wins and, set in italics, the *losses* of one method against all other methods.

**Table 4.3:** Results of paired $t$-test over all corpora: Number of trials in which each method (row) has significantly outperformed each other *method (column)* vs. number of times it *was outperformed*.

| method vs. *method* | *SVM* | *LR* | *Invar-SVM* | *NLR (ILS)* | *NSVM (ILS)* |
|---|---|---|---|---|---|
| SVM | 0:*0* | 40:*2* | 30:*20* | 8:*57* | 2:*65* |
| LR | 2:*40* | 0:*0* | 19:*29* | 5:*59* | 2:*71* |
| Invar-SVM | 20:*30* | 29:*19* | 0:*0* | 5:*57* | 3:*57* |
| NLR (ILS) | 57:*8* | 59:*5* | 57:*5* | 0:*0* | 22:*30* |
| NSVM (ILS) | 65:*2* | 71:*2* | 57:*3* | 30:*22* | 0:*0* |

The Nash logistic regression and the Nash support vector machine have more wins than they have losses against each of the other methods. The ranking continues with the Invar-SVM, the regular SVM, and logistic regression which loses more frequently than it wins against all other methods.

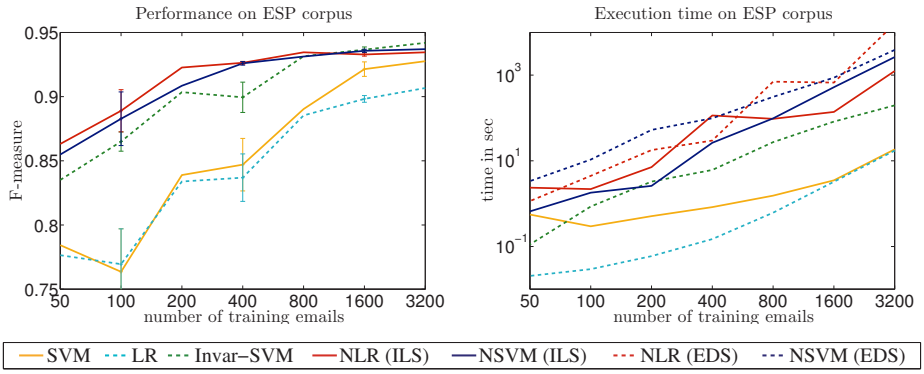


**Figure 4.4:** Predictive performance (left) and execution time (right) for varying sizes of the training data set.

### 4.6.5 Efficiency versus Effectiveness

To assess the predictive performance as well as the execution time as a function of the sample size, we train the baselines and the two NPG instances for a varying number of training examples. We report on the results for the ESP data set in Figure 4.4. The game-theoretic models significantly outperform the trivial baseline methods logistic regression and the SVM, especially for small data sets. However, this comes at the price of considerably higher computational cost. The ILS algorithm requires in general only a couple of iterations to converge; however in each iteration several optimization problems have to be solved so that the total execution time is up to a factor 150 larger than that of the corresponding *i.i.d.* baseline. In contrast to the ILS algorithm, a single iteration of the EDS algorithm does not require solving nested optimization problems. However, the execution time of the EDS algorithm is still higher as it often requires several thousand iterations to fully converge.

For larger data sets, the discrepancy in predictive performance between game-theoretic models and *i.i.d.* baseline decreases. Regarding the whether ILS or EDS is faster at solving the optimization problems that lead to the Nash equilibria our results are not conclusive. We conclude that the benefit of the NPG prediction models over the classification baseline is greatest for small to medium sample sizes.

### 4.6.6 Nash-Equilibrial Transformation

In contrast to the Invar-SVM, the Nash models allow the data generator to modify non-spam emails. However, in practice most senders of legitimate messages do not deliberately change their writing behavior in order to bypass spam filters, perhaps with the exception of senders of newsletters who must be careful not to trigger filtering mechanisms. In a final experiment, we want to study whether the Nash model reflects this aspect of reality, and how the data generator's regularizer effects this transformation.

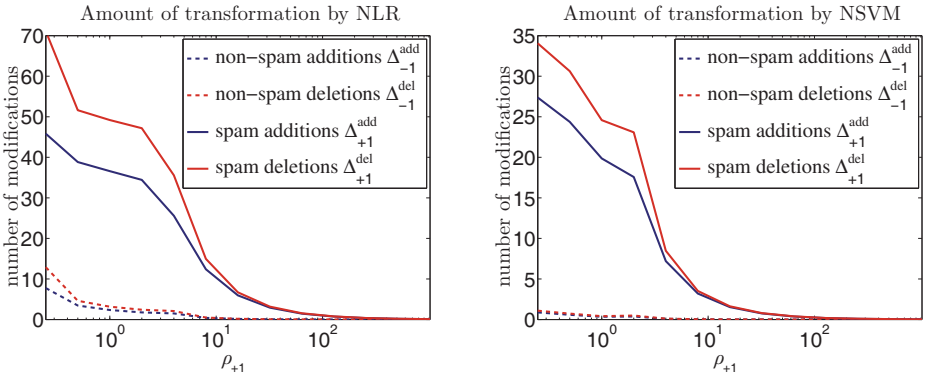


**Figure 4.5:** Average number of additions and deletions per spam/non-spam email for NLR (left) and NSVM (right) with respect to the data generator's regularization parameter $\rho_{+1}$ for fixed $\rho_{-1} = n^{-1}$.

The training portion contains again $n_{+1} = 200$ spam and $n_{-1} = 200$ non-spam instances randomly chosen from the oldest $4,000$ emails. We determine the Nash equilibrium and measure the number of additions and deletions to spam and non-spam emails in the perturbated sample $\dot{D}$,

$$\Delta^{\text{add}}_{-1} := \frac{1}{n_{-1}} \sum_{i:y_i=-1} \sum_{j=1}^{m} \max(0, \dot{\mathbf{x}}_{i,j} - \mathbf{x}_{i,j}) \qquad \Delta^{\text{add}}_{+1} := \frac{1}{n_{+1}} \sum_{i:y_i=+1} \sum_{j=1}^{m} \max(0, \dot{\mathbf{x}}_{i,j} - \mathbf{x}_{i,j})$$

$$\Delta^{\text{del}}_{-1} := \frac{1}{n_{-1}} \sum_{i:y_i=-1} \sum_{j=1}^{m} \max(0, \mathbf{x}_{i,j} - \dot{\mathbf{x}}_{i,j}) \qquad \Delta^{\text{del}}_{+1} := \frac{1}{n_{+1}} \sum_{i:y_i=+1} \sum_{j=1}^{m} \max(0, \mathbf{x}_{i,j} - \dot{\mathbf{x}}_{i,j})$$

where $\mathbf{x}_{i,j}$ indicates the presence of token $j$ in the $i$-th training email, that is, $\Delta^{\text{add}}_v$ and $\Delta^{\text{del}}_v$ denote the average number of word additions and deletions per spam and non-spam email performed by the sender.

Figure 4.5 shows the number of additions and deletions of the Nash-equilibrial transformation as a function of the adversary's regularization parameter for the ESP data set. As expected, there is no conflict of interests between the learner and the senders with respect to non-spam emails. Hence, the Nash-equilibrial transformation imposes almost no changes on any non-spam email. In addition, we observe for all data sets that even if the total amount of modifications differs, both instances NLR and NSVM behave similarly insofar as that the number of word additions and deletions continues to grow when the adversary's regularizer decreases. Table 4.4 reports on the average number of word additions and deletions for all data sets. For Invar-SVM, we set the number of possible changes to $K = 25$.

We observe that the number of additions and deletions varies between the distinct data sets even if the data generator's regularization parameter is kept fixed. The reason for this are the differing regularization parameters of the learner which are tuned by cross-validation.

**Table 4.4:** Average number of word additions and deletions per training email.

ESP

| *game* | *non-spam* | | *spam* | |
|---|---|---|---|---|
| *model* | *add* | *del* | *add* | *del* |
| Invar-SVM | 0.0 | 0.0 | 0.0 | 24.8 |
| NLR | 0.7 | 1.0 | 22.5 | 31.2 |
| NSVM | 0.4 | 0.5 | 17.9 | 23.8 |

Mailinglist

| *game* | *non-spam* | | *spam* | |
|---|---|---|---|---|
| *model* | *add* | *del* | *add* | *del* |
| Invar-SVM | 0.0 | 0.0 | 0.0 | 23.9 |
| NLR | 0.3 | 0.4 | 8.6 | 10.9 |
| NSVM | 0.3 | 0.3 | 6.9 | 8.4 |

Private

| *game* | *non-spam* | | *spam* | |
|---|---|---|---|---|
| *model* | *add* | *del* | *add* | *del* |
| Invar-SVM | 0.0 | 0.0 | 0.0 | 24.2 |
| NLR | 0.4 | 0.2 | 24.3 | 11.2 |
| NSVM | 0.1 | 0.1 | 15.6 | 7.3 |

TREC 2007

| *game* | *non-spam* | | *spam* | |
|---|---|---|---|---|
| *model* | *add* | *del* | *add* | *del* |
| Invar-SVM | 0.0 | 0.0 | 0.0 | 24.7 |
| NLR | 0.2 | 0.2 | 15.0 | 11.4 |
| NSVM | 0.2 | 0.1 | 11.1 | 8.4 |

# 5 Stackelberg Prediction Games

In this chapter we study prediction games where the learner acts *before* the data generator. We call such games *Stackelberg prediction games* (SPG). In contrast to (static) *Nash prediction games* in which players move simultaneously, an execution of the Stackelberg prediction game is carried out in two steps: At first, the learner commits to a predictive model. In our example of spam email filtering, the recipient commits to a filter, and discloses it to the sender. Only then, the data generator gets to move. In our running example, the sender creates a new sample of messages. The game may be executed repeatedly to model the course of continuous interaction between the players. However, by Assumption 3.2, the game is nevertheless an *one-shot* game where both players attempt to maximize their benefit for the current round only.

We aim at finding an optimal move of the learner in this setting. Starting with the general problem formulation in Section 3.3, we will derive the Stackelberg solution of prediction games. To this end, we consider the previously defined prediction game (*cf.* Definition 3.5) with the players' cost functions $\hat{\theta}_v(\mathbf{w}, \dot{D})$ defined in Equations 3.1 and 3.2, and assume that the learner acts before the data generator. As in the previous chapter, we define the data generator's transformation costs to be the average squared $\ell^2$-distance between $x_i$ and $\dot{x}_i$ in feature space (*cf.* Equation 4.16), and the learner's regularizer to be the squared $\ell^2$-norm of $\mathbf{w}$ (*cf.* Equation 4.17).

For these choices we derive a relaxed optimization problem to determine the Stackelberg-optimal solution in Section 5.1. We continue with a brief introduction to sequential quadratic programming to solve the derived optimization problem and show how to employ kernel functions in Sections 5.2 and 5.3, respectively. In Section 5.4 we present several instances of Stackelberg prediction games and discuss their relation to existing prediction models. We review related learning methods in Section 5.5 and perform an experimental evaluation of Stackelberg prediction games in Section 5.6.

## 5.1 Stackelberg Solution to Prediction Games

A Stackelberg game is one of the simplest dynamic games: In the first stage, the *leader*—in our case, the learner—decides on a predictive model $h_{\mathbf{w}}(x) = \operatorname{sign} g_{\mathbf{w}}(x)$ with parameter vector $\mathbf{w}$. In the second stage, the data generator, who plays the part of the *follower*, observes the leader's decision and chooses a transformation that changes the distribution of *past* instances into the distribution of *future* instances. In this scenario, the learner has to commit to a set of parameters unilaterally whereas the data generator can take the model parameters $\mathbf{w}$ into account when preparing the data transformation.

The optimality of a *Stackelberg equilibrium*, which we will now introduce, rests on the following assumption.

**Assumption 5.1.** *The following statements hold:*

1. *The learner acts before the data generator;*
2. *The learner has full knowledge about both (empirical) cost functions $\hat{\theta}_v(\mathbf{w}, \dot{D})$ defined in (3.1) and (3.2), and both action spaces $\mathcal{W}$ and $(\mathcal{X} \times \mathcal{Y})^n$;*
3. *The data generator has full access to the learner's chosen action $\mathbf{w} \in \mathcal{W}$;*
4. *Both players act rational with respect to their cost function in the sense of securing their lowest possible costs.*

To reach minimal costs given $\mathbf{w}$, the data generator has to identify a sample $\dot{D}$ that constitutes a global minimum of the cost function $\hat{\theta}_{+1}(\mathbf{w}, \dot{D})$. There may be several global minima with identical values of the cost function. In general, the data generator has to identify any element $\dot{D}$ from their *rational reaction set* (*cf.* Definition 3.3), that is, the set of optimal responses to $\mathbf{w}$,

$$A^*_{+1}(\mathbf{w}) := \left\{ \{(\dot{x}_i, y_i)\}_{i=1}^n \;\middle|\; \{\dot{x}_i\}_{i=1}^n \in \operatorname*{argmin}_{\dot{x}'_1,\ldots,\dot{x}'_n \in \mathcal{X}} \hat{\theta}_{+1}\left(\mathbf{w}, \{(\dot{x}'_i, y_i)\}_{i=1}^n\right) \right\}.$$

Identifying an element $\dot{D} \in A^*_{+1}(\mathbf{w})$ amounts to solving a regular optimization problem because $\mathbf{w}$ can be observed before $\dot{D}$ has to be chosen. A Stackelberg equilibrium is now identified by backward induction. Assuming that the data generator will decide for any $\dot{D} \in A^*_{+1}(\mathbf{w})$, the learner has to choose model parameters $\mathbf{w}^*$ that minimize the learner's cost function $\hat{\theta}_{-1}$ for any of the possible reactions $\dot{D} \in A^*_{+1}(\mathbf{w})$ that are optimal for the data generator,

$$\mathbf{w}^* \in \operatorname*{argmin}_{\mathbf{w} \in \mathcal{W}} \max_{\dot{D} \in A^*_{+1}(\mathbf{w})} \hat{\theta}_{-1}(\mathbf{w}, \dot{D}). \tag{5.1}$$

This formulation is similar to the minimax strategy under the worst-case assumption that the data generator aims at maximizing the learner's costs. The essential distinction is that the Stackelberg model restricts the data generator to choose an action from their rational reaction set rather than from their whole action space.

An action $\mathbf{w}^*$ that minimizes the learner's costs and a corresponding optimal action $\dot{D} \in A^*_{+1}(\mathbf{w})$ of the data generator are called a Stackelberg equilibrium (*cf.* Definition 3.4). Recall that a Stackelberg equilibrium is a special case of a subgame perfect equilibrium which is an extension of the Nash equilibrium for games that are played non-simultaneously.

Equation 5.1 establishes a hierarchical mathematical program—specifically, a *bilevel optimization problem*—with upper-level objective $\hat{\theta}_{-1}$ and lower-level objective $\hat{\theta}_{+1}$.

$$\min_{\mathbf{w} \in \mathcal{W}} \max_{\forall i \colon \dot{x}_i \in \mathcal{X}} \hat{\theta}_{-1}(\mathbf{w}, \{(\dot{x}_i, y_i)\}_{i=1}^n) \tag{5.2}$$

$$\text{s.t.} \qquad \{\dot{x}_i\}_{i=1}^n \in \operatorname*{argmin}_{\dot{x}'_1,\ldots,\dot{x}'_n \in \mathcal{X}} \hat{\theta}_{+1}\left(\mathbf{w}, \{(\dot{x}'_i, y_i)\}_{i=1}^n\right) \tag{5.3}$$

Bilevel programs are intrinsically hard to solve. Even the simplest instance, in which all constraints and objectives are linear, is known to be NP-hard [46]. The main difficulties arise from the constraints $\dot{x}_i' \in \mathcal{X}$ of the lower-level optimization problem which generally render constraint (5.3) of the upper-level optimization problem to be non-differentiable in $\mathbf{w}$, even if $\hat{\theta}_{+1}$ is continuously differentiable in $\mathbf{w}$ and $\dot{x}_i'$ for $i = 1, \ldots, n$.

Numerous approaches that address bilevel programs have been studied, for instance, based on gradient descent, penalty function methods, and trust-region methods (see, for instance, [21] for a detailed survey). Commonly, these methods reformulate the optimization problem into a mathematical program with equilibrium constraints. In this, the lower-level optimization problem is replaced by its Karush-Kuhn-Tucker (KKT) conditions. The resulting optimization problem with equilibrium constraints can be solved approximately by relaxing the complementary conditions [74]. However, these methods do not necessarily converge to a (local) optimum and are applicable to small-size problems only.

That is why we focus on a relaxed version of the above bilevel program: The following theorem reformulates the lower-level optimization problem into an unconstrained problem, so that constraint (5.3) becomes continuously differentiable in $\mathbf{w}$. This requires the feature space induced by mapping $\boldsymbol{\phi}$, but not necessarily the input space $\mathcal{X}$, to be unrestricted, and the data generator's loss function $\ell_{+1}(z, y)$ to be convex and continuously differentiable in $z \in \mathbb{R}$.

**Theorem 5.1.** *Let the leader's cost function $\hat{\theta}_{-1}$ and the follower's cost function $\hat{\theta}_{+1}$ be defined as in* (3.1) *and* (3.2) *with regularizers $\Omega_{-1}$ and $\Omega_{+1}$ defined as in* (4.17) *and* (4.16)*, respectively. Let feature mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^m$ be surjective, and let the data generator's loss function $\ell_{+1}(z, y)$ be convex and continuously differentiable with respect to $z \in \mathbb{R}$ for any fixed $y \in \mathcal{Y}$. Further, let weight vector $\mathbf{w}^* \in \mathbb{R}^m$ and factors $\tau_1^*, \ldots, \tau_n^* \in \mathbb{R}$ be a solution of the optimization problem*

$$\begin{aligned} \min_{\mathbf{w}, \forall i \colon \tau_i} \quad & \sum_{i=1}^{n} c_{-1,i} \ell_{-1}\left(g_{\mathbf{w}}(x_i) + \tau_i \|\mathbf{w}\|^2, y_i\right) + \frac{\rho_{-1}}{2}\|\mathbf{w}\|_2^2 \\ \text{s.t.} \quad & 0 = \tau_i + \frac{n}{\rho_{+1}} c_{+1,i} \ell_{+1}'\left(g_{\mathbf{w}}(x_i) + \tau_i \|\mathbf{w}\|_2^2, y_i\right) \quad \forall i = 1, \ldots, n. \end{aligned} \tag{5.4}$$

*Then, the Stackelberg prediction game in Equation 5.2 attains an equilibrium at $(\mathbf{w}^*, \dot{D}^*)$ with $\dot{D}^* = \{(\dot{x}_i^*, y_i)\}_{i=1}^n$ and $\dot{x}_i^* \in \{\dot{x} \in \mathcal{X} \mid \boldsymbol{\phi}(\dot{x}) = \boldsymbol{\phi}(x_i) + \tau_i^* \mathbf{w}^*\}$.*

*Proof.* Constraint 5.3 states that $\{\dot{x}_i^*\}_{i=1}^n$ is a solution of the restricted optimization problem

$$\min_{\forall i \colon \dot{x}_i \in \mathcal{X}} \sum_{i=1}^{n} c_{+1,i} \ell_{+1}(\mathbf{w}^\mathsf{T} \boldsymbol{\phi}(\dot{x}_i), y_i) + \frac{\rho_{+1}}{n} \sum_{i=1}^{n} \frac{1}{2} \|\boldsymbol{\phi}(\dot{x}_i) - \boldsymbol{\phi}(x_i)\|_2^2.$$

As the objective as well as the constraints are entirely defined in terms of $\dot{\mathbf{x}}_i^* = \boldsymbol{\phi}(\dot{x}_i^*)$, this condition is equivalent to enforcing $\{\mathbf{x}_i^*\}_{i=1}^n$ to be a solution of the unrestricted optimization

problem

$$\min_{\forall i\,:\,\dot{\mathbf{x}}_i \in \mathbb{R}^m} \sum_{i=1}^{n} c_{+1,i}\ell_{+1}(\mathbf{w}^\mathsf{T}\dot{\mathbf{x}}_i, y_i) + \frac{\rho_{+1}}{n}\sum_{i=1}^{n}\frac{1}{2}\|\dot{\mathbf{x}}_i - \boldsymbol{\phi}(x_i)\|_2^2. \tag{5.5}$$

This solution is uniquely defined for any fixed $\mathbf{w}$ as loss function $\ell_{+1}(z, y)$ is required to be convex in $z$, and consequently in $\dot{\mathbf{x}}_i$, and the term $\|\dot{\mathbf{x}}_i - \boldsymbol{\phi}(x_i)\|_2^2$ is quadratic in $\dot{\mathbf{x}}_i$ and therefore strictly convex for any fixed $\boldsymbol{\phi}(x_i)$. Given $\mathbf{w} \in \mathcal{W}$ and minimizer $\dot{\mathbf{x}}_i^* \in \mathbb{R}^m$, the set $\dot{\mathcal{X}}_{\mathbf{w}}^i := \{\dot{x}^* \in \mathcal{X} \mid \boldsymbol{\phi}(\dot{x}^*) = \dot{\mathbf{x}}_i^*\}$ contains all instances $\dot{x}^*$ which correspond to the optimally transformed instance in feature space $\dot{\mathbf{x}}_i^*$. Since $\boldsymbol{\phi}$ is surjective, $\dot{\mathcal{X}}_{\mathbf{w}}^i$ is guaranteed to be non-empty, and consequently, for any solution $\{\dot{\mathbf{x}}_i^*\}_{i=1}^n$, there exist at least one corresponding set of instances $\{\dot{x}_i^*\}_{i=1}^n$. As $\boldsymbol{\phi}$ is not required to be a bijective mapping, there may exist multiple instances $\dot{x} \in \dot{\mathcal{X}}_{\mathbf{w}}^i$ which are optimal in the sense of minimizing the data generator's loss. However, since all of these instances share the same feature representation $\dot{\mathbf{x}}_i^*$, the inner maximization of the upper-level optimization problem in (5.2) vanishes,

$$\min_{\mathbf{w}\in\mathcal{W}} \max_{\forall i\,:\,\dot{x}_i \in \dot{\mathcal{X}}_{\mathbf{w}}^i} \hat{\theta}_{-1}\left(\mathbf{w}, \{(\dot{x}_i, y_i)\}_{i=1}^n\right) = \min_{\mathbf{w}\in\mathcal{W}} \sum_{i=1}^{n} c_{-1,i}\ell_{-1}\left(\mathbf{w}^\mathsf{T}\dot{\mathbf{x}}_i^*, y_i\right) + \frac{\rho_{-1}}{2}\|\mathbf{w}\|_2^2, \tag{5.6}$$

where $\{\mathbf{x}_i^*\}_{i=1}^n$ is the solution of the optimization problem in (5.5). Since 5.5 is convex, this constraint can be replaced by its complementary conditions which are given by $\nabla_{\dot{\mathbf{x}}_i}\hat{\theta}_{+1}(\mathbf{w}, \dot{D}) = \mathbf{0}$ for $i = 1, \ldots, n$, where

$$\nabla_{\dot{\mathbf{x}}_i}\hat{\theta}_{+1}(\mathbf{w}, \dot{D}) = c_{+1,i}\ell'_{+1}(\mathbf{w}^\mathsf{T}\dot{\mathbf{x}}_i^*, y_i)\mathbf{w} + \frac{\rho_{+1}}{n}(\dot{\mathbf{x}}_i - \boldsymbol{\phi}(x_i)).$$

The mapped instance $\dot{\mathbf{x}}_i^*$, that satisfies the $i$-th complementary condition, is given by

$$\dot{\mathbf{x}}_i^* = \boldsymbol{\phi}(x_i) + \tau_i\mathbf{w} \tag{5.7}$$

with

$$\begin{aligned}
\tau_i &= -\frac{n}{\rho_{+1}}c_{+1,i}\ell'_{+1}\left(\mathbf{w}^\mathsf{T}\dot{\mathbf{x}}_i^*, y_i\right) \\
&= -\frac{n}{\rho_{+1}}c_{+1,i}\ell'_{+1}\left(\mathbf{w}^\mathsf{T}\boldsymbol{\phi}(x_i) + \tau_i\mathbf{w}^\mathsf{T}\mathbf{w}, y_i\right) \\
&= -\frac{n}{\rho_{+1}}c_{+1,i}\ell'_{+1}\left(g_{\mathbf{w}}(x_i) + \tau_i\|\mathbf{w}\|_2^2, y_i\right).
\end{aligned} \tag{5.8}$$

When replacing $\dot{\mathbf{x}}_i^*$ by (5.7) in the upper-level optimization problem in (5.6) and enforcing Equation 5.8, the optimization problem in (5.4) follows. Hence, a solution $\mathbf{w}^*$ of (5.4) with corresponding $\tau_1^*, \ldots, \tau_n^*$ is also a solution of (5.2) with $\dot{x}_i^* \in \dot{\mathcal{X}}_{\mathbf{w}^*}^i := \{\dot{x} \in \mathcal{X} \,|\, \boldsymbol{\phi}(\dot{x}) = \boldsymbol{\phi}(x_i) + \tau_i^*\mathbf{w}^*\}$. □

Theorem 5.1 translates the bilevel problem of finding a Stackelberg equilibrium into a compact optimization problem which can be solved, for instance, by applying interior point methods such as Ipopt [75] or by sequential quadratic programming (SQP) methods which are introduced in the following section.

## 5.2 An SQP Method for Stackelberg Prediction Games

The objective as well as the constraints of the optimization problem in Theorem 5.1 are generally not jointly convex in $\mathbf{w}$ and $\boldsymbol{\tau} := [\tau_1, \ldots, \tau_n]^\mathsf{T}$. However, under the assumptions of the following proposition, a locally optimal solution can still be found efficiently by standard SQP methods.

**Proposition 5.2.** *Let loss function $\ell_{-1}(z, y)$ be twice-continuously differentiable and loss function $\ell_{+1}(z, y)$ be convex and thrice-continuously differentiable with respect to $z \in \mathbb{R}$ for any fixed $y \in \mathcal{Y}$. Then, a point satisfying the KKT conditions of the optimization problem in Equation* (5.4) *can be obtained by* sequential quadratic programming *(SQP) methods.*

The objective as well as the constraints in (5.4) are twice-continuously differentiable with respect to $\mathbf{w}$ and $\tau_i$ for $i = 1, \ldots, n$. Hence, the corresponding complementary conditions are continuously differentiable which is a sufficient condition to apply sequential quadratic programming methods. This proves the proposition.

We shall briefly review sequential quadratic programming methods to solve Stackelberg prediction games. The basic idea of SQP methods is to search for a point satisfying the KKT conditions of the original *non-linear program* (NLP) by applying Newton's method. Following this approach, Equation (5.9) states the Lagrangian of the NLP of Equation 5.4,

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\tau}, \boldsymbol{\alpha}) := q(\mathbf{w}, \boldsymbol{\tau}) - \sum_{i=1}^{n} \alpha_i u_i(\mathbf{w}, \boldsymbol{\tau}), \tag{5.9}$$

where $\boldsymbol{\alpha} := [\alpha_1, \ldots, \alpha_n]^\mathsf{T}$ and

$$\begin{aligned} q(\mathbf{w}, \boldsymbol{\tau}) &:= \sum_{i=1}^{n} c_{-1,i} \ell_{-1}\left(g_\mathbf{w}(x_i) + \tau_i \|\mathbf{w}\|_2^2, y_i\right) + \frac{\rho_{-1}}{2}\|\mathbf{w}\|_2^2, \\ u_i(\mathbf{w}, \boldsymbol{\tau}) &:= \tau_i + \frac{n}{\rho_{+1}} c_{+1,i} \ell'_{+1}\left(g_\mathbf{w}(x_i) + \tau_i \|\mathbf{w}\|_2^2, y_i\right). \end{aligned}$$

The corresponding KKT conditions are given by

$$\begin{aligned} \nabla \mathcal{L}(\mathbf{w}, \boldsymbol{\tau}, \boldsymbol{\alpha}) &= \begin{bmatrix} \nabla_\mathbf{w} \mathcal{L}(\mathbf{w}, \boldsymbol{\tau}, \boldsymbol{\alpha}) \\ \nabla_{\boldsymbol{\tau}} \mathcal{L}(\mathbf{w}, \boldsymbol{\tau}, \boldsymbol{\alpha}) \\ \nabla_{\boldsymbol{\alpha}} \mathcal{L}(\mathbf{w}, \boldsymbol{\tau}, \boldsymbol{\alpha}) \end{bmatrix} \\ &= \begin{bmatrix} \nabla_\mathbf{w} q(\mathbf{w}, \boldsymbol{\tau}) - \operatorname{diag}(\alpha_1, \ldots, \alpha_n)\, \mathbf{J}_\mathbf{w}(\mathbf{w}, \boldsymbol{\tau})^\mathsf{T} \\ \nabla_{\boldsymbol{\tau}} q(\mathbf{w}, \boldsymbol{\tau}) - \operatorname{diag}(\alpha_1, \ldots, \alpha_n)\, \mathbf{J}_{\boldsymbol{\tau}}(\mathbf{w}, \boldsymbol{\tau})^\mathsf{T} \\ -u(\mathbf{w}, \boldsymbol{\tau}) \end{bmatrix} \\ &= \mathbf{0} \end{aligned}$$

where

$$\begin{aligned} \mathbf{J}_\mathbf{w}(\mathbf{w}, \boldsymbol{\tau}) &= [\nabla_\mathbf{w} u_1(\mathbf{w}, \boldsymbol{\tau}), \ldots, \nabla_\mathbf{w} u_n(\mathbf{w}, \boldsymbol{\tau})]^\mathsf{T} \\ \mathbf{J}_{\boldsymbol{\tau}}(\mathbf{w}, \boldsymbol{\tau}) &= [\nabla_{\boldsymbol{\tau}} u_1(\mathbf{w}, \boldsymbol{\tau}), \ldots, \nabla_{\boldsymbol{\tau}} u_n(\mathbf{w}, \boldsymbol{\tau})]^\mathsf{T} \end{aligned}$$

denote the partial Jacobians of the vector-valued function $u(\mathbf{w}, \boldsymbol{\tau}) := [u_1(\mathbf{w}, \boldsymbol{\tau}), \ldots, u_n(\mathbf{w}, \boldsymbol{\tau})]^\mathsf{T}$ with respect to $\mathbf{w}$ and $\boldsymbol{\tau}$, respectively. In order to find a solution $(\mathbf{w}^*, \boldsymbol{\tau}^*)$ of the NLP with optimal multipliers $\alpha_i^* \in \mathbb{R}$, $i = 1, ..., n$, Newton's method is applied to the non-linear system of equations $\nabla \mathcal{L}(\mathbf{w}, \boldsymbol{\tau}, \boldsymbol{\alpha}) = \mathbf{0}$. Given an initial estimate $(\mathbf{w}^{(0)}, \boldsymbol{\tau}^{(0)}, \boldsymbol{\alpha}^{(0)})$ of the solution, a sequence $\{(\mathbf{w}^{(k)}, \boldsymbol{\tau}^{(k)}, \boldsymbol{\alpha}^{(k)})\}$ is generated by

$$(\mathbf{w}^{(k+1)}, \boldsymbol{\tau}^{(k+1)}, \boldsymbol{\alpha}^{(k+1)}) := (\mathbf{w}^{(k)}, \boldsymbol{\tau}^{(k)}, \boldsymbol{\alpha}^{(k)}) + (\mathbf{d}_\mathbf{w}^{(k)}, \mathbf{d}_{\boldsymbol{\tau}}^{(k)}, \mathbf{d}_{\boldsymbol{\alpha}}^{(k)}), \tag{5.10}$$

which is known to converge to $(\mathbf{w}^*, \boldsymbol{\tau}^*, \boldsymbol{\alpha}^*)$ under some mild conditions (see, for instance, [4]). Thereby, the descent direction $(\mathbf{d}_\mathbf{w}^{(k)}, \mathbf{d}_{\boldsymbol{\tau}}^{(k)}, \mathbf{d}_{\boldsymbol{\alpha}}^{(k)})$ in iteration $k$ is obtained as solution of the linearization of the KKT conditions, that is,

$$\nabla^2 \mathcal{L}(\mathbf{w}^{(k)}, \boldsymbol{\tau}^{(k)}, \boldsymbol{\alpha}^{(k)}) \begin{bmatrix} \mathbf{d}_\mathbf{w}^{(k)} \\ \mathbf{d}_{\boldsymbol{\tau}}^{(k)} \\ \mathbf{d}_{\boldsymbol{\alpha}}^{(k)} \end{bmatrix} = -\nabla \mathcal{L}(\mathbf{w}^{(k)}, \boldsymbol{\tau}^{(k)}, \boldsymbol{\alpha}^{(k)}), \tag{5.11}$$

where $\nabla^2 \mathcal{L}(\mathbf{w}^{(k)}, \boldsymbol{\tau}^{(k)}, \boldsymbol{\alpha}^{(k)})$ denotes the Hessian of the Lagrangian. It can be shown, that solving the linear system of equations in (5.11) is equivalent to computing the solution of a quadratic program in each iteration for which reason these methods are called sequential quadratic programming methods. There exist a large variety of literature on NLP solvers. We refer the interested reader, for instance, to [4].

## 5.3 Applying Kernels

Theorem 5.1 states that a Stackelberg equilibrium with parameter vector $\mathbf{w} \in \mathcal{W}$ can be obtained by solving the optimization problem in (5.4), which requires an explicit feature representation $\boldsymbol{\phi}(x_i)$ of the training instances. However, in some applications, such a feature mapping is unwieldy or even not existing. Instead, one is often equipped with a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which measures the similarity between two instances (see the discussion on kernel functions in Section 2.6). Generally, kernel function $k$ is a positive semi-definite mapping, that is, a function that can be stated as an inner product with $k(x, x') := \boldsymbol{\phi}(x)^\mathsf{T} \boldsymbol{\phi}(x')$. Making use of the representer theorem [63], we can now express weight vector $\mathbf{w}$ as a linear combination of the mapped training instances, that is,

$$\mathbf{w} = \sum_{i=1}^{n} \omega_i \boldsymbol{\phi}(x_i) \tag{5.12}$$

where feature mapping $\boldsymbol{\phi}$ is implicitly defined by kernel $k$. When substituting $\mathbf{w}$ in (5.4) by (5.12), the squared $l^2$-norm of $\mathbf{w}$ and function $g_\mathbf{w}$ can be completely expressed in terms of the kernel,

$$\|\mathbf{w}\|_2^2 = \sum_{j,k=1}^{n} \omega_j \omega_k k(x_j, x_k) \qquad \text{and} \qquad g_\mathbf{w}(x_i) = \sum_{j=1}^{n} \omega_j k(x_i, x_j).$$

Hence, the optimization problem in (5.4) can be reformulated into an optimization problem over $\boldsymbol{\tau} := [\tau_1, \ldots, \tau_n]^\mathsf{T}$ and the dual weights $\boldsymbol{\omega} := [\omega_1, \ldots, \omega_n]^\mathsf{T}$ without the need of an explicit feature mapping $\boldsymbol{\phi}$. However, inferring an optimal transformed sample $\dot{D}^*$ still requires the knowledge of an explicit mapping $\boldsymbol{\phi}$ and its inverse $\boldsymbol{\phi}^{-1}$. Of course, this is not a restriction, as we are interested in the predictive model $g_\mathbf{w}$ rather than the transformed sample $\dot{D}^*$.

Note, that because of computational issues, it may be advisable to firstly construct an explicit feature mapping from the kernel matrix, and then to train the Stackelberg model in the primal. To this end, we may use the kernel PCA mapping (*cf.* Equation 2.30) as introduced in Section 2.6. Within our experiments on Stackelberg prediction games in Chapter 5.6 we use the linear kernel and study all three variants: Computing the model in input space, computing the kernelized version, and computing the kernel PCA map-induced variant. Even though all variants yield the same solution, using an explicit mapping is generally fastest for reasonable sample sizes.

## 5.4 Instances of the Stackelberg Prediction Game

By the choice of $\ell_v$, distinct instances of the Stackelberg prediction game (SPG) can be identified which, to some extent, generalize existing prediction models such as the *SVM for invariances* [71] and the *SVM with uneven margins* [50].

### 5.4.1 Worst-Case Loss

The *SPG with worst-case loss* is an instance of the Stackelberg prediction game that is characterized by an antagonicity of the empirical costs of learner and data generator, that is, the data generator employs the loss function

$$\ell^{\text{wc}}_{+1}(z, y) := -\ell_{-1}(z, y)$$

and cost factors $c_{+1,i} := c_{-1,i}$. This instance models a worst-case setting, and for the special case of $\rho_{-1} = \rho_{+1} = 0$, establishes a zero-sum game.

Loss functions $\ell^{\text{wc}}_{+1}$ and $\ell_{-1}$ cannot both be convex at the same time, and so the requirements of either Theorem 5.1 or Proposition 5.2 are violated. As we cannot apply Theorem 5.1, we consider the original optimization problem (Equations 5.2-5.3). We substitute $\ell^{\text{wc}}_{+1}$ and $c_{+1,i}$ in the objective (Equation 3.2) of the lower-level optimization problem which gives

$$\min_{\forall i \,:\, \dot{x}_i \in \mathcal{X}} \sum_{i=1}^{n} c_{+1,i} \ell^{\text{wc}}_{+1}\left(g_\mathbf{w}(\dot{x}_i), y_i\right) + \rho_{+1} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \|\boldsymbol{\phi}(\dot{x}_i) - \boldsymbol{\phi}(x_i)\|_2^2.$$

This optimization problem decouples into $n$ maximization problems for $i = 1, \ldots, n$,

$$\max_{\dot{x}_i \in \mathcal{X}} c_{-1,i} \ell_{-1}(g_\mathbf{w}(\dot{x}_i), y_i) - \rho_{+1} \frac{1}{2n} \|\boldsymbol{\phi}(\dot{x}_i) - \boldsymbol{\phi}(x_i)\|_2^2 \tag{5.13}$$

or equivalently,

$$\max_{\dot{x}_i \in \mathcal{X}'_i} c_{-1,i}\ell_{-1}(g_{\mathbf{w}}(\dot{x}_i), y_i) \tag{5.14}$$

where $\mathcal{X}'_i := \{\dot{x} \in \mathcal{X} \,|\, \rho'_{+1} \leq \frac{1}{2n}\|\boldsymbol{\phi}(\dot{x}) - \boldsymbol{\phi}(x_i)\|_2^2\}$ are feasible sets of transformed instances for some $\rho'_{+1}$. The difference between both formulations is that in (5.14), regularization parameter $\rho'_{+1}$ explicitly restricts the amount of transformation of each instance $x_i$. However, from the Lagrangian of (5.14) we observe that for every finite $\rho_{+1} > 0$ there exist $\rho'_{+1} > 0$ so that a maximizer of (5.14) is also a maximizer of (5.13) and vice versa.

As now the inner maximization of the upper-level optimization problem in (5.2) can be stated in terms of the solution of the lower-level optimization problem, $c_{-1,i}\ell_{-1}(g_{\mathbf{w}}(\dot{x}_i^*), y_i)$, the entire bilevel optimization problem reduces to the following constrained minimization problem.

$$\min_{\mathbf{w}, \forall i\,:\,\xi_i} \quad \sum_{i=1}^{n} \xi_i + \rho_{-1}\frac{1}{2}\|\mathbf{w}\|_2^2 \tag{5.15}$$

$$\text{s.t.} \quad \xi_i \geq \max_{\dot{x}_i \in \mathcal{X}'_i} \; c_{-1,i}\ell_{-1}(g_{\mathbf{w}}(\dot{x}_i), y_i) \quad i = 1, \ldots, n \tag{5.16}$$

If the lower-level maximization problem (5.16) has a unique solution for any fixed $\mathbf{w} \in \mathcal{W}$, then the above optimization problem can be solved by gradient descent where in each iteration the maximization problem in (5.16) has to be solved for the current iterate $\mathbf{w}^{(k)}$. Note, that at the optimum the constrains in (5.16) will become equations and when inserting those in (5.15), the above problem can be stated as a minimax problem.

In case the learner choses the hinge loss (*cf.* Definition 2.2), the SPG with worst-case loss reduces to a variant of the *SVM for invariances* [71]. Here, the feasible transformed instances of $\boldsymbol{\phi}(x_i)$ are assumed to lie within a hypersphere of radius $\sqrt{2n\rho'_{+1}}$ centered at $\boldsymbol{\phi}(x_i)$, that is, $\mathcal{X}'_i := \{\dot{x} \in \mathcal{X} \,|\, \rho'_{+1} \leq \frac{1}{2n}\|\boldsymbol{\phi}(\dot{x}) - \boldsymbol{\phi}(x_i)\|_2^2\}$, where the corresponding *costs under the worst-case transformation* are proportional to the shift of the instances, $\sum_{i=1}^{n} \|\boldsymbol{\phi}(\dot{x}) - \boldsymbol{\phi}(x_i)\|_2^2$.

### 5.4.2 Linear Loss

A second instance of the Stackelberg prediction game is the *SPG with linear loss* where the data generator employs a linear loss function,

$$\ell_{+1}^{\text{lin}}(z, y) := z,$$

which penalizes high decision values $z$ independently of the target attribute. This choice is appropriate, for instance, in email spam filtering where the data generator is purely interested in the delivery of an email $x$, which becomes unlikely for large values of $z$, independently of the corresponding true label $y$.

For the linear loss, that is continuously differentiable and convex, the constraints in (5.4) reduce to

$$\tau_i = -\frac{n}{\rho_{+1}} c_{+1,i} \tag{5.17}$$

for $i = 1, \ldots, n$.

When choosing the hinge loss (*cf.* Definition 2.2) for the learner and replacing $\tau_i$ in (5.4) by (5.17) we arrive at the following minimization problem.

$$\begin{aligned} \min_{\mathbf{w}, \forall i:\, \xi_i} \quad & \sum_{i=1}^{n} c_{-1,i} \xi_i + \rho_{-1} \frac{1}{2} \|\mathbf{w}\|_2^2 \\ \text{s.t.} \quad & \xi_i \geq 0,\ \xi_i \geq 1 - y_i \left( \mathbf{w}^\mathsf{T} \boldsymbol{\phi}(x_i) - \frac{n}{\rho_{+1}} c_{+1,i} \|\mathbf{w}\|_2^2 \right) \quad i = 1, \ldots, n \end{aligned}$$

The latter constraints can be reformulated to

$$y_i \mathbf{w}^\mathsf{T} \boldsymbol{\phi}(x_i) \geq 1 + y_i \kappa_i - \xi_i$$

which amounts to the constraints of the *SVM with uneven margins* [50]. The only syntactic distinction is that $\kappa_i := \frac{n}{\rho_{+1}} c_{+1,i} \|\mathbf{w}\|_2^2$ is implicitly defined by $\rho_{+1}$ and $c_{+1,i}$. However, for each choice of $\kappa_i > 0$ in the SVM with uneven margins, there exist appropriate parameters $\rho_{+1}$ and $c_{+1,i}$ of an equivalent SPG with linear loss and vice versa.

Consider the special case of equal factors $c_{+1,i} = c_{+1,j}$, and consequently $\kappa = \kappa_i = \kappa_j$, for all $i, j = 1, \ldots, n$. Then the margin of negative instances becomes $1 - \kappa$ whereas the margin of positive instances is $1 + \kappa$. In our example of spam filtering, this goes with the intuition that the margin of spam instances that vary greatly has to be larger than the margin of non-spam instances that remain almost unmodified. This effect is stronger when the data generator's regularization parameter $\rho_{+1}$ is small. By contrast, if $\rho_{+1}$ goes to infinity, and consequently $\kappa$ attains zero, then the SPG with linear loss reduces to the regular support vector machine.

### 5.4.3 Logistic Loss

Finally, this section introduces the *SPG with logistic loss*. This instantiation meets the preconditions of Theorem 5.1 and Proposition 5.2, and the resulting optimization criterion can be solved with standard tools. The learner may use any loss function that is convex and twice-continuously differentiable (*e.g.,* the trigonometric loss, the exponential loss, or the logistic loss, see Definitions 2.4-2.6) while the data generator uses as for the Nash logistic regression (*cf.* Section4.4.1), the class-independent logistic loss

$$\ell^{\mathrm{l}}_{+1}(z, y) := \log\left(1 + \exp(z)\right).$$

The rational behind this loss is the same as for the linear loss: Mapping $\ell^{\mathrm{l}}_{+1}(z, y)$ penalizes large decision values $z$ whereas it approaches zero for small values of $z$.

For this choice, the constraints in (5.4) resolve to $u_i(\mathbf{w}, \tau_i) = 0$ for $i = 1, \ldots, n$ with

$$u_i(\mathbf{w}, \tau_i) := \tau_i \left(1 + \exp(-g_{\mathbf{w}}(x_i) - \tau_i \|\mathbf{w}\|_2^2)\right) + \frac{n}{\rho_{+1}} c_{+1,i}.$$

Functions $u_i(\mathbf{w}, \tau_i)$ are not jointly convex in $\mathbf{w}$ and $\tau_i$. However, as they are smooth (*i.e.,* infinitely differentiable) in all arguments, their roots can be obtained efficiently and, consequently, a locally optimal solution of the resulting optimization problem

$$\begin{aligned} \min_{\mathbf{w}, \forall i:\, \tau_i} \quad & \sum_{i=1}^{n} c_{-1,i} \ell_{-1}\left(g_{\mathbf{w}}(x_i) + \tau_i \|\mathbf{w}\|_2^2, y_i\right) + \frac{\rho_{-1}}{2} \|\mathbf{w}\|_2^2 \\ \text{s.t.} \quad & 0 = u_i(\mathbf{w}, \tau_i) \quad i = 1, \ldots, n \end{aligned}$$

can be identified (see Section 5.2).

## 5.5 Related Work

Dynamic games, especially Stackelberg games, are commonly used to model economic applications (for a survey see, for instance, [54]). In the context of machine learning, the concept of Stackelberg games has rarely been used. For instance, some authors [47, 48, 52] study the case in which the data generator acts as leader and the learner as follower. This reflects a setting in which the adversary discloses how the future distribution will differ from the current distribution before the learner has to commit to a model. Such a model addresses the problem of finding an optimal data transformation from perspective of a data generator who has *no* access to the learner's chosen prediction model. However, this model contradicts the intuition of an adversarial model-building problem: If the learner is assumed to react on the data generator, the data generator's action is fixed at this time so that the learner can consider this action, *i.e.,* the (unlabeled) test data, when building the predictive model. This problem is better modeled as a distribution shift problem which we address in Chapter 6. In case the learner does not react on the data generator and under the made assumption that the data generator has no access to the predictive model, the setting amounts to a game of simultaneously acting players which we study in Chapter 4.

Parameswaran et al. [56] study a setting where an IP blacklist provider aims at finding an optimal blocking policy which accounts for a spam sender who controls the rate of outgoing spam messages. As in the Stackelberg prediction game, the adversary is assumed to react on the provider's decision by changing their sending behavior with respect to the amount of sent spam emails per time frame. Similar to our approach, the adversary's optimal response is expressed in terms of optimality conditions which are considered in the learner's optimization problem.

A special case of a Stackelberg competition is the case where the data generator desires to impose the highest possible costs on the learner, *i.e.,* the adversarial learning problem is modeled under the worst-case assumption (*cf.* Section 5.4.1). In this setting, the Stackelberg prediction game can be expressed as a minimax problem. El Ghaoui et al. [35]

derive a minimax model for input data that are known to lie within some hyper-rectangles around the training instances. Their solution minimizes the worst-case loss over all possible choices of the data in these intervals. Similarly, worst-case solutions to classification games in which the data generator deletes input features [25, 26, 36, 37] or performs an arbitrary feature transformation [71] have been studied. As discussed in Section 5.4.1, the latter approach equals the Stackelberg prediction game, if the learner chooses the hinge loss and data generator chooses the worst-case loss, that is, the negative hinge loss.

## 5.6 Empirical Evaluation

In this section we empirically evaluate the presented instances of Stackelberg prediction games: *SPG with worst-case loss* ($\text{SPG}^{\text{wc}}$, *cf.* Section 5.4.1), *SPG with linear loss* ($\text{SPG}^{\text{lin}}$, *cf.* Section 5.4.2), and *SPG with logistic loss* ($\text{SPG}^{\text{log}}$, *cf.* Section 5.4.3). We compare the new models with the existing baseline methods *logistic regression* (LR), the *support vector machine* (SVM), and the worst-case solution *SVM for invariances* with feature removal (Invar-SVM, *cf.* [71]).

For all Stackelberg instances we choose the logistic loss function (*cf.* Definition 2.6) for the learner which is convex and smooth, and consequently satisfies Proposition 5.2. We set again $c_{v,i} := \frac{1}{n}$ for all $v \in V = \{-1, +1\}$, $i = 1, \ldots, n$, and use the same data sets and feature representation as in the previous chapter (*cf.* Section 4.6).

Our evaluation protocol is as follows. We use the 4,000 oldest emails as training portion and set the remaining emails aside as test instances. We use the F-measure as evaluation measure and train all methods 20 times on a stratified subset of 200 spam and 200 non-spam messages sampled from the training portion. In order to tune the regularization parameters $\rho_{-1}$ and $\rho_{+1}$, we perform a 5-fold cross validation on the training sample within each repetition of an experiment and for each method separately.

### 5.6.1 A Case Study on Email Spam Filtering

In the first experiment, we consider the same experiment as in Section 4.6.4, that is, we evaluate all methods *into the future* by processing the test set in chronological order. Each test sample is split into 20 disjoint subsets. We again average the F-measure on each of those subsets and for each method over all resulting models of the 20 iterations, and perform a statistical test of significance.

Figure 5.1 shows that, for all data sets, the Stackelberg prediction games with linear loss and with logistic loss outperform the regular SVM and logistic regression, that do not explicitly factor the adversarial data generator into the optimization criterion. On the ESP corpus, the SPG with linear loss is slightly better than the SPG with logistic loss whereas for the Mailinglist corpus the SPG with logistic loss outperforms the SPG with linear loss. On the TREC 2007 data set, most of the methods behave comparably with a slight advantage for the SPG instances with logistic loss and worst-case loss. The period over which the TREC 2007 data have been collected is very short. Therefore we believe that the training

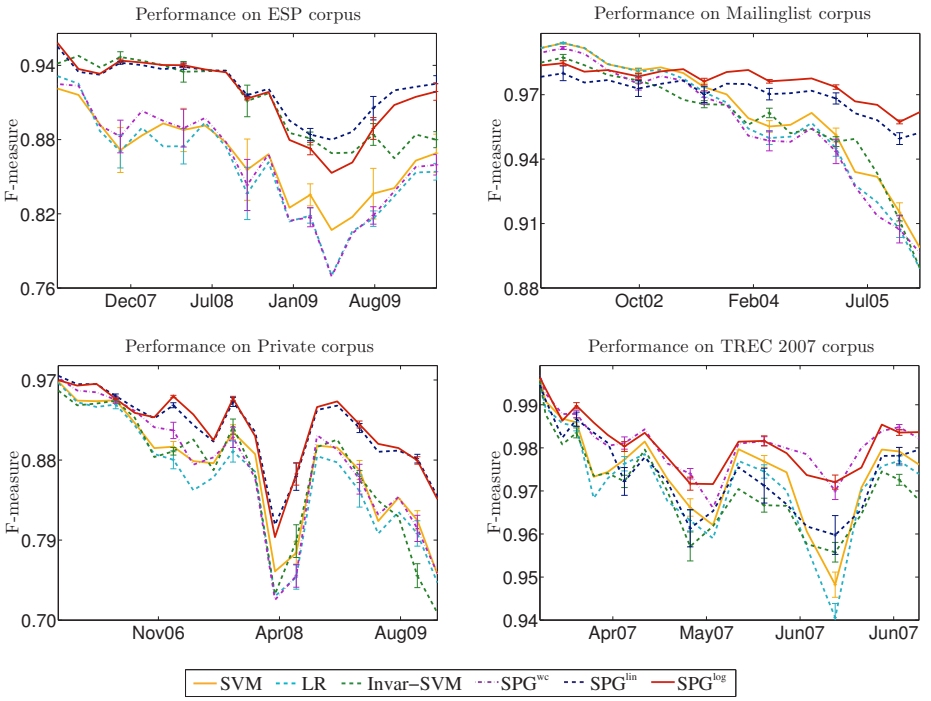


**Figure 5.1:** F-measure of predictive models. Error bars indicate standard errors.

and test instances are governed by nearly identical distributions. Consequently the game-theoretic models do only gain a slight advantage over logistic regression that assumes *i.i.d.* samples.

Table 5.1 shows aggregated results over all four data sets. For each point in each of the diagrams of Figure 5.1, we conduct a pairwise comparison of all methods based on a paired $t$-test at a confidence level of $1 - \alpha$ with $\alpha = 0.05$. When a difference is significant, we count this as a win for the method that achieves a higher F-measure. Each line of Table 5.1 details the wins and, set in italics, the *losses* of one method against all other methods.

**Table 5.1:** Results of paired $t$-test over all corpora: Number of trials in which each method (row) has significantly outperformed each other *method (column)* vs. number of times it *was outperformed*.

| method vs. *method* | *SVM* | *LR* | *Invar-SVM* | *SPG$^{wc}$* | *SPG$^{lin}$* | *SPG$^{log}$* |
|---|---|---|---|---|---|---|
| SVM | 0:*0* | 40:*2* | 30:*20* | 11:*26* | 9:*51* | 5:*63* |
| LR | 2:*40* | 0:*0* | 19:*29* | 6:*36* | 7:*54* | 5:*68* |
| Invar-SVM | 20:*30* | 29:*19* | 0:*0* | 24:*36* | 5:*40* | 1:*56* |
| SPG$^{wc}$ | 26:*11* | 36:*6* | 36:*24* | 0:*0* | 17:*46* | 9:*48* |
| SPG$^{lin}$ | 51:*9* | 54:*7* | 40:*5* | 46:*17* | 0:*0* | 10:*23* |
| SPG$^{log}$ | 63:*5* | 68:*5* | 56:*1* | 48:*9* | 23:*10* | 0:*0* |

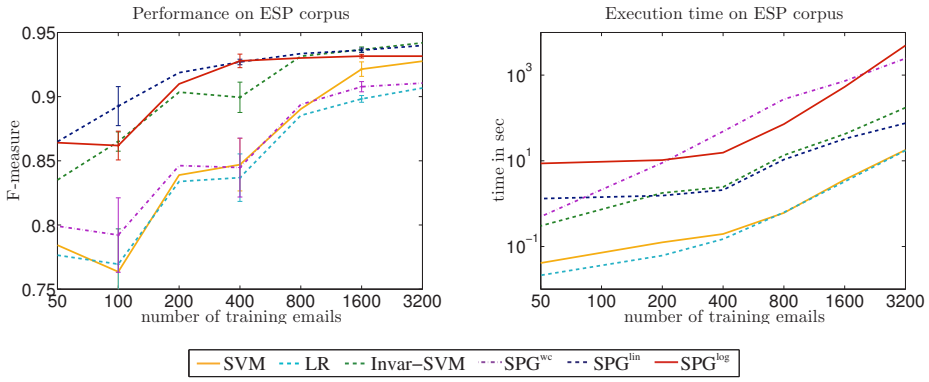


**Figure 5.2:** Predictive performance (left) and execution time (right) for varying sizes of the training data set.

The Stackelberg prediction game with logistic loss has more wins than it has losses against each of the other methods. The Stackelberg prediction game with linear loss has more wins than losses against each of the other methods except for the SPG with logistic loss. The ranking continues with the SPG with worst-case loss, the Invar-SVM, the regular support vector machine, and logistic regression which loses more frequently than it wins against all other methods.

### 5.6.2 Efficiency versus Effectiveness

To study the predictive performance as well as running time behavior with respect to the size of the data set, we train the baselines and the three SPG instances for a varying number of training examples. We report on the results for the representative ESP data set in Figure 5.2. Except for SPG$^{wc}$, the game models significantly outperform the trivial baseline methods SVM and logistic regression, especially for small corpus sizes. However, this comes at the price of considerably higher computational cost. For the game models, the Stackelberg instance SPG$^{lin}$ clearly outperforms all reference methods with respect to efficiency. Though, the larger the size of the data set, the stronger the computational differences where at the same time the discrepancy of the predictive performance diminishes.

### 5.6.3 Transformation

Similar to the experiment in Section 4.6.6, we finally study to which extend the training instances are transformed by the data generator. As for the Nash models, the Stackelberg prediction games allow the data generator to modify positive as well as negative instances. In practice, only positive instances are expected to be modified. We therefore study whether the Stackelberg model reflects this aspect of reality. Table 5.2 shows the average number of modifications—*i.e.,* word additions and deletions—performed by the sender per spam and per non-spam email depending on the sender's regularization parameter $\rho_{+1}$ for fixed $\rho_{-1}$.

**Table 5.2:** Average number of word additions and deletions per instance for $\text{SPG}^{\log}$.

| $\rho_{+1}$ | non-spam | | spam | |
|---|---|---|---|---|
| | *additions* | *deletions* | *additions* | *deletions* |
| 4 | 1.4 | 1.6 | 14.6 | 17.6 |
| 16 | 0.3 | 0.3 | 9.9 | 11.6 |
| 64 | 0.0 | 0.0 | 7.1 | 8.7 |
| 256 | 0.0 | 0.0 | 2.4 | 2.8 |
| 1024 | 0.0 | 0.0 | 0.8 | 0.9 |

Like for the Nash models, the number of transformations increases inversely proportional to the regularization parameter. Since the interests of sender and recipient are coherent for legitimate emails, these messages are rarely modified, even for equal cost factors $c_{v,i}$.

# 6 Covariate Shift

We finally study the setting where the learner acts *after* the data generator. In this case, the data generator first modifies the data generation process—potentially, in response to a previously released predictive model—and produces a sample of test instances. Only afterwards, the learner builds the predictive model from the past labeled training instances and the new unlabeled test instances. This model addresses settings where the prediction is performed with delay. For instance, an email service provider may apply spam blocking techniques when receiving an email, *e.g.,* IP blacklists. However, the accepted emails need only be classified when being fetched by the users. Since in practice, emails which are received over night are typically fetched in the morning, there is often a delay between the creation of the emails and the application of the predictive model. Hence, the emails which were received during this time can be used by the learner in the model-building process.

In general, we aim at finding an adversary-aware predictive model, that is, we are only interested in identifying an optimal move from the perspective of the learner. As the data generator's action is fixed when the learner gets to move, the game reduces to a special semi-supervised learning problem where the underlying probability distributions of the training and test data, $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}$ and $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$, may differ. This distributional shift is caused by the change of the data generation process from $\gamma$ to $\dot{\gamma}$.

For simplicity, we only consider the case where the data generator is *not* in control of the labeling process. In this case, the conditional distributions of $\boldsymbol{Y}$ are the same in the training as well as the test distribution. However, the marginal distributions of $\boldsymbol{X}$ may be different. This difference is called *covariate shift*. In our running example of email spam filtering, this restriction accounts for the fact that the class of an email is determined by the recipient, and not by the sender who generates the email.

In Section 6.1, we contribute a predictive model for learning under covariate shift. The model directly characterizes the divergence between training and test distribution, without the intermediate—intrinsically model-based—step of estimating training and test distribution. We formulate the search for all model parameters as an integrated optimization problem. This complements the predominant procedure of first estimating the bias of the training sample, and then learning the predictive model on an unbiased version of the training sample. In Section 6.2, we derive a gradient descent procedure to find a locally optimal solution of the integrated model, called *logistic regression importance estimation*. In Section 6.3 we state a two-stage approximation of this method which is conceptually simpler than the integrated model and may have the greatest practical utility. We further derive kernelized variants in Section 6.4. We review related models for differing training and test distributions in Section 6.5. Finally, we provide empirical results in Section 6.6.

## 6.1 Learning under Covariate Shift

As for Nash and Stackelberg prediction games, in the covariate shift problem setting a training sample $D = \{(x_i, y_i)\}_{i=1}^n$ with $\boldsymbol{x} := (x_1, \ldots, x_n) \in \mathcal{X}^n$ and $\boldsymbol{y} := (y_1, \ldots, y_n) \in \mathcal{Y}^n$ is available to the learner. These object-target pairs are governed by an unknown distribution $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}$, where $\gamma$ is the corresponding data generation model. We assume that the labels are drawn according to an unknown target concept which is *not* under the control of the data generator, *i.e.,* the joint training data distribution decomposes into

$$P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}(x, y) = P_{\boldsymbol{Y}|\boldsymbol{X}=x}(y) P_{\boldsymbol{X}|\boldsymbol{\Gamma}=\gamma}(x). \tag{6.1}$$

In contrast to the settings of the previous chapters, the learner also observes the data generator's chosen action, that is, the tuple of instances $\dot{\boldsymbol{x}} := (\dot{x}_1, \ldots, \dot{x}_l) \in \mathcal{X}^l$ (but not the corresponding tuple of target labels $\dot{\boldsymbol{y}} := (\dot{y}_1, \ldots, \dot{y}_l) \in \mathcal{Y}^l$) of test sample $\dot{D} = \{(\dot{x}_j, \dot{y}_j)\}_{j=1}^l$. Those pairs of test objects and targets are governed by an unknown test distribution

$$P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}(\dot{x}, \dot{y}) = P_{\boldsymbol{Y}|\boldsymbol{X}=\dot{x}}(\dot{y}) P_{\boldsymbol{X}|\boldsymbol{\Gamma}=\dot{\gamma}}(\dot{x}). \tag{6.2}$$

The adversarially modified data generation model $\dot{\gamma}$ may differ from $\gamma$, so that the training and test distribution may differ as well. However, there is only one unknown target distribution $P_{\boldsymbol{Y}|\boldsymbol{X}=x}$.

Hence, we consider the following data generation process: The data are generated either from the fixed data generation model $\gamma$ or $\dot{\gamma}$, which were determined by the data generator. The instances $x \in \mathcal{X}$ are drawn from $P_{\boldsymbol{X}|\boldsymbol{\Gamma}=\gamma}$ or $P_{\boldsymbol{X}|\boldsymbol{\Gamma}=\dot{\gamma}}$, respectively, where we assume that the training distribution covers the entire support of the test distribution. Finally, the corresponding targets $y \in \mathcal{Y}$ are drawn from $P_{\boldsymbol{Y}|\boldsymbol{X}=x}$. The following assumption summarizes these considerations.

**Assumption 6.1.** *The following statements hold:*

1. *The learner acts after the data generator;*
2. *The data generator has committed to action $\dot{D} = \{(\dot{x}_j, \dot{y}_j)\}_{j=1}^l$, where objects $\dot{x}_j$, but not the corresponding targets $\dot{y}_j$, are observable by the learner;*
3. *The training data $D$ are an i.i.d. sample of the training distribution $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}$ and the test data $\dot{D}$ are an i.i.d. sample of the test distribution $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$;*
4. *Training and test distribution factorize as in* (6.1) *and* (6.2)*, so that there is only one target distribution $P_{\boldsymbol{Y}|\boldsymbol{X}=x}$ and two covariate distributions with $P_{\boldsymbol{X}|\boldsymbol{\Gamma}=\dot{\gamma}}(x) > 0 \Rightarrow P_{\boldsymbol{X}|\boldsymbol{\Gamma}=\gamma}(x) > 0$, $\forall x \in \mathcal{X}$.*

The goal of the learner is to build a model $h : \mathcal{X} \to \mathcal{Y}$ which has optimal predictive performance for object-target pairs drawn from the test distribution $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$. To this end, we model the problem of covariate shift from a Bayesian perspective (*cf.* Section 2.2), that is, we aim at estimating the density $p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h^*}(y)$ where $h^*$ is the true underlying

relationship between test objects $x$ and targets $y$. One typically approximates this quantity by

$$p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{X}^n=\boldsymbol{x}',\boldsymbol{Y}^n=\boldsymbol{y}'}(y) \;\propto\; \int_H p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h}(y)\, p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x}',\boldsymbol{H}=h}(\boldsymbol{y}')\, p_{\boldsymbol{H}}(h)\,\mathrm{d}h,$$

where $\boldsymbol{x}' = (x'_1, \ldots, x'_n)$ and $\boldsymbol{y}' = (y'_1, \ldots, y'_n)$ form a sample of $n$ object-target pairs drawn from the same distribution as used to generate new data at application time, that is, test distribution $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$. As in (2.15), we again consider a point estimate, precisely the MAP estimate

$$h^{D'}_{\mathrm{MAP}} := \operatorname*{argmax}_{h\in H} p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x}',\boldsymbol{H}=h}(\boldsymbol{y}')\, p_{\boldsymbol{H}}(h), \tag{6.3}$$

to approximate the posterior of $h \in H$ by a single-point distribution so that the above integral reduces to $p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h^{D'}_{\mathrm{MAP}}}(y)$ (*cf.* Equation 2.16).

The conditional $p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x}',\boldsymbol{H}=h}(\boldsymbol{y}')$ in (6.3) denotes the label likelihood of the sample $(\boldsymbol{x}', \boldsymbol{y}') \in (\mathcal{X}\times\mathcal{Y})^n$ which is governed by $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$. It states the plausibility of model $h$ with respect to that sample. However, in contrast to the standard setting of identical training and test distributions, we have *not* observed such a sample, but a sample $(\boldsymbol{x}, \boldsymbol{y}) \in (\mathcal{X}\times\mathcal{Y})^n$ governed by $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}$. As training and test distribution may differ, the label likelihood of any finite sample $(\boldsymbol{x}', \boldsymbol{y}')$ will systematically differ from that of sample $(\boldsymbol{x}, \boldsymbol{y})$ as well, *i.e.,* substituting $(\boldsymbol{x}', \boldsymbol{y}')$ in (6.3) by $(\boldsymbol{x}, \boldsymbol{y})$ would yield an inconsistent label likelihood. Since the MAP estimate is the maximizer of the product of label likelihood and prior, this inconsistency will affect the resulting model.

### 6.1.1 MAP Estimation under Covariate Shift

To derive a MAP hypothesis based on a consistent label likelihood, we first analyze the theoretical quantity that the (normalized) label likelihood converges to, *i.e.,* the *theoretical label likelihood*. We state a consistent estimate of the theoretical label likelihood under $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$ based on a resampled data set that is governed by $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}$. Finally, we express the corresponding resampling weights in terms of the conditional $P_{\boldsymbol{\Gamma}|\boldsymbol{X}}$.

**Definition 6.1.** *For any joint data distribution $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$ with density $p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$, the* theoretical label likelihood *of model $h \in \mathcal{H}$ is defined by*

$$\mathrm{L}[h, p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}] := \exp \mathrm{E}_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}[\log p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h}(y)]. \tag{6.4}$$

**Lemma 6.1.** *For any finite sample $D' = \{(x'_i, y'_i)\}_{i=1}^n$ drawn i.i.d. from $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$ with corresponding empirical density $p_{D'}$ (cf. Equation 2.3), the theoretical label likelihood under $p_{D'}$ equals the per-instance label likelihood of $D'$, that is,*

$$\mathrm{L}[h, p_{D'}] = p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x}',\boldsymbol{H}=h}(\boldsymbol{y}')^{\frac{1}{n}}, \tag{6.5}$$

*where $\boldsymbol{x}' = (x'_1, \ldots, x'_n)$ and $\boldsymbol{y}' = (y'_1, \ldots, y'_n)$.*

*Proof.* Starting with (6.4),

$$\begin{aligned} \mathrm{L}[h, p_{D'}] &= \exp \mathrm{E}_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}[\log p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h}(y)] \\ &= \exp \sum_{i=1}^{n} \frac{1}{n} \log p_{\boldsymbol{Y}|\boldsymbol{X}=x_i',\boldsymbol{H}=h}(y_i') \\ &= \prod_{i=1}^{n} p_{\boldsymbol{Y}|\boldsymbol{X}=x_i',\boldsymbol{H}=h}(y_i')^{\frac{1}{n}} \\ &= p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x}',\boldsymbol{H}=h}(\boldsymbol{y}')^{\frac{1}{n}}, \end{aligned}$$

Lemma 6.1 follows. □

By this lemma and the strong law of large numbers (*cf.* Chapter 2 in [73]), the per-instance label likelihood converges almost surely to the theoretical label likelihood,

$$\mathrm{L}[h, p_{D'}] \xrightarrow{a.s.} \mathrm{L}[h, p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}],$$

where data sample $D'$ is governed by $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$. That is, the per-instance label likelihood $p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x}',\boldsymbol{H}=h}(\boldsymbol{y}')^{\frac{1}{n}}$ for an infinite data sample drawn from $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$ approaches the theoretical label likelihood. Apart from normalization, the label likelihood in (6.3) establishes a consistent estimate of the theoretical label likelihood. Theoretically, we could replace the label likelihood by its expectation,

$$h^{\dot{\gamma}}_{\mathrm{MAP}} := \operatorname*{argmax}_{h \in H} \mathrm{L}[h, p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}]^n p_{\boldsymbol{H}}(h), \tag{6.6}$$

which would yield an ideal MAP estimate.

**Lemma 6.2.** *For any joint data distributions $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}$ and $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$ with density functions $p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}$ and $p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$, respectively, equation*

$$\mathrm{L}[h, p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}] = \exp \mathrm{E}_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma} \left[\kappa(x) \log p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h}(y)\right] \tag{6.7}$$

*holds for*

$$\kappa(x) := \frac{p_{\boldsymbol{X}|\boldsymbol{\Gamma}=\dot{\gamma}}(x)}{p_{\boldsymbol{X}|\boldsymbol{\Gamma}=\gamma}(x)}. \tag{6.8}$$

*Proof.* By Assumption 6.1.4 and the definition of $\kappa$ in (6.8),

$$\kappa(x) = \frac{p_{\boldsymbol{Y}|\boldsymbol{X}=x}(y) p_{\boldsymbol{X}|\boldsymbol{\Gamma}=\dot{\gamma}}(x)}{p_{\boldsymbol{Y}|\boldsymbol{X}=x}(y) p_{\boldsymbol{X}|\boldsymbol{\Gamma}=\gamma}(x)} = \frac{p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}(x,y)}{p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}(x,y)}$$

holds for all $y \in \mathcal{Y}$. These ratios are well-defined for all training instances, as $p_{\boldsymbol{X}|\boldsymbol{\Gamma}=\dot{\gamma}}(x) > 0$ holds for all training instances, and by Assumption 6.1.4, this implies $p_{\boldsymbol{X}|\boldsymbol{\Gamma}=\gamma}(x) > 0$.

Since

$$\begin{aligned}
\mathrm{L}[h, p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}] &= \exp \int_{\mathcal{X}\times\mathcal{Y}} p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}(x,y) \log p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h}(y)\,\mathrm{d}(x,y) \\
&= \exp \int_{\mathcal{X}\times\mathcal{Y}} p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}(x,y) \frac{p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}(x,y)}{p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}(x,y)} \log p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h}(y)\,\mathrm{d}(x,y) \\
&= \exp \int_{\mathcal{X}\times\mathcal{Y}} p_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}(x,y)\kappa(x) \log p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h}(y)\,\mathrm{d}(x,y) \\
&= \exp \mathrm{E}_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}\left[\kappa(x) \log p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h}(y)\right],
\end{aligned}$$

Lemma 6.2 holds. □

Lemma 6.2 links the theoretical label likelihood under test distribution $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$ to the training distribution $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}$. By making again use of the strong law of large numbers, we can consistently estimate the expectation in (6.7) by the empirical mean with respect to any sample $D = \{(x_i, y_i)\}_{i=1}^n$ governed by $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}$, that is,

$$\frac{1}{n}\sum_{i=1}^{n} \kappa(x_i) \log p_{\boldsymbol{Y}|\boldsymbol{X}=x_i,\boldsymbol{H}=h}(y_i) \xrightarrow{a.s.} \mathrm{E}_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\gamma}\left[\kappa(x) \log p_{\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{H}=h}(y)\right],$$

where mapping $\kappa$ is defined as in (6.8). By applying Lemma 6.2, we observe that

$$h^D_{\mathrm{MAP}} := \operatorname*{argmax}_{h\in H} \left(\prod_{i=1}^{n} p_{\boldsymbol{Y}|\boldsymbol{X}=x_i,\boldsymbol{H}=h}(y_i)^{\kappa(x_i)}\right) p_{\boldsymbol{H}}(h) \tag{6.9}$$

is, such as $h^{\dot{\gamma}}_{\mathrm{MAP}}$, a MAP hypothesis based on a consistent estimate of the label likelihood.

Intuitively, *resampling weight* $\kappa_i := \kappa(x_i)$ dictates how many times, on average, object-target pair $(x_i, y_i)$ of sample $D$ should occur in the training sample if it was governed by the test distribution $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$. The product $\prod_{i=1}^n p_{\boldsymbol{Y}|\boldsymbol{X}=x_i,\boldsymbol{H}=h}(y_i)^{\kappa(x_i)}$ states the label likelihood of such a resampled data set.

To compute $h^D_{\mathrm{MAP}}$, we finally discuss how to estimate the resampling weights $\kappa_i$ for $i = 1, \ldots, n$ from the available data. Starting with the definition of $\kappa(x)$ in (6.8), we apply Bayes' rule twice. In (6.11) we use the fact that the data were generated either by model $\gamma$ or $\dot{\gamma}$, so that $p_{\boldsymbol{\Gamma}|\boldsymbol{X}=x}(\dot{\gamma}) = 1 - p_{\boldsymbol{\Gamma}|\boldsymbol{X}=x}(\gamma)$.

$$\begin{aligned}
\kappa(x) &= \frac{p_{\boldsymbol{\Gamma}|\boldsymbol{X}=x}(\dot{\gamma}) p_{\boldsymbol{X}}(x)}{p_{\boldsymbol{\Gamma}|\boldsymbol{X}=x}(\gamma) p_{\boldsymbol{X}}(x)} \frac{p_{\boldsymbol{\Gamma}}(\gamma)}{p_{\boldsymbol{\Gamma}}(\dot{\gamma})} && (6.10) \\
&= \frac{p_{\boldsymbol{\Gamma}}(\gamma)}{p_{\boldsymbol{\Gamma}}(\dot{\gamma})} \left(\frac{1}{p_{\boldsymbol{\Gamma}|\boldsymbol{X}=x}(\gamma)} - 1\right) && (6.11)
\end{aligned}$$

In the above equation, ratio $\frac{p_{\boldsymbol{\Gamma}}(\gamma)}{p_{\boldsymbol{\Gamma}}(\dot{\gamma})}$ states the normalization factor which ensures that

$$\mathrm{E}_{\boldsymbol{X}|\boldsymbol{\Gamma}=\gamma}[\kappa(x)] = \int_{\mathcal{X}} p_{\boldsymbol{X}|\boldsymbol{\Gamma}=\gamma}\kappa(x)\,\mathrm{d}x = \int_{\mathcal{X}} p_{\boldsymbol{X}|\boldsymbol{\Gamma}=\dot{\gamma}}\,\mathrm{d}x = 1.$$

Consequently, for any finite sample the non-negative exponents

$$\kappa(x_i) \propto \frac{1}{p_{\boldsymbol{\Gamma}|\boldsymbol{X}=x_i}(\gamma)} - 1 \tag{6.12}$$

in (6.9) only need to be scaled so that they sum up to $n$, that is, the empirical mean of these resampling weights equals 1.

The significance of Equation 6.11 is that it shows how the optimal resampling weights, the testing-to-training ratio $\kappa(x)$, can be determined without the knowledge of either training or test density: The right-hand side of (6.11) can be evaluated based on a model that discriminates training against test objects and outputs how much more likely an instance is to occur in the test data than in the training data. Hence, instead of potentially high-dimensional densities $p_{\boldsymbol{X}|\boldsymbol{\Gamma}=\gamma}$ and $p_{\boldsymbol{X}|\boldsymbol{\Gamma}=\dot{\gamma}}$, a single conditional $p_{\boldsymbol{\Gamma}|\boldsymbol{X}=x}(\gamma)$ of the binary random variable $\boldsymbol{\Gamma}$ needs to be modeled.

### 6.1.2 An Integrated Model

In the previous section, we have derived a model for covariate shift which requires the knowledge of the conditional $p_{\boldsymbol{\Gamma}|\boldsymbol{X}=x}$. Hence, besides modeling the label likelihood we need to formulate this model likelihood. To this end, we introduce the random variable $\boldsymbol{Z}$ which can take $z = -1$ and $z = +1$, and the *selector function* $s^*$ with

$$\begin{aligned} p_{\boldsymbol{Z}|\boldsymbol{X}=x,\boldsymbol{S}=s^*}(+1) &:= p_{\boldsymbol{\Gamma}|\boldsymbol{X}=x}(\gamma), \\ p_{\boldsymbol{Z}|\boldsymbol{X}=x,\boldsymbol{S}=s^*}(-1) &:= p_{\boldsymbol{\Gamma}|\boldsymbol{X}=x}(\dot{\gamma}). \end{aligned}$$

Similar as hypothesis $h^* : \mathcal{X} \to \mathcal{Y}$ explains the true relation between objects $\boldsymbol{X}$ and targets $\boldsymbol{Y}$, the selector function $s^* : \mathcal{X} \to \{-1, +1\}$ explains the underlying relation between objects $\boldsymbol{X}$ and the data generation model induced by $\boldsymbol{Z}$.

This refinement enables us to express data samples $D$ and $\dot{D}$ by triple $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, where we redefine

- $\boldsymbol{x} := (x_1, \ldots, x_n, \dot{x}_1, \ldots, \dot{x}_l) \in \mathcal{X}^{n+l}$ as a concatenation of training and test objects;
- $\boldsymbol{y} := (y_1, \ldots, y_n, \dot{y}_1, \ldots, \dot{y}_l) \in \mathcal{Y}^{n+l}$, where $y_i$ are the targets of the training objects and $\dot{y}_i$ are the (unknown) targets of the test objects;
- $\boldsymbol{z} := (z_1, \ldots, z_{n+l}) \in \{-1, +1\}^{n+l}$ where $z_i := +1$ for $i = 1, \ldots, n$ and $z_i := -1$ for $i = n+1, \ldots, n+l$.

By these definitions, tuple $(x_i, y_i, z_i)$ states a realization of the joint random variable $\boldsymbol{XYZ}$, that is, any object $x_i$ from the training or test sample, the corresponding target $y_i$ which is observed for $i = 1, \ldots, n$, and the origin $z_i$ of that object, that is, whether the instance is drawn from the training or the test distribution. Vectors $\boldsymbol{x}_k$, $\boldsymbol{y}_k$, and $\boldsymbol{z}_k$ denote the first $1 \leq k \leq n + l$ elements of $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$, respectively.

Having refined the notation, we now seek to estimate $h^*$ and $s^*$. In the predominant two-step procedure for learning under covariate shift (*cf.* Section 6.5), one typically first

estimates the resampling weights, for instances, by inferring a point estimate $\hat{s}$ from the posterior $p_{\boldsymbol{S}|\boldsymbol{X}^{n+l}=\boldsymbol{x},\boldsymbol{Z}^{n+l}=\boldsymbol{z}}(s)$ and computing the resampling weights as in (6.12). Only then, one estimates $h^*$ from $p_{\boldsymbol{H}|\boldsymbol{X}^n=\boldsymbol{x}_n,\boldsymbol{Y}^n=\boldsymbol{y}_n,\boldsymbol{S}=\hat{s}}(h)$ based on the resampled training instances as discussed in the previous section. We present this sequential estimation approach in detail in Section 6.3.

However, the sequential estimation amounts to an additional degree of approximation since $s^*$ can only be estimated up to a certain amount. This additional error will propagate to the estimation problem of $h^*$ for which reason we directly address the conditional

$$p_{\boldsymbol{YZ}|\boldsymbol{X}=x,\boldsymbol{X}^{n+l}=\boldsymbol{x},\boldsymbol{Y}^n=\boldsymbol{y}_n,\boldsymbol{Z}^{n+l}=\boldsymbol{z}}(y,z) \tag{6.13}$$

$$= \int_{H\times S} p_{\boldsymbol{YZ}|\boldsymbol{X}=x,\boldsymbol{H}=h,\boldsymbol{S}=s}(y,z)\, p_{\boldsymbol{HS}|\boldsymbol{X}^{n+l}=\boldsymbol{x},\boldsymbol{Y}^n=\boldsymbol{y}_n,\boldsymbol{Z}^{n+l}=\boldsymbol{z}}(h,s)\,\mathrm{d}(h,s) \tag{6.14}$$

to jointly estimate $h^*$ and $s^*$. As before, we focus on the MAP estimate

$$(h_{\mathrm{MAP}}, s_{\mathrm{MAP}}) := \operatorname*{argmax}_{h\in H, s\in S} p_{\boldsymbol{HS}|\boldsymbol{X}^{n+l}=\boldsymbol{x},\boldsymbol{Y}^n=\boldsymbol{y}_n,\boldsymbol{Z}^{n+l}=\boldsymbol{z}}(h,s)$$

to approximate the above integral.

The posterior of $(h,s)$ factorizes as in Equation 6.15. Here, we exploit the facts that $h$ is independent of the test objects $x_{n+1},\ldots,x_{n+l}$, since targets $y_{n+1},\ldots,y_{n+l}$ are unobserved, $h$ is independent of the selector variables $z_i$ given $s$, and by Assumption 6.1.4, $s$ is independent of the targets. In (6.16) we apply Bayes' rule twice, and in (6.17) we use the fact that targets $y$ are independent of $s$ for given $x$. Equation 6.18 follows as both denominators in (6.17) are constant for the given data, $h$ is again independent of $s$ and $x$ for unobserved $y$, and $s$ is independent of $x$ for unobserved $z$.

$$p_{\boldsymbol{HS}|\boldsymbol{X}^{n+l}=\boldsymbol{x},\boldsymbol{Y}^n=\boldsymbol{y}_n,\boldsymbol{Z}^{n+l}=\boldsymbol{z}}(h,s)$$

$$= p_{\boldsymbol{H}|\boldsymbol{X}^n=\boldsymbol{x}_n,\boldsymbol{Y}^n=\boldsymbol{y}_n,\boldsymbol{S}=s}(h)\, p_{\boldsymbol{S}|\boldsymbol{X}^{n+l}=\boldsymbol{x},\boldsymbol{Z}^{n+l}=\boldsymbol{z}}(s) \tag{6.15}$$

$$= \frac{p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x}_n,\boldsymbol{H}=h,\boldsymbol{S}=s}(\boldsymbol{y}_n)\, p_{\boldsymbol{X}^n\boldsymbol{HS}}(\boldsymbol{x}_n,h,s)}{p_{\boldsymbol{X}^n\boldsymbol{Y}^n\boldsymbol{S}}(\boldsymbol{x}_n,\boldsymbol{y}_n,s)} \frac{p_{\boldsymbol{Z}^{n+l}|\boldsymbol{X}^{n+l}=\boldsymbol{x},\boldsymbol{S}=s}(\boldsymbol{z})\, p_{\boldsymbol{X}^{n+l}\boldsymbol{S}}(\boldsymbol{x},s)}{p_{\boldsymbol{X}^{n+l}\boldsymbol{Z}^{n+l}}(\boldsymbol{x},\boldsymbol{z})} \tag{6.16}$$

$$= \frac{p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x}_n,\boldsymbol{H}=h,\boldsymbol{S}=s}(\boldsymbol{y}_n)\, p_{\boldsymbol{H}|\boldsymbol{X}^n=\boldsymbol{x}_n,\boldsymbol{S}=s}(h)}{p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x}_n}(\boldsymbol{y}_n)} \frac{p_{\boldsymbol{Z}^{n+l}|\boldsymbol{X}^{n+l}=\boldsymbol{x},\boldsymbol{S}=s}(\boldsymbol{z})\, p_{\boldsymbol{S}|\boldsymbol{X}^{n+l}=\boldsymbol{x}}(s)}{p_{\boldsymbol{Z}^{n+l}|\boldsymbol{X}^{n+l}=\boldsymbol{x}}(\boldsymbol{z})} \tag{6.17}$$

$$\propto p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x}_n,\boldsymbol{H}=h,\boldsymbol{S}=s}(\boldsymbol{y}_n)\, p_{\boldsymbol{H}}(h)\, p_{\boldsymbol{Z}^{n+l}|\boldsymbol{X}^{n+l}=\boldsymbol{x},\boldsymbol{S}=s}(\boldsymbol{z})\, p_{\boldsymbol{S}}(s) \tag{6.18}$$

In the right-hand side of (6.18), the first factor denotes the label likelihood of the resampled data set (*cf.* Equation 6.9 and 6.12) with

$$p_{\boldsymbol{Y}^n|\boldsymbol{X}^n=\boldsymbol{x}_n,\boldsymbol{H}=h,\boldsymbol{S}=s}(\boldsymbol{y}_n) = \left(\prod_{i=1}^n p_{\boldsymbol{Y}|\boldsymbol{X}=x_i,\boldsymbol{H}=h}(y_i)^{\left(\frac{1}{p_{\boldsymbol{Z}|\boldsymbol{X}=x_i,\boldsymbol{S}=s}(+1)}-1\right)}\right)^{\tau(s)},$$

where the exponent $\tau(s)$ is a normalizer with $\tau(s)^{-1} := \frac{1}{n}\sum_{i=1}^n \left(\frac{1}{p_{\boldsymbol{Z}|\boldsymbol{X}=x_i,\boldsymbol{S}=s}(+1)}-1\right)$.

The second and fourth term are the priors of models $h$ and $s$, respectively, and the third factor states the model likelihood,

$$p_{\boldsymbol{Z}^{n+l}|\boldsymbol{X}^{n+l}=\boldsymbol{x},\boldsymbol{S}=s}(\boldsymbol{z}) = \prod_{i=1}^{n+l} p_{\boldsymbol{Z}|\boldsymbol{X}=x_i,\boldsymbol{S}=s}(z_i).$$

When inserting those likelihoods in (6.18), we arrive at

$$\begin{aligned} &p_{\boldsymbol{HS}|\boldsymbol{X}^{n+l}=\boldsymbol{x},\boldsymbol{Y}^{n}=\boldsymbol{y}_n,\boldsymbol{Z}^{n+l}=\boldsymbol{z}}(h,s) \\ &\quad\propto \left(\prod_{i=1}^{n} p_{\boldsymbol{Y}|\boldsymbol{X}=x_i,\boldsymbol{H}=h}(y_i)^{\left(\frac{1}{p_{\boldsymbol{Z}|\boldsymbol{X}=x_i,\boldsymbol{S}=s}(+1)}-1\right)}\right)^{\tau(s)} p_{\boldsymbol{H}}(h) \left(\prod_{i=1}^{n+l} p_{\boldsymbol{Z}|\boldsymbol{X}=x_i,\boldsymbol{S}=s}(z_i)\right) p_{\boldsymbol{S}}(s). \end{aligned}$$

In the next chapter we model $h$ and $s$ in terms of linear decision functions such as discussed in Chapter 2.3, and consider logistic models for their corresponding likelihood functions. We derive a gradient descent-based algorithm to infer $h_{\text{MAP}}$ and $s_{\text{MAP}}$ from the data based on the above expression of the joint posterior.

## 6.2 Logistic Regression Importance Estimation

As presented in Chapter 2.3, we use linear decision functions to model the predictive model $h : \mathcal{X} \to \mathcal{Y}$ with $\mathcal{Y} = \{-1, +1\}$ and the selector model $s : \mathcal{X} \to \{-1, +1\}$. To this end, we define $f^h_{\mathbf{w}}(x, y) := y\mathbf{w}^\mathsf{T}\boldsymbol{\phi}^h(x)$ and $f^s_{\mathbf{v}}(x, z) := z\mathbf{v}^\mathsf{T}\boldsymbol{\phi}^s(x)$ with the corresponding hypotheses,

$$\begin{aligned} h(x) &:= \operatorname*{argmax}_{y' \in \{-1,+1\}} f^h_{\mathbf{w}}(x, y') = \operatorname{sign} \mathbf{w}^\mathsf{T}\boldsymbol{\phi}^h(x), \\ s(x) &:= \operatorname*{argmax}_{z' \in \{-1,+1\}} f^s_{\mathbf{v}}(x, z') = \operatorname{sign} \mathbf{v}^\mathsf{T}\boldsymbol{\phi}^s(x). \end{aligned}$$

The potentially distinct functions $\boldsymbol{\phi}^h(x)$ and $\boldsymbol{\phi}^s(x)$ map instance $x$ into the corresponding Hilbert space (*cf.* Section 2.6). The weight vectors $\mathbf{w}$ and $\mathbf{v}$, together with the above definition, fully define the models $h$ and $s$, respectively. Consequently, we can state the joint posterior of $(h, s)$ in terms of those parameter vectors. The corresponding MAP parameters $\mathbf{w}_{\text{MAP}}$ and $\mathbf{v}_{\text{MAP}}$ are the maximizing arguments of

$$\max_{\mathbf{w}\in\mathcal{W},\mathbf{v}\in\mathcal{V}} \left(\prod_{i=1}^{n} p_{\boldsymbol{Y}|\boldsymbol{X}=x_i,\boldsymbol{W}=\mathbf{w}}(y_i)^{\kappa_{\mathbf{v}}(x_i)}\right) p_{\boldsymbol{W}}(\mathbf{w}) \left(\prod_{i=1}^{n+l} p_{\boldsymbol{Z}|\boldsymbol{X}=x_i,\boldsymbol{V}=\mathbf{v}}(z_i)\right) p_{\boldsymbol{V}}(\mathbf{v}), \tag{6.19}$$

with

$$\kappa_{\mathbf{v}}(x_i) = \left(\frac{1}{p_{\boldsymbol{Z}|\boldsymbol{X}=x_i,\boldsymbol{V}=\mathbf{v}}(+1)} - 1\right) \tau(\mathbf{v})$$

and $\tau(\mathbf{v})^{-1} = \frac{1}{n}\sum_{i=1}^{n} (\frac{1}{p_{\boldsymbol{Z}|\boldsymbol{X}=x_i,\boldsymbol{V}=\mathbf{v}}(+1)} - 1)$. To solve for $(\mathbf{w}_{\text{MAP}}, \mathbf{v}_{\text{MAP}})$, we need to specify the label likelihood, the model likelihood, and the priors in (6.19). For the partial likelihoods

we choose the logistic function, which gives

$$p_{\boldsymbol{Y}|\boldsymbol{X}=x_i,\boldsymbol{W}=\mathbf{w}}(y_i) := \frac{1}{1+\exp(-y_i\mathbf{w}^\mathsf{T}\boldsymbol{\phi}^h(x_i))}, \tag{6.20}$$

$$p_{\boldsymbol{Z}|\boldsymbol{X}=x_i,\boldsymbol{V}=\mathbf{v}}(z_i) := \frac{1}{1+\exp(-z_i\mathbf{v}^\mathsf{T}\boldsymbol{\phi}^s(x_i))}. \tag{6.21}$$

This choice corresponds to the logistic loss (Definition 2.6) and a label likelihood function of the exponential family such as in (2.17). By the above definitions, $\kappa_\mathbf{v}$ resolves to

$$\kappa_\mathbf{v}(x_i) = \exp(-\mathbf{v}^\mathsf{T}\boldsymbol{\phi}^s(x_i))\tau(\mathbf{v}) \qquad \text{where} \qquad \tau(\mathbf{v}) = \frac{n}{\sum_{i=1}^n \exp(-\mathbf{v}^\mathsf{T}\boldsymbol{\phi}^s(x_i))}.$$

In addition, we use Gaussian priors

$$p_{\boldsymbol{W}}(\mathbf{w}) := \mathcal{N}\left(\mathbf{w}\,|\,\mathbf{0},\rho_\mathbf{w}^{-1}\mathbf{I}_{m^h}\right) \qquad \text{and} \qquad p_{\boldsymbol{V}}(\mathbf{v}) := \mathcal{N}\left(\mathbf{v}\,|\,\mathbf{0},\rho_\mathbf{v}^{-1}\mathbf{I}_{m^s}\right),$$

where $m^h$ and $m^s$ denote the dimensionalities of the weight vectors, and $\rho_\mathbf{w}$ and $\rho_\mathbf{v}$ denote the regularization parameters. When taking the negative logarithm of the objective in 6.19 and inserting the above choices we arrive at

$$(\mathbf{w}_{\text{MAP}},\mathbf{v}_{\text{MAP}}) = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W},\mathbf{v}\in\mathcal{V}} \mathrm{Q}(\mathbf{w},\mathbf{v}) \tag{6.22}$$

with

$$\begin{aligned}\mathrm{Q}(\mathbf{w},\mathbf{v}) := {} & \tau(\mathbf{v})\sum_{i=1}^n \exp(-\mathbf{v}^\mathsf{T}\boldsymbol{\phi}^s(x_i))\,\log\left(1+\exp(-y_i\mathbf{w}^\mathsf{T}\boldsymbol{\phi}^h(x_i))\right) + \\ & \sum_{j=1}^{n+l}\log\left(1+\exp(-z_j\mathbf{v}^\mathsf{T}\boldsymbol{\phi}^s(x_j))\right) + \frac{\rho_\mathbf{w}}{2}\mathbf{w}^\mathsf{T}\mathbf{w} + \frac{\rho_\mathbf{v}}{2}\mathbf{v}^\mathsf{T}\mathbf{v}.\end{aligned} \tag{6.23}$$

In the above expression, normalizer $\tau(\mathbf{v})$ balances the trade-off between the weighted label likelihood (first sum) and the model likelihood (second sum). The impact of the $l^2$-regularizers is controlled by the corresponding regularization parameters $\rho_\mathbf{w}$ and $\rho_\mathbf{v}$.

To solve the optimization problem in (6.22), we propose a gradient descent method with inexact line search. To this end, we state the gradient of the objective in (6.23), that is, $\nabla\mathrm{Q}(\mathbf{w},\mathbf{v})^\mathsf{T} = [\nabla_\mathbf{w}\mathrm{Q}(\mathbf{w},\mathbf{v})^\mathsf{T}, \nabla_\mathbf{v}\mathrm{Q}(\mathbf{w},\mathbf{v})^\mathsf{T}]$. A straightforward calculation gives the partial gradients,

$$\begin{aligned}\nabla_\mathbf{w}\mathrm{Q}(\mathbf{w},\mathbf{v}) &= -\sum_{i=1}^n \kappa_\mathbf{v}(x_i)\frac{1}{1+\exp(y_i\mathbf{w}^\mathsf{T}\boldsymbol{\phi}^h(x_i))}y_i\boldsymbol{\phi}^h(x_i) + \rho_\mathbf{w}\mathbf{w} \\ \nabla_\mathbf{v}\mathrm{Q}(\mathbf{w},\mathbf{v}) &= -\sum_{i=1}^n \kappa_\mathbf{v}(x_i)\log\left(1+\exp(-y_i\mathbf{w}^\mathsf{T}\boldsymbol{\phi}^h(x_i))\right)\boldsymbol{\phi}_\mathbf{v}^s(x_i) + \\ &\quad \sum_{j=1}^{n+l}\frac{1}{1+\exp(z_j\mathbf{v}^\mathsf{T}\boldsymbol{\phi}^s(x_j))}z_j\boldsymbol{\phi}^s(x_j) + \rho_\mathbf{v}\mathbf{v}\end{aligned}$$

where $\kappa_{\mathbf{v}}(x_i)$ is defined as above, and we define $\boldsymbol{\phi}^s_{\mathbf{v}}(x_i) := \boldsymbol{\phi}^s(x_i) - \frac{1}{n}\sum_{j=1}^{n} \kappa_{\mathbf{v}}(x_j)\boldsymbol{\phi}^s(x_j)$. Based on these partial gradients, we can now state an optimization algorithm to find $\mathbf{w}_{\text{MAP}}$ and $\mathbf{v}_{\text{MAP}}$.

**Algorithm 4** LORIE: Logistic Regression Importance Estimation

**Require:** Objective $\mathrm{Q}(\mathbf{w}, \mathbf{v})$ as defined in (6.23).

1: Select initial $\mathbf{w}^{(0)} := \mathbf{0}$, $\mathbf{v}^{(0)} := \mathbf{0}$, set $k := 0$, and select $\sigma \in (0,1)$ and $\beta \in (0,1)$.
2: **repeat**
3: Set $\nu_i^{(k)} := \exp\left(-y_i \mathbf{w}^{(k)\mathsf{T}} \boldsymbol{\phi}^h(x_i)\right)$ for all $i = 1, \ldots, n$.
4: Set $\mu_j^{(k)} := \exp\left(-z_j \mathbf{v}^{(k)\mathsf{T}} \boldsymbol{\phi}^s(x_j)\right)$ for all $j = 1, \ldots, n+l$.
5: Set $e^{(k)} := \sum_{i=1}^{n} \kappa_{\mathbf{v}}(x_i) \log(1 + \nu_i^{(k)})$.
6: Set $\mathbf{d}_{\mathbf{w}}^{(k)} := \sum_{i=1}^{n} \kappa_{\mathbf{v}}(x_i) \frac{\nu_i^{(k)}}{1+\nu_i^{(k)}} y_i \boldsymbol{\phi}^h(x_i) - \rho_{\mathbf{w}} \mathbf{w}^{(k)}$.
7: Set $\mathbf{d}_{\mathbf{v}}^{(k)} := \sum_{i=1}^{n} \kappa_{\mathbf{v}}(x_i) \left(\log(1 + \nu_i^{(k)}) - e^{(k)}\right) \boldsymbol{\phi}^s(x_i) - \sum_{j=1}^{n+l} \frac{\mu_j^{(k)}}{1+\mu_j^{(k)}} z_j \boldsymbol{\phi}^s(x_j) - \rho_{\mathbf{v}} \mathbf{v}^{(k)}$.
8: Find maximal step size $t^{(k)} \in \left\{\beta^l \mid l \in \mathbb{N}\right\}$ with

$$\mathrm{Q}\left(\mathbf{w}^{(k)}, \mathbf{v}^{(k)}\right) - \mathrm{Q}\left(\mathbf{w}^{(k)} + t^{(k)}\mathbf{d}_{\mathbf{w}}^{(k)}, \mathbf{v}^{(k)} + t^{(k)}\mathbf{d}_{\mathbf{v}}^{(k)}\right) \geq \sigma\, t^{(k)} \left(\left\|\mathbf{d}_{\mathbf{w}}^{(k)}\right\|_2^2 + \left\|\mathbf{d}_{\mathbf{v}}^{(k)}\right\|_2^2\right).$$

9: Set $\mathbf{w}^{(k+1)} := \mathbf{w}^{(k)} + t^{(k)}\mathbf{d}_{\mathbf{w}}^{(k)}$.
10: Set $\mathbf{v}^{(k+1)} := \mathbf{v}^{(k)} + t^{(k)}\mathbf{d}_{\mathbf{v}}^{(k)}$.
11: Set $k := k + 1$.
12: **until** $\|\mathbf{w}^{(k)} - \mathbf{w}^{(k-1)}\|_2^2 + \|\mathbf{v}^{(k)} - \mathbf{v}^{(k-1)}\|_2^2 \leq \epsilon$.

**Remark 6.3.** In general, mapping 6.23 is not jointly convex in $\mathbf{w}$ and $\mathbf{v}$: If $\mathrm{Q}(\mathbf{w}, \mathbf{v})$ is convex for arbitrary instances $x_1, \ldots, x_{n+l} \in \mathcal{X}$ with targets $y_1, \ldots, y_n \in \mathcal{Y} = \{-1, +1\}$ and selectors $z_1, \ldots, z_{n+l} \in \{-1, +1\}$, then $\mathrm{Q}(\mathbf{w}, \mathbf{v})$ must also be convex in $\mathbf{v}$ (with fixed $\mathbf{w}$) for data sets which contain exactly two labeled instances. As $\mathbf{w}$ is assumed to be fixed, the log-terms of the first sum in 6.23 are constant as well. Hence, a necessary condition for the general convexity of Q is that the sum

$$c_1 \frac{\exp(-\mathbf{v}^{\mathsf{T}}\boldsymbol{\phi}^s(x_1))}{\exp(-\mathbf{v}^{\mathsf{T}}\boldsymbol{\phi}^s(x_1)) + \exp(-\mathbf{v}^{\mathsf{T}}\boldsymbol{\phi}^s(x_2))} + c_2 \frac{\exp(-\mathbf{v}^{\mathsf{T}}\boldsymbol{\phi}^s(x_2))}{\exp(-\mathbf{v}^{\mathsf{T}}\boldsymbol{\phi}^s(x_1)) + \exp(-\mathbf{v}^{\mathsf{T}}\boldsymbol{\phi}^s(x_2))} \tag{6.24}$$

is convex in $\mathbf{v}$ for any fixed $x_1, x_2 \in \mathcal{X}$ and $c_1, c_2 > 0$. As the inner mapping $\mathbf{v}^{\mathsf{T}}\boldsymbol{\phi}^s(x)$ is linear in $\mathbf{v}$, this condition is equal to require that $c_1 \frac{1}{1+\exp(a)} + c_2 \frac{1}{1+\exp(-a)}$ is convex in $a \in \mathbb{R}$. The first derivate of this term is $(c_2 - c_1)\frac{1}{1+\exp(a)}\frac{1}{1+\exp(-a)}$ and the second derivate is

$$(c_2 - c_1)\frac{1}{1+\exp(a)}\frac{1}{1+\exp(-a)}\frac{1-\exp(a)}{1+\exp(a)}, \tag{6.25}$$

which cannot be non-negative for all $a \in \mathbb{R}$ and distinct positive constants $c_1$ and $c_2$.

Consequently, Equation 6.24 is non-convex in $\mathbf{v}$ and thus $\mathrm{Q}(\mathbf{w}, \mathbf{v})$ is generally non-convex in $\mathbf{w}, \mathbf{v}$ as well. ◇

The algorithm converges as the objective Q is continuous in all $\mathbf{w} \in \mathbb{R}^{m^h}$ and $\mathbf{v} \in \mathbb{R}^{m^s}$ and lower bounded by zero. However, the method does not necessarily obtain a globally optimal solution as Q is generally *not* jointly convex in $\mathbf{w}$ and $\mathbf{v}$ (*cf.* Remark 6.3). Thus, the benefit from jointly estimating all parameters and thereby reducing the level of approximation may vanish. To this end, we study a two-step approximation of this algorithm in the following section, where a global optimal solution in each stage can be found efficiently.

## 6.3 A Two-Stage Approximation

The previous section describes a complete solution to the learning problem under covariate shift. Unfortunately, as discussed in Remark 6.3, the convexity of the underlying optimization criterion cannot be guaranteed and, consequently, the found solution is generally *not* globally optimal. Furthermore, the logistic regression classifier is deeply embedded into the objective so that it would not be easy to replace it by a different type of classifier like, for instance, a decision tree.

We will now discuss a slight approximation to the integrated model (LORIE) which solves two consecutive optimization problems. The first optimization problem produces example-specific weights and the second step generates a classifier from the weighted examples. Both optimization problems are convex, not only for the logistic model. But most significantly, the two-stage approximation is conceptually simple: The second optimization step can be carried out by any learning procedure and, as a result of the decomposition into two optimization problems, parameter tuning becomes much easier because cross-validation can be used in the both stages.

The derivation in Section 6.1.2 approximates the integral in (6.13) by simultaneously selecting a pair of MAP hypotheses $(h_{\mathrm{MAP}}, s_{\mathrm{MAP}})$. At a higher degree of approximation, one may factorize the posterior as in Equation 6.18 and first maximize the posterior $p_{\boldsymbol{S}|\boldsymbol{X}^{n+l}=\boldsymbol{x},\boldsymbol{Z}^{n+l}=\boldsymbol{z}}(s)$ with respect to $s$. The obtained point estimate is then used to compute the resampling weights. Finally, the posterior over $h$ is maximized given the fixed resampling weights.

This procedure results in two optimization problems: When using the same likelihood and prior functions as in Sections 6.2, in the first stage one solves the logistic regression problem

$$\mathbf{v}_{\mathrm{MAP}} := \operatorname*{argmin}_{\mathbf{v} \in \mathcal{V}} \sum_{j=1}^{n+l} \log\left(1 + \exp(-z_j \mathbf{v}^\mathsf{T} \boldsymbol{\phi}^s(x_j))\right) + \frac{\rho_{\mathbf{v}}}{2} \mathbf{v}^\mathsf{T} \mathbf{v} \tag{6.26}$$

and computes the resampling weights $\kappa_{\mathbf{v}_{\mathrm{MAP}}}(x_i)$ for $i = 1, \ldots, n$ accordingly. Only then, in the next stage, one solves a second (weighted) logistic regression problem given in Equation 6.27. The criterion of the second stage weights the loss terms that each example

incurs so that the sample is matched to the test distribution.

$$\mathbf{w}_{\mathrm{MAP}} := \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \sum_{i=1}^{n} \kappa_{\mathbf{v}_{\mathrm{MAP}}}(x_i) \log\left(1+\exp(-y_i \mathbf{w}^\mathsf{T}\boldsymbol{\phi}^h(x_i))\right) + \frac{\rho_{\mathbf{w}}}{2}\mathbf{w}^\mathsf{T}\mathbf{w} \tag{6.27}$$

This optimization problem can easily be adapted to virtually any type of classification mechanism. Operationally, an arbitrary classification procedure can be applied to a sample that is resampled from the training data according to the empirical sampling distribution induced by $\kappa_{\mathbf{v}_{\mathrm{MAP}}}(x_i)$.

## 6.4 Applying Kernels

By applying the representer theorem [63], we can directly state a kernelized variant of Algorithm 4 as well as of its two-stage version. We define $\mathbf{w} := \sum_{i=1}^{n} \omega_i \boldsymbol{\phi}^h(x_i)$ and $\mathbf{v} := \sum_{j=1}^{n+l} \upsilon_j \boldsymbol{\phi}^s(x_j)$ where $\boldsymbol{\omega} := [\omega_1, \ldots, \omega_n]^\mathsf{T}$ and $\boldsymbol{\upsilon} := [\upsilon_1, \ldots, \upsilon_{n+l}]^\mathsf{T}$ denote the corresponding dual weight vectors. Line 3 and 4 of Algorithm 4 can now be expressed in terms of these dual vectors, that is,

$$\nu_i \;=\; \exp\left(-y_i \sum_{u=1}^{n} \omega_u k^h(x_u, x_i)\right) \qquad \text{and} \qquad \mu_j \;=\; \exp\left(-z_j \sum_{u=1}^{n+l} \upsilon_u k^s(x_u, x_j)\right)$$

where $k^h(x_u, x_i) := \boldsymbol{\phi}^h(x_u)^\mathsf{T}\boldsymbol{\phi}^h(x_i)$ and $k^s(x_u, x_j) := \boldsymbol{\phi}^s(x_u)^\mathsf{T}\boldsymbol{\phi}^s(x_j)$ are the kernel functions induced by mappings $\boldsymbol{\phi}^h$ and $\boldsymbol{\phi}^s$, respectively. In addition, we can substitute $\kappa_{\mathbf{v}}(x_j)$ in Lines 5 by $\kappa_{\boldsymbol{\upsilon}}(x_j) = \frac{\mu_j}{\sum_{i=1}^{n}\mu_i}$. Line 6 and 7 are replaced by the dual descent vectors $\mathbf{d}_{\boldsymbol{\omega}} := [d_1^{\boldsymbol{\omega}}, \ldots, d_n^{\boldsymbol{\omega}}]^\mathsf{T}$ and $\mathbf{d}_{\boldsymbol{\upsilon}} := [d_1^{\boldsymbol{\upsilon}}, \ldots, d_{n+l}^{\boldsymbol{\upsilon}}]^\mathsf{T}$ with

$$\begin{aligned}
d_i^{\boldsymbol{\omega}} &= \kappa_{\boldsymbol{\upsilon}}(x_i)\frac{\nu_i}{1+\nu_i} y_i - \rho_{\mathbf{w}}\omega_i && \forall\, i = 1, \ldots, n \\
d_i^{\boldsymbol{\upsilon}} &= \kappa_{\boldsymbol{\upsilon}}(x_i)\left(\log(1+\nu_i) - e\right) - \frac{\mu_i}{1+\mu_i} - \rho_{\mathbf{v}}\upsilon_i && \forall\, i = 1, \ldots, n \\
d_j^{\boldsymbol{\upsilon}} &= \frac{\mu_j}{1+\mu_j} - \rho_{\mathbf{v}}\upsilon_j && \forall\, j = n+1, \ldots, n+l
\end{aligned}$$

where $e$ is defined in Line 5 of the algorithm. Finally, the line search in Line 8 requires the computation of the objective,

$$\begin{aligned}
\mathrm{Q}(\boldsymbol{\omega}, \boldsymbol{\upsilon}) \;=\; & \sum_{i=1}^{n} \kappa_{\boldsymbol{\upsilon}}(x_i)\mu_i \log(1+\nu_i) + \sum_{j=1}^{n+l} \log(1+\mu_i) + \\
& \frac{\rho_{\mathbf{w}}}{2}\sum_{i,j=1}^{n} \omega_i\omega_j k^h(x_i, x_j) + \frac{\rho_{\mathbf{v}}}{2}\sum_{i,j=1}^{n+l} \upsilon_i\upsilon_j k^s(x_i, x_j).
\end{aligned}$$

As the LORIE algorithm can be fully expressed in terms of dual weight vectors, we can directly apply kernel functions $k^h$ and $k^s$ without the need for explicit feature mappings $\boldsymbol{\phi}^h$ and $\boldsymbol{\phi}^s$ (*cf.* Section 2.6).

## 6.5 Related Work

Starting with Lemma 6.2, a straightforward approach to compensating for covariate shift is to compute the resampling weights $\kappa(x) := \frac{p_{\boldsymbol{X}|\boldsymbol{\Gamma}=\dot{\gamma}}(x)}{p_{\boldsymbol{X}|\boldsymbol{\Gamma}=\gamma}(x)}$ by first estimating the densities $p_{\boldsymbol{X}|\boldsymbol{\Gamma}=\dot{\gamma}}$ and $p_{\boldsymbol{X}|\boldsymbol{\Gamma}=\gamma}$ from the test and training data, respectively, using *kernel density estimation* (KDE) [66,69]. In a second step, the estimated density ratio is used to resample the training instances, or to train with weighted examples. However, this approach requires to estimate potentially high dimensional densities which is a non-trivial task and only loosely related to the ultimate goal of accurate classification.

*Kernel mean matching* (KMM) [44] is a method that first finds weights for the training instances so that the first momentum of training and test sets—*i.e.,* their mean value—matches in feature space. This is equal to minimize the *maximum mean discrepancy* (MMD) [38] between the resampled training sample and the test sample with respect to the resampling weights, which are used in the subsequent training step. Gretton et al. [39] show that matching the means in feature space is equivalent to matching all moments of the distributions if a characteristic kernel is used. This criterion is satisfied for many common kernels, such as the RBF kernel. Huang et al. [44] derive a quadratic program to find the KMM weights that can be solved with standard optimization tools.

The *Kullback-Leibler importance estimation procedure* (KLIEP) [70] is a third two-step method which estimates the resampling weights for the training examples by minimizing the Kullback-Leibler divergence between the test distribution and the resampled training distribution. Tsuboi et al. [72] derive an extension to KLIEP for large-scale applications and reveal a close relationship to kernel mean matching. An extensive study of KLIEP, KMM as well as the presented logistic regression-based methods can be found in [6].

The covariate shift problem is to some extend similar to the problem of *sample selection bias*. A line of work on learning under sample selection bias has meandered from the statistics and econometrics community into machine learning [41,76]. Sample selection bias relies on the following model of the data generation process: The test data are drawn under the test distribution $P_{\boldsymbol{XY}|\boldsymbol{\Gamma}=\dot{\gamma}}$. The training data are drawn by first sampling $(x, y)$ from the test distribution. Then, a selector variable decides whether $(x, y)$ is moved into the training set or moved into the rejected set, where the labels of the instances in the rejected set are unknown. A typical scenario for sample selection bias is credit scoring. The labeled training sample consists of customers who where given a loan in the past and the rejected sample are customers that asked for but where not given a loan. New customers asking for a loan reflect the test distribution.

In contrast to covariate shift, learning under sample selection bias does not assume the existence of an unlabeled sample of instances drawn from the test distribution, but a labeled sample of selected instances and a set of unlabeled instances which where rejected. In the *missing at random* case of sample selection bias, the selector variable is only dependent on $x$, but not on $y$. In this case, covariate shift models can be applied to learning under sample selection bias by treating the selected examples as the labeled training sample and the union of selected and rejected examples as the unlabeled test sample.

Maximum entropy density estimation under sample selection bias has been studied by Dudik et al. [27]. Bickel and Scheffer [10] impose a Dirichlet process prior on several learning problems with related sample selection bias. Elkan [29] as well as Japkowicz and Stephen [45] investigate the case of training data that are only biased with respect to the class ratio, which can be seen as sample selection bias where the selection only depends on $y$. Cortes et al. [24] theoretically analyze the error that gets introduced by estimating sample selection bias from data. Their analysis also covers the kernel mean matching procedure and a cluster-based estimation technique.

Propensity scores [55,60] are applied in settings related to sample selection bias. Here, the training data are again assumed to be drawn from the test distribution $P_{\boldsymbol{XY}|\boldsymbol{I}=\dot{\gamma}}$ followed by a selection process. The difference to the setting of sample selection bias is that the selected *and* the rejected examples are labeled. Those samples can again be corrected by resampling, which results in two unbiased samples with respect to the test distribution.

Propensity scoring can precede a variety of analysis steps. This can be the training of a target model on reweighted data or just a statistical analysis of the two reweighted samples. A typical application for propensity scores is the analysis of the success of a medical treatment. Patients are selected to be given the treatment and some other patients are selected into the control group. If the selector variable is not independent of $x$ (patients may be chosen for an experimental therapy only if they meet specific requirements), the outcome (*e.g.,* ratio of cured patients) of the two groups cannot be compared directly and propensity scores have to be applied.

## 6.6 Empirical Evaluation

In this section we empirically analyze several methods to compensate for covariate shift. We compare the proposed integrated method *logistic regression importance estimation* (LORIE) against two-step methods which estimate the resampling weights in a first step and subsequently train a weighted logistic regression (*cf.* Equation 6.27) in the second step. Hence, for reasons of comparison, all investigated methods use a logistic model to estimate the parameters of the main predictive model $h$. We consider the following existing baselines to estimate the resampling weights: The trivial choice of *i.i.d.* weights which all equal one (iid), the *kernel mean matching* weights (KMM), and the *Kullback-Leibler importance estimation procedure* weights (KLIEP). In addition, we use *logistic regression* weights (LR) as a two-stage approximation of LORIE (*cf.* Equation 6.26). We use the same data sets and feature representation as in the previous two chapters.

### 6.6.1 A Case Study on Email Spam Filtering

In the first experiment, we consider a similar experiment as in Sections 4.6.4 and 5.6.1. We use the 4,000 oldest emails as training portion and set the remaining emails aside as

test instances. We again use the F-measure as evaluation measure, repeat all experiments 20 times, and perform a paired $t$-test ($\alpha = 0.05$). For each repetition, we first draw a training set of 200 spam and 200 non-spam messages from the training portion. We split the chronologically sorted test data into 20 disjoint, equally-sized sets and draw another 400 emails from each split. For each pair of training and test sets—each containing 400 emails—we train all methods, where the resulting predictive models are then applied to the test instances of the corresponding split. Instead of the whole splits, we use only 400 test instances from each split to keep the computational costs reasonable and to avoid skewed testing-to-training ratios.

We construct the feature representation of the emails as in the experiments of the previous chapters. Because of computational issues, for each experiment and each repetition, we construct the joint kernel PCA mapping (2.30) with respect to the corresponding 400 training and 400 test emails using the linear kernel resulting in 800-dimensional training and test instances. This procedure is equivalent to use the identity mapping for $\boldsymbol{\phi}^h$ and $\boldsymbol{\phi}^s$ in LORIE, and to use a linear logistic regression in the second stage of the two-step methods, respectively.

We use the default parameters of KMM and the internal procedure for parameter tuning of KLIEP. In order to tune the parameters of the two-stage approximation of LORIE, we perform a 5-fold cross validation with respect to the regularization parameter of the first stage within each repetition of an experiment. To tune the regularization parameter of the *i.i.d.* baseline and the logistic regression of the second stage of the two-step methods, we use the resampled training portion to perform a second 5-fold cross validation. This is again carried out in each repetition of an experiment and for each method.

For LORIE, the regularization parameters $\rho_{\mathbf{v}}$ and $\rho_{\mathbf{w}}$ need to be estimated simultaneously which is generally impossible, as we have no access to a sample of labeled test instances. We therefore exploit the similarity of LORIE and its two-stage approximation: We use 5-fold cross validation to jointly tune both regularization parameters. However, in contrast to standard CV, for each of the five foldings we use a resampled test fold, where we use the logistic regression weights of the two-stage approximation of LORIE. We expect that this resampled test fold is a reasonable test sample, which is sufficient to tune the hyperparameters of LORIE.

Figure 6.1 shows that, except for the Mailinglist data set, all reweighting methods do *not* significantly improve the performance in comparison to the unweighted *i.i.d.* baseline. This is in contrast to reported results in other domains where KMM, KLIEP, LORIE etc. where shown to outperform the unweighted logistic regression (see, for instance, [5, 7–9, 44, 72]).

This negative result in the context of email spam filtering may be caused by a violation of Assumption 6.1.4, that is, the training distribution may *not* cover the entire support of the test distribution. The spam emails used for training reflect certain spam campaigns, for instance, to advertise watches. As future spam emails may cover completely different campaign topics such as drugs, the learning algorithm of the second stage will not benefit from a resampling of the older spam emails.

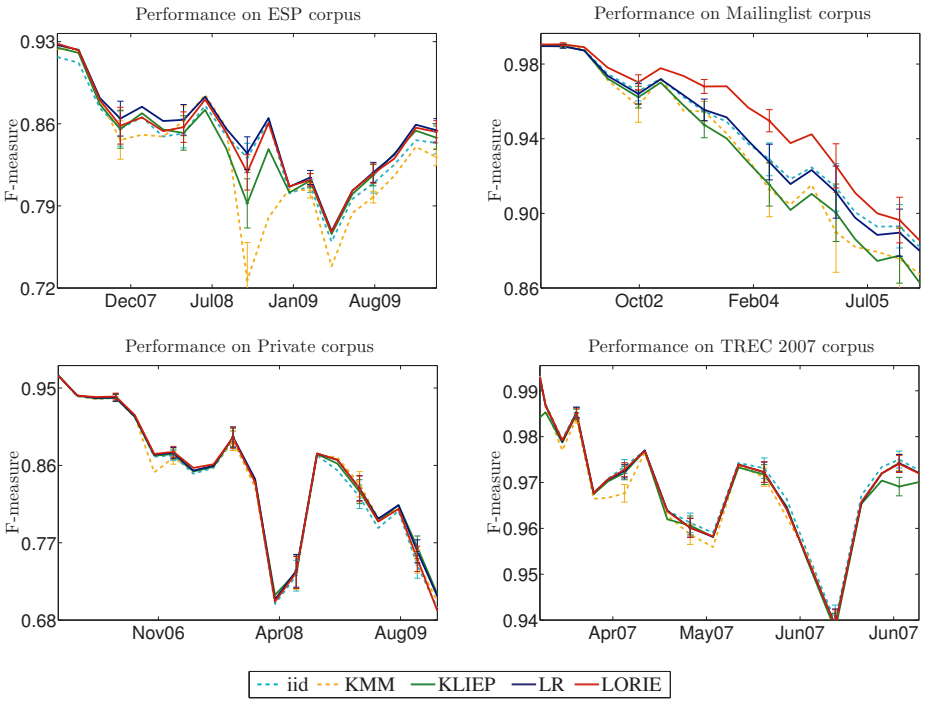


**Figure 6.1:** F-measure of predictive models. Error bars indicate standard errors.

### 6.6.2 Inspection of the Resampling Weights

In the following experiment we aim at investigating the reason for the negative result of the previous experiment. One potential source is the reduction of the effective sample size (see, for instance, [49])

$$\hat{n} := \frac{n^2}{\sum_{i=1}^{n} \kappa_i^2}\,, \tag{6.28}$$

where $\kappa_i$ are the positive resampling weights which sum up to $n$. We compute the effective sample size for each method and each data set. Figure 6.2 reports on the results.

We observe that, except for the Mailinglist corpus, the effective sample size does not change a lot for KMM whereas for the other reweighting approaches it reduces typically by 20% to 50%. In LORIE and its two-stage approximation the effective sample size falls even below 100 for the Private corpus. We may conclude, that the potential improvement induced by a consistent likelihood may vanish because of the reduction of the effective sample size. However, LORIE significantly outperforms all baselines for the Mailinglist data set while still having a reduced effective sample size. Hence, this lowering is not neccesarily the cause of the previous negative result.

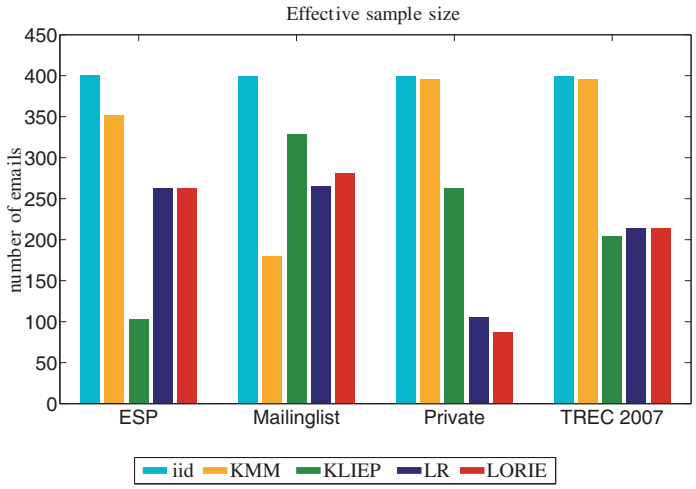


**Figure 6.2:** Effective sample size induced by the weights of the individual methods.

We finally shall study to which extend the (normalized) resampling weights $\frac{\kappa_i}{n}$ can be estimated from the data. If Assumption 6.1.4 is violated, we will expect this estimation problem to be nearly impossible to solve. In contrast, if the assumption is satisfied, the distinct methods should produce to some extend similar weights, that is, the methods should agree about under and over-represented instances.

In order to compute to which extend the weights differ between the particular methods, we compute the Kullback-Leibler divergence in (6.29) between the (normalized) weights for all pairs of methods. We report on the results for all methods and all four data sets in Table 6.1.

$$\mathrm{KL}(\boldsymbol{\kappa}^{(1)}||\boldsymbol{\kappa}^{(2)}) = -\frac{1}{n}\sum_{i=1}^{n}\kappa_i^{(1)}\log\frac{\kappa_i^{(2)}}{\kappa_i^{(1)}}, \tag{6.29}$$

With the exception of KMM, we observe a significant KL-divergence between the resulting weights of the particular methods and the *i.i.d.* baseline where all weights equal one. However, the divergence between the individual reweighting methods is, except between LORIE and LR, of the same order. From this we conclude that the estimated weights heavily differ between most of the methods. For our experimental setting, it seems to be hard to derive meaningful weights, which is an indication for our previously made hypothesis that Assumption 6.1.4 is violated.

Since in the experiment of the previous section the resulting predictive models perform comparably, the difference of the weights seems to have only little impact on the resulting model. From this we conclude, that in our running example of email spam filtering, a resampling of the training data is unlikely to improve the predictive performance of the resulting classifier, independently of the particular resampling weights.

**Table 6.1:** Kullback-Leibler divergence (*cf.* Equation 6.29) between the normalized weights. The method in the row has produced the weights $\kappa_i^{(1)}$ and the method in the column has produced the weights $\kappa_i^{(2)}$ for $i = 1, \ldots, n$ and $n = 400$.

ESP

| | iid | KMM | KLIEP | LR | LORIE |
|---|---|---|---|---|---|
| iid | 0.000 | 0.781 | 2.842 | 0.304 | 0.302 |
| KMM | 0.102 | 0.000 | 2.842 | 0.304 | 0.302 |
| KLIEP | 1.110 | 1.110 | 0.000 | 0.638 | 0.654 |
| LR | 0.243 | 0.243 | 1.656 | 0.000 | 0.000 |
| LORIE | 0.242 | 0.242 | 1.698 | 0.000 | 0.000 |

Mailinglist

| | iid | KMM | KLIEP | LR | LORIE |
|---|---|---|---|---|---|
| iid | 0.000 | 3.318 | 0.226 | 0.221 | 0.221 |
| KMM | 0.460 | 0.000 | 0.303 | 0.173 | 0.174 |
| KLIEP | 0.129 | 3.192 | 0.000 | 0.130 | 0.130 |
| LR | 0.211 | 1.960 | 0.151 | 0.000 | 0.091 |
| LORIE | 0.320 | 2.039 | 0.133 | 0.090 | 0.000 |

Private

| | iid | KMM | KLIEP | LR | LORIE |
|---|---|---|---|---|---|
| iid | 0.000 | 0.348 | 0.702 | 0.616 | 0.642 |
| KMM | 0.010 | 0.000 | 0.724 | 0.633 | 0.660 |
| KLIEP | 0.344 | 0.933 | 0.000 | 0.368 | 0.398 |
| LR | 0.614 | 1.087 | 0.490 | 0.000 | 0.002 |
| LORIE | 0.676 | 1.128 | 0.551 | 0.002 | 0.000 |

TREC 2007

| | iid | KMM | KLIEP | LR | LORIE |
|---|---|---|---|---|---|
| iid | 0.000 | 0.009 | 1.421 | 0.349 | 0.349 |
| KMM | 0.007 | 0.000 | 1.353 | 0.338 | 0.338 |
| KLIEP | 0.476 | 0.475 | 0.000 | 0.139 | 0.139 |
| LR | 0.307 | 0.307 | 0.250 | 0.000 | 0.000 |
| LORIE | 0.307 | 0.306 | 0.250 | 0.000 | 0.000 |

# 7 Conclusions

In this thesis we addressed the problem of building prediction models from data which are robust against active adversaries. This problem frequently occurs, for instance, in security applications where an attacker aims at circumventing detections mechanisms such as spam filters, firewalls, virus scanners, or intrusion and fraud detection systems. To this end, we introduced the concept of *prediction games* which formulates the learning problem as a game between two parties, a *learner* who has to commit to a predictive model using past data and a *data generator* who may change the process of data generation. We focused on *one-shot games* of *complete information* and studied three distinct scenarios: The case where both players act simultaneously, the case where the learner moves first and the data generator reacts, and finally the case where the data generator first changes the data generation process on which the learner has to respond. We studied all settings from the perspective of the learner to derive their game-theoretic optimal action and empirically analyzed the performance of these game solutions. In the following we summarize key results.

**Adversarial prediction problems are two-player one-shot games.** We analyzed the common properties of adversarial prediction problems: First, there are two parties involved—a learner and a data generator—whose interests are in conflict. This setting establishes a two-player game. Second, each player is expected to minimize their costs for each round separately so that the ongoing interactions between learner and data generator generally decouple into a sequence of one-shot games.

**Players have not necessarily antagonistic interests.** In many applications, the interests of both players are highly conflicting but not necessarily fully antagonistic. This means that the data generator's goal is generally *not* to explicitly choose an action which harms the learner most, but to follow their own interests. Existing methods, which rely on the assumption that the costs are perfectly antagonistic, could not correctly model this scenario.

**Players have complete information.** To model the theoretical prediction costs of the players, we focused on regularized empirical estimates based on the given training data. This discriminative approach leads to games of complete information.

**Simultaneously acting players establish Nash prediction game.** In Chapter 4, we focused on static games in which learner and data generator have to commit *simultaneously* to a prediction model and a transformation on the data generation process, respectively. In the absence of information about the opponent's move, both players may choose to play a Nash equilibrium which constitutes a cost-minimizing move for each player *if* the other player follows the equilibrium as well.

**Existence of unique Nash equilibrium is easily verifiable.** Playing a Nash-strategy is only advisable if a unique Nash equilibrium exist. In Assumption 4.1 we summarized sufficient conditions under which it is rational for both parties to play a Nash equilibrium. We have discussed these requirements in Section 4.1. In particular, we have studied conditions under which a prediction game can be guaranteed to possess a unique Nash equilibrium. Lemma 4.1 identifies conditions (*cf.* Assumption 4.2) under which at least one equilibrium exists and Theorem 4.8 elaborates on when this equilibrium is unique (*cf.* Assumption 4.3).

**Nash-optimal prediction models can be computed efficiently.** We proposed an inexact linesearch approach and a modified extragradient approach to identifying this unique equilibrium. The first method is based on the Nikaido-Isoda function that is defined so that a minimax solution of this function is an equilibrium of the Nash prediction game and vice versa. We solved this minimax problem by a descent method with inexact linesearch. In the second approach, we reformulated the Nash prediction game into a variational inequality problem which was solved by a modified extragradient method. Empirically, both approaches turned out to perform quite similarly. We derived Nash logistic regression and Nash support vector machine models, and derived kernelized versions of these methods.

**Nash-optimal prediction models are empirically effective.** Empirically, we found that both methods identify unique Nash equilibria when the bounds laid out in Corollaries 4.9 and 4.10 are satisfied or violated by a factor of up to 4. From our experiment on several email corpora we concluded that Nash logistic regression and the Nash support vector machine significantly outperform their *i.i.d.* baselines and the Invar-SVM for the problem of classifying future emails based on training data from the past.

**Responsive data generator establishes Stackelberg prediction game.** In Chapter 5, we modeled the adversarial prediction problem as a dynamic two-stage game, that is, a Stackelberg competition. This game assumes a learner who first commits to a predictive model, whereas the data generator may choose a transformation of the data generation process *after* the predictive model has been disclosed. That model reflects applications such as the detection of network attacks and spam filtering in which an assailant can probe the filter.

**Solution to Stackelberg prediction game always exists.** Playing the Stackelberg equilibrium instead of a worst-case strategy based on a zero-sum model is advisable when the data generator can be assumed to behave rational in the sense of minimizing their own cost function. We stated the conditions for the optimality of a Stackelberg equilibrium in Assumption 5.1. In contrast to the Nash strategy, the Stackelberg model does not rely on the existence of a unique equilibrium and the assumptions that the adversary has no information about the predictive model and is able to identify and follow the equilibrial strategy.

**Stackelberg equilibrium can be efficiently estimated.** We derived a compact optimization problem that determines the solution of the Stackelberg prediction game. An algorithm to solve this optimization problem in the primal as well as a kernelized version are provided. We showed that the Stackelberg model generalizes existing prediction models such as SVM with uneven margins and SVM for invariances.

**Stackelberg-optimal prediction models are empirically effective.** As for the Nash prediction game, we evaluated spam filters resulting from a regular SVM, logistic regression, existing game-theoretical models, and three instances of the Stackelberg prediction game on several spam-filtering data sets. The relative performance of the distinct game-theoretic models varies, but we observed that, when compared to any other model, the Stackelberg model with logistic loss has more wins than it has losses against each of the baseline methods.

**Responsive learner establishes covariate shift learning problem.** In Chapter 6 we studied the case where the data generator acts *before* the learner. In this setting, the data generator moves first by producing a sample of test data (*cf.* Assumption 6.1). These instances, but not the corresponding target attributes, are then observed by the learner who builds a predictive model based on the training data and this unlabeled test sample. As the training and test sample originate from potentially distinct data generation processes, this problem amounts to learning under *covariate shift*.

**MAP estimate based on an unbiased likelihood.** We showed that the classical approach to learning, which only takes the labeled training data into account, implicitly relies on a biased label likelihood, which may lead to an inaccurate prediction model. We introduced the *theoretical label likelihood* which is the theoretical quantity the (instance) label likelihood converges to. Based on this concept, we derived an unbiased label likelihood by resampling the training instances. This unbiased label likelihood gives rise to a discriminative model which accounts for the covariate shift. We showed that the contribution of each training instance to the induced optimization problem ideally needs to be weighted with its testing-to-training density ratio. We also showed that this ratio can be expressed—without modeling either training or test density—by a discriminative model that characterizes how much more likely an instance is to occur in the test sample than it is to occur in the training sample.

**Reweighting methods to derive covariate shift-compensating MAP estimate.** We derived a primal and a kernelized gradient descent procedure for the joint optimization problem called *logistic regression importance estimation*. We showed that this algorithm converges to a locally but, unfortunately, not necessarily globally optimal solution, as the objective is generally not jointly convex in all parameters. We therefore proposed a two-stage method which approximates the integrated method. The two-stage model is conceptually simpler than the integrated model and may, in some cases, have the greatest practical utility. Beside the convexity in each stage, the

main advantage compared to the integrated model is that regularization parameters can be tuned without prior knowledge by cross-validation. Another advantage of the two-stage model is that in the second stage, after the example-specific weights have been derived, virtually any learning mechanism can be employed to produce the final classifier from a resampled training sample.

**Reweighting methods improve spam filters only marginally.** We observed in our experiments, that in the case of email spam filtering, the proposed as well as existing reweighting methods such as kernel mean matching and the Kullback-Leibler importance estimation procedure do *not* outperform the *i.i.d.* baseline for most of the data sets. As this finding is in contrast to previously published results, we investigated possible causes. We identified two potential reasons: First, the reweighting of the instances reduces the effective sample size dramatically which increases the variance of the estimation error and may lead to less accurate prediction models. And second, the made assumption that the training distribution covers the entire support of the test distribution may be violated such that it becomes impossible to identify meaningful resampling weights.

In the investigated field of prediction games, many opportunities for future work are available. First of all, in this thesis we focused on the task of spam filtering where we had access to real-world data. It may be interesting to study how effective the proposed methods are in other domains apart from email spam such as network intrusion or credit card fraud. In these areas, domain experts may provide detailed information about the likelihood of particular changes of the data generation process. This may give rise to an application-specific measure of the divergence between the original and the perturbated training sample.

We mainly studied binary classification. For Stackelberg prediction games and learning under covariate shift, there are only weak restrictions on the loss functions. These models can be directly applied to other learning task such as multi-class classification and regression. However, for the Nash prediction game it may be non-trivial to verify whether the requirement of Lemma 4.2, that is, the existence of a unique Nash equilibrium, is met in other learning settings.

We modeled the adversarial prediction problem as a two-player game of complete information. In some applications, there may be indeed more than two parties competing with each other. It is unclear to which extend the presented prediction models can be extended to three or more players and how this affects the computational costs as well as predictive performance. In addition, we decided for games of complete information so that we did not have to explicitly model the test distribution $P_{\boldsymbol{XY}|\boldsymbol{T}=\hat{\gamma}}$. In some domains it may be desirable to know the optimal move (the optimal test distribution) of the data generator, too. This would require to make an assumption on the functional form of the test distribution leading to Bayesian games. It therefore may be worthwhile to extend prediction games to Bayesian prediction games.

Finally, most of the proposed methods are still computationally expensive. Especially for Nash and Stackelberg prediction games, finding the perturbation which is optimal for the data generator requires to solve an optimization problems with a very high number of variables—precisely, the product of the number of instances and attributes. In practice, for instance, spam filters are derived from millions of emails with hundreds of thousands attributes, *i.e.,* words and word combinations. In such a setting, not all of the proposed methods are directly applicable. Techniques to significantly speed up the learning processes and allowing for large-scale experiments are therefore of great interest.

# Bibliography


[1] Altman, E., Boulogne, T., Azouzi, R.E., Jiménez, T., Wynter, L.: A survey on networking games in telecommunications. Computers & OR 33, 286–311 (2006)

[2] Androutsopoulos, I., Magirou, E.F., Vassilakis, D.K.: A game theoretic model of spam e-mailing. In: Proceedings of the Conference on Email and Anti-Spam (2005)

[3] Basar, T., Olsder, G.J.: Dynamic Noncooperative Game Theory. Society for Industrial and Applied Mathematics (1999)

[4] Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear Programming: Theory and Algorithms. Wiley and Sons, 3th edn. (2006)

[5] Bickel, S., Sawade, C., Scheffer, T.: Transfer learning by distribution matching for targeted advertising. In: Advances in Neural Information Processing Systems (NIPS). MIT Press (2009)

[6] Bickel, S.: Learning under Differing Training and Test Distributions. Ph.D. thesis, Mathematisch-Naturwissenschaftlichen Fakultät der Universität Potsdam (2009)

[7] Bickel, S., Brückner, M., Scheffer, T.: Discriminative learning for differing training and test distributions. In: Proceedings of International Conference on Machine Learning (ICML). ACM Press, Oregon, USA (2007)

[8] Bickel, S., Brückner, M., Scheffer, T.: Dataset Shift in Machine Learning, chap. Discriminative learning under covariate shift with a single optimization problem, pp. 161–178. MIT Press (2009)

[9] Bickel, S., Brückner, M., Scheffer, T.: Discriminative learning under covariate shift. Journal of Machine Learning Research (JMLR) 10, 2137–2155 (2009)

[10] Bickel, S., Scheffer, T.: Dirichlet-enhanced spam filtering based on biased samples. In: Advances in Neural Information Processing Systems (NIPS). MIT Press, Cambridge, USA (2007)

[11] Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)

[12] Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning 3(1), 1–122 (2011)

[13] Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2004)

[14] Brückner, M., Bickel, S., Scheffer, T.: Optimal spamming: Solving a family of adversarial classification games. In: Proceedings of NIPS Workshop on Machine Learning in Adversarial Environments for Computer Security (2007)

[15] Brückner, M., Haider, P., Scheffer, T.: Highly scalable discriminative spam filtering. In: Proceedings of 15th Text REtrieval Conference (TREC). National Institute of Standards and Technology (NIST) (2006)

[16] Brückner, M., Kanzow, C., Scheffer, T.: Static prediction games for adversarial learning problems. Journal of Machine Learning Research (JMLR) 13, 2589–2626 (2012)

[17] Brückner, M., Scheffer, T.: Nash equilibria of static prediction games. In: Advances in Neural Information Processing Systems (NIPS). MIT Press (2009)

[18] Brückner, M., Scheffer, T.: Stackelberg games for adversarial prediction problems. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), San Diego, CA, USA. ACM Press (2011)

[19] Chapelle, O.: Training a support vector machine in the primal. Neural Computation 19(5), 1155–1178 (2007)

[20] Charilas, D.E., Panagopoulos, A.D.: A survey on game theory applications in wireless networks. Computer Networks 54(18), 3421–3430 (2010)

[21] Colson, B., Marcotte, P., Savard, G.: An overview of bilevel optimization. Annals of Operations Research 153(1), 235–256 (2007)

[22] Contreras, J., Klusch, M., Krawczyk, J.B.: Numerical solutions to Nash-Cournot equilibria in coupled constraint electricity markets. IEEE Transaction on Power Systems 19(1), 195–206 (2004)

[23] Cormack, G.V.: TREC 2007 spam track overview. In: Voorhees, E.M., Buckland, L.P. (eds.) TREC. vol. Special Publication 500-274. National Institute of Standards and Technology (NIST) (2007)

[24] Cortes, C., Mohri, M., Riley, M., Rostamizadeh, A.: Sample selection bias correction theory. In: Proceedings of the International Conference on Algorithmic Learning Theory (2008)

[25] Dekel, O., Shamir, O.: Learning to classify with missing and corrupted features. In: Proceedings of the International Conference on Machine Learning (ICML). pp. 216–223. ACM Press (2008)

[26] Dekel, O., Shamir, O., Xiao, L.: Learning to classify with missing and corrupted features. Machine Learning 81(2), 149–178 (2010)

[27] Dudik, M., Schapire, R., Phillips, S.: Correcting sample selection bias in maximum entropy density estimation. In: Advances in Neural Information Processing Systems (NIPS). MIT Press (2005)

[28] Duflo, M.: Random Iterative Models. Springer-Verlag New York, Secaucus, NJ, USA (1997)

[29] Elkan, C.: The foundations of cost-sensitive learning. In: Proceedings of the International Joint Conference on Artificial Intellligence (2001)

[30] Facchinei, F., Fischer, A., Piccialli, V.: On generalized Nash games and variational inequalities. Operations Research Letters 35(2), 159–164 (2007)

[31] Facchinei, F., Fischer, A., Piccialli, V.: Generalized Nash equilibrium problems and Newton methods. Mathematical Programming 117(1-2), 163–194 (2009)

[32] Facchinei, F., Kanzow, C.: Generalized Nash equilibrium problems. 4OR 5(3), 173–210 (2007)

[33] Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: A statistical view of boosting. The Annals of Statistics 28(2), 337–374 (2000)

[34] Geiger, C., Kanzow, C.: Theorie und Numerik restringierter Optimierungsaufgaben. Springer (1999)

[35] Ghaoui, L.E., Lanckriet, G.R.G., Natsoulis, G.: Robust classification with interval data. Tech. Rep. UCB/CSD-03-1279, EECS Department, University of California, Berkeley (2003)

[36] Globerson, A., Roweis, S.T.: Nightmare at test time: Robust learning by feature deletion. In: Proceedings of the International Conference on Machine Learning (ICML). ACM Press (2006)

[37] Globerson, A., Teo, C.H., Smola, A.J., Roweis, S.T.: Dataset Shift in Machine Learning, chap. An adversarial view of covariate shift and a minimax approach, pp. 179–198. MIT Press (2009)

[38] Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A.J.: A kernel method for the two-sample problem. CoRR abs/0805.2368 (2008)

[39] Gretton, A., Fukumizu, K., Harchaoui, Z., Sriperumbudur, B.: A fast, consistent kernel two-sample test. In: Advances in Neural Information Processing Systems (NIPS). MIT Press (2009)

[40] Harker, P.T., Pang, J.S.: Finite-dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms and applications. Mathematical Programming 48(2), 161–220 (1990)

[41] Heckman, J.: Sample selection bias as a specification error. Econometrica 47, 153–161 (1979)

[42] von Heusinger, A., Kanzow, C.: Optimization reformulations of the generalized Nash equilibrium problem using Nikaido-Isoda-type functions. Computational Optimization and Applications 43(3), 353–377 (2007)

[43] von Heusinger, A., Kanzow, C.: Relaxation methods for generalized Nash equilibrium problems with inexact line search. Journal of Optimization Theory and Applications 143(1), 159–183 (2009)

[44] Huang, J., Smola, A.J., Gretton, A., Borgwardt, K., Schölkopf, B.: Correcting sample selection bias by unlabeled data. In: Advances in Neural Information Processing Systems (NIPS). MIT Press (2007)

[45] Japkowicz, N., Stephen, S.: The class imbalance problem: A systematic study. Intelligent Data Analysis 6, 429–449 (2002)

[46] Jeroslow, R.: The polynomial hierarchy and a simple model for competitive analysis. Mathematical Programming 32, 146–164 (1985)

[47] Kantarcioglu, M., Xi, B., Clifton, C.: A game theoretical framework for adversarial learning (2008)

[48] Kantarcioglu, M., Xi, B., Clifton, C.: Classifier evaluation and attribute selection against active adversaries. Data Mining and Knowledge Discovery 22(1-2), 291–335 (2011)

[49] Kong, A.: A note on importance sampling using standardized weights. Tech. Rep. 348, University of Chicago, Department of Statistics (1992)

[50] Li, Y., Shawe-Taylor, J.: The SVM with uneven margins and chinese document categorization. In: Proceedings of the Pacific Asia Conference on Language, Information and Computation. pp. 216–227 (2003)

[51] Lippert, R.A., Rifkin, R.M.: Infinite-sigma limits for Tikhonov regularization. Journal of Machine Learning Research (JMLR) 7, 855–876 (2006)

[52] Liu, W., Chawla, S.: A game theoretical model for adversarial learning. In: ICDM Workshops. pp. 25–30. IEEE Computer Society (2009)

[53] Liu, W., Chawla, S.: Mining adversarial patterns via regularized loss minimization. Machine Learning 81(1), 69–83 (2010)

[54] Long, N.V.: A Survey of Dynamic Games in Economics, vol. 1. World Scientific Publishing Co. Pte. Ltd. (2010)

[55] Lunceford, J., Davidian, M.: Stratification and weighting via the propensity score in estimation of causal treatment effects: A comparative study. Statistics in Medicine 23(19), 2937–2960 (2004)

[56] Parameswaran, M., Rui, H., Sayin, S.: A game theoretic model and empirical analysis of spammer strategies. In: Proceedings of the Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (2010)

[57] Rasmusen, E.: Games and Information: An Introduction to Game Theory. Wiley-Blackwell, 4 edn. (2006)

[58] Robbins, H., Siegmund, D.: A convergence theorem for non-negative almost supermartingales and some applications. In: Optimizing methods in Statistics. pp. 233–257. Academic Press, New York (1971)

[59] Rosen, J.B.: Existence and uniqueness of equilibrium points for concave n-person games. Econometrica 33(3), 520–534 (1965)

[60] Rosenbaum, P., Rubin, D.: The central role of the propensity score in observational studies for causal effects. Biometrika 70(1), 41–55 (1983)

[61] Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psych. Rev. 65, 386–407 (1958)

[62] Roy, S., Ellis, C., Shiva, S., Dasgupta, D., Shandilya, V., Wu, Q.: A survey of game theory as applied to network security. In: HICSS. pp. 1–10. IEEE Computer Society (2010)

[63] Schölkopf, B., Herbrich, R., Smola, A.J.: A generalized representer theorem. In: COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers (2001)

[64] Schölkopf, B., Smola, A.J.: Learning with Kernels. MIT Press, Cambridge, MA (2002)

[65] Shalev-Shwartz, S., Singer, Y., Srebro, N.: Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In: Proceedings of the 24th International Conference on Machine Learning (ICML). ACM Press (2007)

[66] Shimodaira, H.: Improving predictive inference under covariate shift by weighting the log-likelihood function. Journal of Statistical Planning and Inference 90, 227–244 (2000)

[67] Siefkes, C., Assis, F., Chhabra, S., Yerazunis, W.S.: Combining Winnow and orthogonal sparse bigrams for incremental spam filtering. In: Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD). Lecture Notes in Artificial Intelligence, vol. 3202, pp. 410–421. Springer (2004)

[68] Spall, J.C.: Introduction to Stochastic Search and Optimization. John Wiley & Sons, Inc., New York, NY, USA (2003)

[69] Sugiyama, M., Müller, K.R.: Input-dependent estimation of generalization error under covariate shift. Statistics and Decision 23(4), 249–279 (2005)

[70] Sugiyama, M., Nakajima, S., Kashima, H., von Bünau, P., Kawanabe, M.: Direct importance estimation with model selection and its application to covariate shift adaptation. In: Advances in Neural Information Processing Systems (NIPS) (2008)

[71] Teo, C.H., Globerson, A., Roweis, S.T., Smola, A.J.: Convex learning with invariances. In: Advances in Neural Information Processing Systems. MIT Press (2007)

[72] Tsuboi, J., Kashima, H., Hido, S., Bickel, S., Sugiyama, M.: Direct density ratio estimation for large-scale covariate shift adaptation. In: Proceedings of the SIAM International Conference on Data Mining (2008)

[73] van der Vaart, A.W.: Asymptotic Statistics. Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press (2000)

[74] Veelken, S.: A New Relaxation Scheme for Mathematical Programs with Equilibrium Constraints: Theory and Numerical Experience. Ph.D. thesis, Technische Universität München (2009)

[75] Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming 106, 25–57 (2006)

[76] Zadrozny, B.: Learning and evaluating classifiers under sample selection bias. In: Proceedings of the International Conference on Machine Learning (ICML) (2004)

[77] Zinkevich, M., Weimer, M., Smola, A., Li, L.: Parallelized stochastic gradient descent. In: Advances in Neural Information Processing Systems (NIPS) (2010)

# Appendix

## Proofs

*Proof of Proposition 2.2.* The Huber loss $\ell^{\mathrm{H}}(g_{\mathbf{w}}(x), y)$ is a function in $z := yg_{\mathbf{w}}(x)$ with fixed $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Hence, it is sufficient to show that this loss function is convex and continuously differentiable in $z \in \mathbb{R}$. Let $\ell'$ and $\ell''$ denote the first and second derivative, respectively, of loss function $\ell^{\mathrm{H}}$ with respect to $z$. Then, a direct calculation shows that

$$\ell'(g_{\mathbf{w}}(x), y) = \begin{cases} 0 & \text{if} \quad z > 1+\delta \\ \frac{z-(1+\delta)}{2\delta} & \text{if} \quad |1-z| \leq \delta \\ -1 & \text{if} \quad z < 1-\delta \end{cases} \tag{A.1}$$

and

$$\ell''(g_{\mathbf{w}}(x), y) = \begin{cases} 0 & \text{if} \quad z > 1+\delta \\ \frac{1}{2\delta} & \text{if} \quad |1-z| \leq \delta \\ 0 & \text{if} \quad z < 1-\delta \end{cases} . \tag{A.2}$$

The Huber loss is continuously differentiable in $z \in \mathbb{R}$ as it is continuous in $z$,

$$\begin{aligned} \lim_{z \to (1+\delta)^+} \ell^{\mathrm{H}}(g_{\mathbf{w}}(x), y) &= 0 = \tfrac{(1+\delta-(1+\delta))^2}{4\delta} = \lim_{z \to (1+\delta)^-} \ell^{\mathrm{H}}(g_{\mathbf{w}}(x), y) \\ \lim_{z \to (1-\delta)^+} \ell_{\mathrm{H}}(g_{\mathbf{w}}(x), y) &= \tfrac{(1+\delta-(1-\delta))^2}{4\delta} = 1-(1-\delta) = \lim_{z \to (1-\delta)^-} \ell^{\mathrm{H}}(g_{\mathbf{w}}(x), y) \end{aligned}$$

and its first derivative is continuous in $z$ as well,

$$\begin{aligned} \lim_{z \to (1+\delta)^+} \ell'(g_{\mathbf{w}}(x), y) &= 0 = \tfrac{(1+\delta)-(1+\delta)}{2\delta} = \lim_{z \to (1+\delta)^-} \ell'(g_{\mathbf{w}}(x), y) \\ \lim_{z \to (1-\delta)^+} \ell'(g_{\mathbf{w}}(x), y) &= \tfrac{(1-\delta)-(1+\delta)}{2\delta} = -1 = \lim_{z \to (1-\delta)^-} \ell'(g_{\mathbf{w}}(x), y) \end{aligned} .$$

As the second derivative $\ell''$ is a piecewise-constant function, the Huber loss is not twice-continuously differentiable. However, it is convex in $z \in \mathbb{R}$ since $\ell''$ is non-negative for all $z \in \mathbb{R}$. □

*Proof of Proposition 2.3.* As in the previous proof, we consider loss function $\ell^{\mathrm{t}}(g_{\mathbf{w}}(x), y)$ as a function in $z := yg_{\mathbf{w}}(x)$ with fixed $(x, y) \in \mathcal{X} \times \mathcal{Y}$, and show that $\ell^{\mathrm{t}}$ is convex and twice-continuously differentiable in $z \in \mathbb{R}$. Again, let $\ell'$ and $\ell''$ denote the first and second derivative, respectively, of loss function $\ell^{\mathrm{t}}$ with respect to $z$ given in (A.3) and (A.4).

$$\ell'(g_{\mathbf{w}}(x), y) = \begin{cases} 0 & \text{if} \quad z > \delta \\ -\frac{1}{2} + \frac{1}{2}\sin\left(\frac{\pi}{2\delta}z\right) & \text{if} \quad |z| \leq \delta \\ -1 & \text{if} \quad z < -\delta \end{cases} \tag{A.3}$$

$$\ell''(g_{\mathbf{w}}(x), y) = \begin{cases} 0 & \text{if} \quad z > \delta \\ \frac{\pi}{4\delta}\cos\left(\frac{\pi}{2\delta}z\right) & \text{if} \quad |z| \leq \delta \\ 0 & \text{if} \quad z < -\delta \end{cases} \tag{A.4}$$

The trigonometric loss is twice-continuously differentiable in $z \in \mathbb{R}$ as it is continuous in $z$,

$$\begin{array}{rcccl} \lim\limits_{z\to\delta^+} \ell^{\mathrm{t}}(g_{\mathbf{w}}(x), y) & = & 0 & = & \frac{\delta-\delta}{2} - \frac{\delta}{\pi}\cos\left(\frac{\pi}{2\delta}\delta\right) = \lim\limits_{z\to\delta^-} \ell^{\mathrm{t}}(g_{\mathbf{w}}(x), y) \\ \lim\limits_{z\to-\delta^+} \ell^{\mathrm{t}}(g_{\mathbf{w}}(x), y) & = & \frac{\delta+\delta}{2} - \frac{\delta}{\pi}\cos\left(-\frac{\pi}{2\delta}\delta\right) = \delta & = & \lim\limits_{z\to-\delta^-} \ell^{\mathrm{t}}(g_{\mathbf{w}}(x), y) \end{array},$$

its first derivative is continuous in $z$,

$$\begin{array}{rcccl} \lim\limits_{z\to\delta^+} \ell'(g_{\mathbf{w}}(x), y) & = & 0 & = & -\frac{1}{2} + \frac{1}{2}\sin\left(\frac{\pi}{2\delta}\delta\right) = \lim\limits_{z\to\delta^-} \ell'(g_{\mathbf{w}}(x), y) \\ \lim\limits_{z\to-\delta^+} \ell'(g_{\mathbf{w}}(x), y) & = & -\frac{1}{2} + \frac{1}{2}\sin\left(-\frac{\pi}{2\delta}\delta\right) = -1 & = & \lim\limits_{z\to-\delta^-} \ell'(g_{\mathbf{w}}(x), y) \end{array},$$

and its second derivative is also continuous in $z$,

$$\begin{array}{rcccl} \lim\limits_{z\to\delta^+} \ell''(g_{\mathbf{w}}(x), y) & = & 0 & = & \frac{\pi}{4\delta}\cos\left(\frac{\pi}{2\delta}\delta\right) = \lim\limits_{z\to\delta^-} \ell''(g_{\mathbf{w}}(x), y) \\ \lim\limits_{z\to-\delta^+} \ell''(g_{\mathbf{w}}(x), y) & = & \frac{\pi}{4\delta}\cos\left(-\frac{\pi}{2\delta}\delta\right) = 0 & = & \lim\limits_{z\to-\delta^-} \ell''(g_{\mathbf{w}}(x), y) \end{array}.$$

Since the cosine function is positive in the open interval $(-\frac{\pi}{2}, \frac{\pi}{2})$, the second derivative $\ell''$ is positive for all $|z| < \delta$. As, in addition, $\ell''$ equals 0 for all $|z| \geq \delta$, the trigonometric loss is also convex in $z \in \mathbb{R}$. □

*Proof of Proposition 2.4.* As before, we consider loss function $\ell^{\mathrm{l}}(g_{\mathbf{w}}(x), y)$ as a function in $z := yg_{\mathbf{w}}(x)$ with fixed $(x, y) \in \mathcal{X} \times \mathcal{Y}$, and show that $\ell^{\mathrm{l}}$ is convex and twice-continuously differentiable in $z \in \mathbb{R}$. Again, let $\ell'$ and $\ell''$ denote the first and second derivative, respectively, of loss function $\ell^{\mathrm{l}}$ with respect to $z$, where

$$\ell'(g_{\mathbf{w}}(x), y) = -\frac{\exp(-z)}{1+\exp(-z)} = \frac{1}{1+\exp(-z)} - 1 \tag{A.5}$$

and

$$\ell''(g_{\mathbf{w}}(x), y) = \frac{1}{1+\exp(-z)}\frac{\exp(-z)}{1+\exp(-z)} = -(\ell'(g_{\mathbf{w}}(x), y) + 1)\ell'(g_{\mathbf{w}}(x), y). \tag{A.6}$$

The logistic loss is infinitely-continuously differentiable as the second derivative, and consequently all higher order derivatives, can be expressed as a polynomial of the first derivative which is by itself continuous in $z$. The logistic loss is strictly convex as the second derivative is a product of two (strictly) positive terms. □

*Proof of Lemma 4.5.* The special structure of $\mathbf{D}$ and $\boldsymbol{\Upsilon}(\mathbf{w},\dot{\mathbf{x}})$ gives

$$\mathbf{J}_{\mathbf{r}}^{(1)}(\mathbf{w},\dot{\mathbf{x}}) = \boldsymbol{\Upsilon}(\mathbf{w},\dot{\mathbf{x}})^{\mathsf{T}} \begin{bmatrix} r_0\mathbf{G}_{-1} & r_0\mathbf{G}_{-1} \\ \operatorname{diag}(r_1,\ldots,r_n)\mathbf{G}_{+1} & \operatorname{diag}(r_1,\ldots,r_n)\mathbf{G}_{+1} \end{bmatrix} \boldsymbol{\Upsilon}(\mathbf{w},\dot{\mathbf{x}}).$$

From the assumption $\ell''_{-1,i} = \ell''_{+1,i}$ and the definition $r_0 = 1$, $r_i = \frac{c_{-1,i}}{c_{+1,i}} > 0$ for all $i = 1,\ldots,n$, it follows that $\mathbf{G}_{-1} = \operatorname{diag}(r_1,\ldots,r_n)\mathbf{G}_{+1}$, so that

$$\mathbf{J}_{\mathbf{r}}^{(1)}(\mathbf{w},\dot{\mathbf{x}}) = \boldsymbol{\Upsilon}(\mathbf{w},\dot{\mathbf{x}})^{\mathsf{T}} \begin{bmatrix} \mathbf{G}_{-1} & \mathbf{G}_{-1} \\ \mathbf{G}_{-1} & \mathbf{G}_{-1} \end{bmatrix} \boldsymbol{\Upsilon}(\mathbf{w},\dot{\mathbf{x}}),$$

which is obviously a symmetric matrix. Furthermore, we show that $\mathbf{z}^{\mathsf{T}}\mathbf{J}_{\mathbf{r}}^{(1)}(\mathbf{w},\dot{\mathbf{x}})\mathbf{z} \geq 0$ holds for all vectors $\mathbf{z} \in \mathbb{R}^{m+m\cdot n}$. To this end, let $\mathbf{z}$ be arbitrarily given, and partition this vector in $\mathbf{z} = \left[\mathbf{z}_0^{\mathsf{T}}, \mathbf{z}_1^{\mathsf{T}}, \ldots, \mathbf{z}_n^{\mathsf{T}}\right]^{\mathsf{T}}$ with $\mathbf{z}_i \in \mathbb{R}^m$ for all $i = 0,1,\ldots,n$. Then, a simple calculation shows that

$$\mathbf{z}^{\mathsf{T}}\mathbf{J}_{\mathbf{r}}^{(1)}(\mathbf{w},\dot{\mathbf{x}})\mathbf{z} = \sum_{i=1}^{n} \left(\mathbf{z}_0^{\mathsf{T}}\mathbf{x}_i + \mathbf{z}_i^{\mathsf{T}}\mathbf{w}\right)^2 c_{-1,i}\ell''_{-1,i} \geq 0,$$

since $\ell''_{-1,i} \geq 0$ for all $i = 1,\ldots,n$ in view of the assumed convexity of mapping $\ell_{-1}(z,y)$. Hence, $\mathbf{J}_{\mathbf{r}}^{(1)}(\mathbf{w},\dot{\mathbf{x}})$ is positive semi-definite. This matrix cannot be positive definite, since we have $\mathbf{z}^{\mathsf{T}}\mathbf{J}_{\mathbf{r}}^{(1)}(\mathbf{w},\dot{\mathbf{x}})\mathbf{z} = 0$ for the particular vector $\mathbf{z}$ defined by $\mathbf{z}_0 := -\mathbf{w}$ and $\mathbf{z}_i := \mathbf{x}_i$ for all $i = 1,\ldots,n$. □

*Proof of Lemma 4.6.* A sufficient and necessary condition for the (possibly asymmetric) matrix $\mathbf{J}_{\mathbf{r}}^{(2)}(\mathbf{w},\dot{\mathbf{x}})$ to be positive definite is that the Hermitian matrix

$$\mathbf{H}(\mathbf{w},\dot{\mathbf{x}}) := \mathbf{J}_{\mathbf{r}}^{(2)}(\mathbf{w},\dot{\mathbf{x}}) + \mathbf{J}_{\mathbf{r}}^{(2)}(\mathbf{w},\dot{\mathbf{x}})^{\mathsf{T}}$$

is positive definite, that is, all eigenvalues of $\mathbf{H}(\mathbf{w},\dot{\mathbf{x}})$ are positive. Let $\mathbf{D}^{\frac{1}{2}}$ denote the square root of $\mathbf{D}$, which is defined in such a way that the diagonal elements of $\mathbf{D}^{\frac{1}{2}}$ are the square roots of the corresponding diagonal elements of $\mathbf{D}$. Furthermore, we denote by $\mathbf{D}^{-\frac{1}{2}}$ the inverse of $\mathbf{D}^{\frac{1}{2}}$. Then, by Sylvester's law of inertia, the matrix

$$\bar{\mathbf{H}}(\mathbf{w},\dot{\mathbf{x}}) = \mathbf{D}^{-\frac{1}{2}} \left(\mathbf{J}_{\mathbf{r}}^{(2)}(\mathbf{w},\dot{\mathbf{x}}) + \mathbf{J}_{\mathbf{r}}^{(2)}(\mathbf{w},\dot{\mathbf{x}})^{\mathsf{T}}\right) \mathbf{D}^{-\frac{1}{2}} \tag{A.7}$$

$$= \mathbf{D}^{-\frac{1}{2}}\mathbf{D} \begin{bmatrix} \rho_{-1}\lambda_{-1}\mathbf{I}_m & c_{-1,1}\ell'_{-1,1}\mathbf{I}_m & \cdots & c_{-1,n}\ell'_{-1,n}\mathbf{I}_m \\ c_{+1,1}\ell'_{+1,1}\mathbf{I}_m & \rho_{+1}\lambda_{+1}\mathbf{I}_m & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ c_{+1,n}\ell'_{+1,n}\mathbf{I}_m & \mathbf{0} & \cdots & \rho_{+1}\lambda_{+1}\mathbf{I}_m \end{bmatrix} \mathbf{D}^{-\frac{1}{2}} +$$

$$\mathbf{D}^{-\frac{1}{2}} \begin{bmatrix} \rho_{-1}\lambda_{-1}\mathbf{I}_m & c_{+1,1}\ell'_{+1,1}\mathbf{I}_m & \cdots & c_{+1,n}\ell'_{+1,n}\mathbf{I}_m \\ c_{-1,1}\ell'_{-1,1}\mathbf{I}_m & \rho_{+1}\lambda_{+1}\mathbf{I}_m & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ c_{-1,n}\ell'_{-1,n}\mathbf{I}_m & \mathbf{0} & \cdots & \rho_{+1}\lambda_{+1}\mathbf{I}_m \end{bmatrix} \mathbf{D}\mathbf{D}^{-\frac{1}{2}} \tag{A.8}$$

has the same number of positive, zero, and negative eigenvalues as matrix $\mathbf{H}(\mathbf{w}, \dot{\mathbf{x}})$ itself. Hence, $\mathbf{J}_{\mathbf{r}}^{(2)}(\mathbf{w}, \dot{\mathbf{x}})$ is positive definite if, and only if, all eigenvalues of $\bar{\mathbf{H}}(\mathbf{w}, \dot{\mathbf{x}})$ are positive. By defining $\tilde{c}_i := \sqrt{c_{-1,i} c_{+1,i}}(\ell'_{-1,i} + \ell'_{+1,i})$, (A.8) can be rewritten as

$$\bar{\mathbf{H}}(\mathbf{w}, \dot{\mathbf{x}}) \quad = \quad \begin{bmatrix} 2\rho_{-1}\lambda_{-1}\mathbf{I}_m & \tilde{c}_1\mathbf{I}_m & \cdots & \tilde{c}_n\mathbf{I}_m \\ \tilde{c}_1\mathbf{I}_m & 2\rho_{+1}\lambda_{+1}\mathbf{I}_m & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{c}_n\mathbf{I}_m & \mathbf{0} & \cdots & 2\rho_{+1}\lambda_{+1}\mathbf{I}_m \end{bmatrix}.$$

Each eigenvalue $\lambda$ of this matrix satisfies

$$\left(\bar{\mathbf{H}}(\mathbf{w}, \dot{\mathbf{x}}) - \lambda \mathbf{I}_{m+m\cdot n}\right) \mathbf{v} = \mathbf{0}$$

for the corresponding eigenvector $\mathbf{v}^\mathsf{T} = \left[\mathbf{v}_0^\mathsf{T}, \mathbf{v}_1^\mathsf{T}, \ldots, \mathbf{v}_n^\mathsf{T}\right]$ with $\mathbf{v}_i \in \mathbb{R}^m$ for $i = 0, 1, \ldots, n$. This eigenvalue equation can be rewritten block-wise as

$$(2\rho_{-1}\lambda_{-1} - \lambda)\mathbf{v}_0 + \sum_{i=1}^{n} \tilde{c}_i \mathbf{v}_i \quad = \quad \mathbf{0}, \tag{A.9}$$

$$(2\rho_{+1}\lambda_{+1} - \lambda)\mathbf{v}_i + \tilde{c}_i \mathbf{v}_0 \quad = \quad \mathbf{0} \quad \forall\, i = 1, \ldots, n. \tag{A.10}$$

To compute all possible eigenvalues, we consider two cases: Firstly, we assume that $\mathbf{v}_0 = \mathbf{0}$. Then, (A.9) and (A.10) reduce to

$$\sum_{i=1}^{n} \tilde{c}_i \mathbf{v}_i = \mathbf{0} \quad \text{and} \quad (2\rho_{+1}\lambda_{+1} - \lambda)\mathbf{v}_i = \mathbf{0} \quad \forall\, i = 1, \ldots, n.$$

Since $\mathbf{v}_0 = \mathbf{0}$ and eigenvector $\mathbf{v} \neq \mathbf{0}$, at least one $\mathbf{v}_i$ is non-zero. This implies that $\lambda = 2\rho_{+1}\lambda_{+1}$ is an eigenvalue. Using the fact that the null space of the linear mapping $\mathbf{v} \mapsto \sum_{i=1}^{n} \tilde{c}_i \mathbf{v}_i$ has dimension $(n-1) \cdot m$ (we have $n \cdot m$ degrees of freedom counting all components of $\mathbf{v}_1, \ldots, \mathbf{v}_n$ and $m$ equations in $\sum_{i=1}^{n} \tilde{c}_i \mathbf{v}_i = \mathbf{0}$), it follows that $\lambda = 2\rho_{+1}\lambda_{+1}$ is an eigenvalue of multiplicity $(n-1) \cdot m$.

Now, we consider the second case where $\mathbf{v}_0 \neq \mathbf{0}$. We may further assume that $\lambda \neq 2\rho_{+1}\lambda_{+1}$ (since otherwise we get the same eigenvalue as before, just with a different multiplicity). We then get from (A.10) that

$$\mathbf{v}_i = -\frac{\tilde{c}_i}{2\rho_{+1}\lambda_{+1} - \lambda}\mathbf{v}_0 \quad \forall\, i = 1, \ldots, n, \tag{A.11}$$

and when substituting this expression into (A.9), we obtain

$$\left((2\rho_{-1}\lambda_{-1} - \lambda) - \sum_{i=1}^{n} \frac{\tilde{c}_i^2}{2\rho_{+1}\lambda_{+1} - \lambda}\right) \mathbf{v}_0 = \mathbf{0}. \tag{A.12}$$

where $\mathbf{v}_0 \neq \mathbf{0}$.

Equation A.12 and $\mathbf{v}_0 \neq \mathbf{0}$ imply

$$0 = 2\rho_{-1}\lambda_{-1} - \lambda - \frac{1}{2\rho_{+1}\lambda_{+1} - \lambda} \sum_{i=1}^{n} \tilde{c}_i^2$$

and, therefore,

$$0 = \lambda^2 - 2(\rho_{-1}\lambda_{-1} + \rho_{+1}\lambda_{+1})\lambda + 4\rho_{-1}\rho_{+1}\lambda_{-1}\lambda_{+1} - \sum_{i=1}^{n} \tilde{c}_i^2.$$

The roots of this quadratic equation are

$$\lambda = \rho_{-1}\lambda_{-1} + \rho_{+1}\lambda_{+1} \pm \sqrt{(\rho_{-1}\lambda_{-1} - \rho_{+1}\lambda_{+1})^2 + \sum_{i=1}^{n} \tilde{c}_i^2}, \tag{A.13}$$

and these are the remaining eigenvalues of $\bar{\mathbf{H}}(\mathbf{w}, \dot{\mathbf{x}})$, each of multiplicity $m$ since there are precisely $m$ linearly independent vectors $\mathbf{v}_0 \neq \mathbf{0}$ whereas the other vectors $\mathbf{v}_i$ $(i = 1, \ldots, n)$ are uniquely defined by (A.11) in this case. In particular, this implies that the dimensions of all three eigenspaces together are $(n-1)m + m + m = (n+1)m$, hence, other eigenvalues cannot exist. Since the eigenvalue $\lambda = 2\rho_{+1}\lambda_{+1}$ is positive by Remark 4.3, it remains to show that the roots in (A.13) are positive as well. By Assumption 4.3, we have

$$\sum_{i=1}^{n} \tilde{c}_i^2 = \sum_{i=1}^{n} c_{-1,i}c_{+1,i}(\ell'_{-1,i} + \ell'_{+1,i})^2 \leq 4\tau^2 \mathbf{c}_{-1}^{\mathsf{T}} \mathbf{c}_{+1} < 4\rho_{-1}\rho_{+1}\lambda_{-1}\lambda_{+1},$$

where $\mathbf{c}_v = [c_{v,1}, c_{v,2}, \cdots, c_{v,n}]^{\mathsf{T}}$. This inequality and Equation A.13 give

$$\begin{aligned} \lambda &= \rho_{-1}\lambda_{-1} + \rho_{+1}\lambda_{+1} \pm \sqrt{(\rho_{-1}\lambda_{-1} - \rho_{+1}\lambda_{+1})^2 + \sum_{i=1}^{n} \tilde{c}_i^2} \\ &> \rho_{-1}\lambda_{-1} + \rho_{+1}\lambda_{+1} - \sqrt{(\rho_{-1}\lambda_{-1} - \rho_{+1}\lambda_{+1})^2 + 4\rho_{-1}\rho_{+1}\lambda_{-1}\lambda_{+1}} \;=\; 0. \end{aligned}$$

As all eigenvalues of $\bar{\mathbf{H}}(\mathbf{w}, \dot{\mathbf{x}})$ are positive, matrix $\mathbf{H}(\mathbf{w}, \dot{\mathbf{x}})$ and, consequently, also the matrix $\mathbf{J}_{\mathbf{r}}^{(2)}(\mathbf{w}, \dot{\mathbf{x}})$ are positive definite. □

*Proof of Lemma 4.7.* By Assumption 4.3, either both players have equal instance-specific costs, or the partial gradient $\nabla_{\dot{\mathbf{x}}_i}\Omega_{+1}(\mathbf{x}, \dot{\mathbf{x}})$ of the *sender's* regularizer is independent of $\dot{\mathbf{x}}_j$ for all $j \neq i$ and $i = 1, \ldots, n$. Let us consider the first case where $c_{-1,i} = c_{+1,i}$, and consequently $r_i = 1$, for all $i = 1, \ldots, n$, so that

$$\mathbf{J}_{\mathbf{r}}^{(3)}(\mathbf{w}, \dot{\mathbf{x}}) = \begin{bmatrix} \rho_{-1}\nabla^2_{\mathbf{w},\mathbf{w}}\Omega_{-1}(\mathbf{w}) - \rho_{-1}\lambda_{-1}\mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & \rho_{+1}\nabla^2_{\dot{\mathbf{x}},\dot{\mathbf{x}}}\Omega_{+1}(\mathbf{x}, \dot{\mathbf{x}}) - \rho_{+1}\lambda_{+1}\mathbf{I}_{m\cdot n} \end{bmatrix}.$$

The eigenvalues of this block diagonal matrix are the eigenvalues of the matrix

$\rho_{-1}(\nabla^2_{\mathbf{w},\mathbf{w}}\Omega_{-1}(\mathbf{w}) - \lambda_{-1}\mathbf{I}_m)$ together with those of $\rho_{+1}(\nabla^2_{\dot{\mathbf{x}},\dot{\mathbf{x}}}\Omega_{+1}(\mathbf{x},\dot{\mathbf{x}}) - \lambda_{+1}\mathbf{I}_{m\cdot n})$. From the definition of $\lambda_v$ in (4.6) and (4.7) follows that these matrices are positive semi-definite for $v \in \{-1,+1\}$. Hence, $\mathbf{J}^{(3)}_{\mathbf{r}}(\mathbf{w},\dot{\mathbf{x}})$ is positive semi-definite as well.

Now, let us consider the second case where we assume that $\nabla_{\dot{\mathbf{x}}_i}\Omega_{+1}(\mathbf{x},\dot{\mathbf{x}})$ is independent of $\dot{\mathbf{x}}_j$ for all $j \neq i$. Hence, $\nabla^2_{\dot{\mathbf{x}}_i,\dot{\mathbf{x}}_j}\Omega_{+1}(\mathbf{x},\dot{\mathbf{x}}) = \mathbf{0}$ for all $j \neq i$, so that

$$\mathbf{J}^{(3)}_{\mathbf{r}}(\mathbf{w},\dot{\mathbf{x}}) = \begin{bmatrix} \rho_{-1}\tilde{\Omega}_{-1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \rho_{+1}\frac{c_{-1,1}}{c_{+1,1}}\tilde{\Omega}_{+1,1} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \rho_{+1}\frac{c_{-1,n}}{c_{+1,n}}\tilde{\Omega}_{+1,n} \end{bmatrix},$$

where $\tilde{\Omega}_{-1} := \nabla^2_{\mathbf{w},\mathbf{w}}\Omega_{-1}(\mathbf{w}) - \lambda_{-1}\mathbf{I}_m$ and $\tilde{\Omega}_{+1,i} = \nabla^2_{\dot{\mathbf{x}}_i,\dot{\mathbf{x}}_i}\Omega_{+1}(\mathbf{x},\dot{\mathbf{x}}) - \lambda_{+1}\mathbf{I}_m$. The eigenvalues of this block diagonal matrix are again the union of the eigenvalues of the single blocks $\rho_{-1}\tilde{\Omega}_{-1}$ and $\rho_{+1}\frac{c_{-1,i}}{c_{+1,i}}\tilde{\Omega}_{+1,i}$ for $i = 1,\ldots,n$. As in the first part of the proof, $\tilde{\Omega}_{-1}$ is positive semi-definite. The eigenvalues of $\nabla^2_{\dot{\mathbf{x}},\dot{\mathbf{x}}}\Omega_{+1}(\mathbf{x},\dot{\mathbf{x}})$ are the union of all eigenvalues of $\nabla^2_{\dot{\mathbf{x}}_i,\dot{\mathbf{x}}_i}\Omega_{+1}(\mathbf{x},\dot{\mathbf{x}})$. Hence, each of these eigenvalues is larger or equal to $\lambda_{+1}$ and thus, each block $\tilde{\Omega}_{+1,i}$ is positive semi-definite. The factors $\rho_{-1} > 0$ and $\rho_{+1}\frac{c_{-1,i}}{c_{+1,i}} > 0$ are multipliers that do not affect the definiteness of the blocks, and consequently, $\mathbf{J}^{(3)}_{\mathbf{r}}(\mathbf{w},\dot{\mathbf{x}})$ is positive semi-definite as well. □

*Proof of Corollary 4.9.* By Definition 4.1, both players employ the logistic loss with $\ell_{-1}(z,y) := \ell^{\mathrm{l}}(z,y)$ and $\ell_{+1}(z,y) := \ell^{\mathrm{l}}(z,-1)$ and the $l^2$-norm regularizers in (4.16) and (4.17), respectively. Let

$$\begin{array}{rcl|rcl} \ell'_{-1}(z,y) & = & -y\frac{1}{1+\exp(yz)} & \ell'_{+1}(z,y) & = & \frac{1}{1+\exp(-z)} \\ \ell''_{-1}(z,y) & = & \frac{1}{1+\exp(z)}\frac{1}{1+\exp(-z)} & \ell''_{+1}(z,y) & = & \frac{1}{1+\exp(z)}\frac{1}{1+\exp(-z)} \end{array} \tag{A.14}$$

denote the first and second derivatives of the players' loss functions with respect to $z \in \mathbb{R}$. Further, let

$$\begin{array}{rcl|rcl} \nabla_{\mathbf{w}}\Omega_{-1}(\mathbf{w}) & = & \mathbf{w} & \nabla_{\dot{\mathbf{x}}}\Omega_{+1}(\mathbf{x},\dot{\mathbf{x}}) & = & \frac{1}{n}(\dot{\mathbf{x}} - \mathbf{x}) \\ \nabla^2_{\mathbf{w},\mathbf{w}}\Omega_{-1}(\mathbf{w}) & = & \mathbf{I}_m & \nabla^2_{\dot{\mathbf{x}},\dot{\mathbf{x}}}\Omega_{+1}(\mathbf{x},\dot{\mathbf{x}}) & = & \frac{1}{n}\mathbf{I}_{m\cdot n} \end{array} \tag{A.15}$$

denote the gradients and Hessians of the players' regularizers. Assumption 4.2 holds as:

1. According to Proposition 2.4, $\ell_v(z,y)$ is convex and twice-continuously differentiable with respect to $z$ for $v \in V$ and fixed $y$.

2. The Hessians of the players' regularizers are fixed, positive definite matrices and, consequently, both regularizers are twice-continuously differentiable and uniformly strongly convex in $\mathbf{w} \in \mathcal{W}$ and $\dot{\mathbf{x}} \in \boldsymbol{\phi}(\mathcal{X})^n$ (for any fixed $\mathbf{x} \in \boldsymbol{\phi}(\mathcal{X})^n$), respectively.

3. By Definition 4.1, the players' action sets are non-empty, compact, and convex subsets of finite-dimensional Euclidean spaces.

Assumption 4.3 holds as for all $z \in \mathbb{R}$ and $y \in \mathcal{Y}$:

1. The second derivatives of $\ell_{-1}(z, y)$ and $\ell_{+1}(z, y)$ in (A.14) are equal.

2. The sum of the first derivatives of the loss functions is bounded,

$$\begin{aligned} \ell'_{-1}(z, y) + \ell'_{+1}(z, y) &= -y\frac{1}{1 + \exp(yz)} + \frac{1}{1 + \exp(-z)} \\ &= \begin{cases} \frac{1-\exp(-z)}{1+\exp(-z)} & , \text{if } y = +1 \\ \frac{2}{1+\exp(-z)} & , \text{if } y = -1 \end{cases} \in (-1, 2), \end{aligned}$$

   which together with Equation 4.10 gives

$$\tau = \sup_{(\mathbf{x},y)\in\boldsymbol{\phi}(\mathcal{X})\times\mathcal{Y}} \frac{1}{2} \left|\ell'_{-1}(g_{\mathbf{w}}(\mathbf{x}), y) + \ell'_{+1}(g_{\mathbf{w}}(\mathbf{x}), y)\right| < 1.$$

   The supremum $\tau$ is strictly less than 1, since $g_{\mathbf{w}}(\mathbf{x})$ is finite for compact action sets $\mathcal{W}$ and $\boldsymbol{\phi}(\mathcal{X})^n$. The smallest eigenvalues of the players' regularizers are $\lambda_{-1} = 1$ and $\lambda_{+1} = \frac{1}{n}$, so that inequalities

$$\rho_{-1}\rho_{+1} \geq n\mathbf{c}_{-1}^{\mathsf{T}}\mathbf{c}_{+1} > \tau^2 \frac{1}{\lambda_{-1}\lambda_{+1}} \mathbf{c}_{-1}^{\mathsf{T}}\mathbf{c}_{+1}$$

   hold.

3. The partial gradient $\nabla_{\dot{\mathbf{x}}_i}\Omega_{+1}(\mathbf{x}, \dot{\mathbf{x}}) = \frac{1}{n}(\dot{\mathbf{x}}_i - \mathbf{x}_i)$ of the data generator's regularizer is independent of $\dot{\mathbf{x}}_j$ for all $j \neq i$ and $i = 1, \ldots, n$.

As Assumptions 4.2 and 4.3 are satisfied, the existence of a unique Nash equilibrium follows immediately from Theorem 4.8. □

*Proof of Corollary 4.10.* By Definition 4.2, both players employ the trigonometric loss with $\ell_{-1}(z, y) := \ell^{\mathrm{t}}(z, y)$ and $\ell_{+1}(z, y) := \ell^{\mathrm{t}}(z, -1)$ and the regularizers in (4.16) and (4.17), respectively. Assumption 4.2 holds as:

1. According to Proposition 2.3, $\ell^{\mathrm{t}}(z, y)$, and consequently $\ell_{-1}(z, y)$ and $\ell_{+1}(z, y)$, are convex and twice-continuously differentiable with respect to $z \in \mathbb{R}$ (for any fixed $y \in \{-1, +1\}$).

2. The regularizers of the Nash support vector machine are equal to that of the Nash logistic regression and possess the same properties as in Theorem 4.9.

3. By Definition 4.2, the players' action sets are non-empty, compact, and convex subsets of finite-dimensional Euclidean spaces.

Assumption 4.3 holds:

1. The second derivatives of $\ell_{-1}(z, y)$ and $\ell_{+1}(z, y)$ are equal for all $z \in \mathbb{R}$ since

$$\ell^{\mathrm{t}\prime\prime}(z, y) = \begin{cases} 0 & , \text{if } |z| > \delta \\ \frac{\pi}{4\delta} \cos\left(\frac{\pi}{2\delta} z\right) & , \text{if } |z| \leq \delta \end{cases}$$

   does not dependent on $y \in \mathcal{Y}$ (*cf.* proof of Proposition 2.3).

2. The sum of the first derivatives of the loss functions is bounded as for $y = -1$:

$$\ell'_{-1}(z, -1) + \ell'_{+1}(z, -1) = 2\ell^{\mathrm{t}\prime}(z, -1) = \begin{cases} 2 & , \text{if } z > \delta \\ 1 - \sin\left(-\frac{\pi}{2\delta} z\right) & , \text{if } |z| \leq \delta \\ 0 & , \text{if } z < -\delta \end{cases} \in [0, 2],$$

   and for $y = +1$:

$$\ell'_{-1}(z, +1) + \ell'_{+1}(z, +1) = \begin{cases} 1 & , \text{if } z > \delta \\ \sin\left(\frac{\pi}{2\delta} z\right) & , \text{if } |z| \leq \delta \\ -1 & , \text{if } z < -\delta \end{cases} \in [-1, 1].$$

   Together with Equation 4.10, it follows that

$$\tau = \sup_{(\mathbf{x}, y) \in \boldsymbol{\phi}(\mathcal{X}) \times \mathcal{Y}} \frac{1}{2} \left| \ell'_{-1}(g_{\mathbf{w}}(\mathbf{x}), y) + \ell'_{+1}(g_{\mathbf{w}}(\mathbf{x}), y) \right| \leq 1.$$

   The smallest eigenvalues of the players' regularizers are $\lambda_{-1} = 1$ and $\lambda_{+1} = \frac{1}{n}$, so that inequalities

$$\rho_{-1}\rho_{+1} > n\mathbf{c}_{-1}^{\mathsf{T}}\mathbf{c}_{+1} \geq \tau^2 \frac{1}{\lambda_{-1}\lambda_{+1}} \mathbf{c}_{-1}^{\mathsf{T}}\mathbf{c}_{+1}$$

   hold.

3. As for Nash logistic regression, the partial gradient $\nabla_{\dot{\mathbf{x}}_i} \Omega_{+1}(\mathbf{x}, \dot{\mathbf{x}}) = \frac{1}{n} (\dot{\mathbf{x}}_i - \mathbf{x}_i)$ of the data generator's regularizer is independent of $\dot{\mathbf{x}}_j$ for all $j \neq i$ and $i = 1, \ldots, n$.

Because Assumptions 4.2 and 4.3 are satisfied, the existence of a unique Nash equilibrium follows immediately from Theorem 4.8. □

# Notation

The general ideas underlying the notation are as follows. Spaces, halfspaces, and sets are denoted by italic capitals like $\mathcal{X}$. Italic lower-case characters such as $\tau$ and $n$ denote real or integer scalar values whereas composed objects such as tuples, vectors, and matrices are denoted by non-italic bold characters like $\mathbf{x}$ and $\mathbf{X}$. Functions of the real Euclidean space are denoted by non-italic lower case characters such as $f(\cdot)$ and $\theta(\cdot)$, and general mappings and multi-dimensional functions are denoted by bold characters like $\mathbf{g}(\cdot)$ and $\boldsymbol{\phi}(\cdot)$. The notions $f[\cdot]$ and $\boldsymbol{\phi}[\cdot]$ refer to functionals. Italic bold capitals such as $\boldsymbol{X}$ denote random variables.

To separate between training and test variables we use a dot such as $\dot{x}$, and to indicate an estimate of a quantity we use the hat symbol, *e.g.,* $\hat{\theta}$ is an estimate of $\theta$. We use (1) as a short form of Equation 1 and use the Latin abbreviations *cf.* (confer), et al. (et alii), etc. (et cetera), *e.g.* (exempli gratia), and *i.e.* (id est). The abbreviation *i.i.d.* stands for *independent and identically distributed*.

In the following, we list frequently used functions and variables.

| | |
|---|---|
| $\mathbf{w}$ | Parameter vector $\mathbf{w} := [w_1, \ldots, w_m]^\mathsf{T} \in \mathcal{W}$ of weights $w_j$, page 17 |
| $\Delta$ | Error functional $\Delta : F \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$ measures the disagreement between label $y$ and prediction $h_\mathbf{w}(x)$ with $h_\mathbf{w}(x) = \operatorname{argmax}_{y \in \mathcal{Y}} f_\mathbf{w}(x, y)$; for binary classification we define $\Delta[f_\mathbf{w}, x, y] := c(x, y)\ell(g_\mathbf{w}(x), y)$ with $g_\mathbf{w}(x) := \phi(x)^\mathsf{T}\mathbf{w}$, page 12 |
| $\mathrm{E}_{\boldsymbol{X}}$ | Expectation operator to compute the expected value $\mathrm{E}_{\boldsymbol{X}}[f(x)]$ of some function $f$ under distribution $P_{\boldsymbol{X}}$ with $\mathrm{E}_{\boldsymbol{X}}[f(x)] = \int f(x)\, p_{\boldsymbol{X}}(x)\, \mathrm{d}x$, page 13 |
| $\ell$ | Loss function $\ell : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}^+$ which quantifies the induced loss for a given decision function value and a target value, page 18 |
| $\ell'$ | First derivative of the loss function $\ell(z, y)$ with respect to $z \in \mathbb{R}$, page 45 |
| $\ell''$ | Second derivative of the loss function $\ell(z, y)$ with respect to $z \in \mathbb{R}$, page 45 |
| $\gamma$ | Data generation model, page 29 |
| $\hat{\theta}_v$ | Player $v$'s cost function; $\hat{\theta}_{-1}(\mathbf{w}, \dot{D})$ denotes the learner's prediction costs and $\hat{\theta}_{+1}(\mathbf{w}, \dot{D})$ denotes the data generator's prediction costs, page 38 |
| $\mathcal{H}$ | Feature space with $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{H}$; for linear decision functions, $\mathcal{H}$ is typically a Hilbert space such as $\mathbb{R}^m$, page 17 |

| | |
|---|---|
| $\mathcal{W}$ | Parameter space such as $\mathbb{R}^m$, page 17 |
| $\mathcal{X}$ | Input space, page 11 |
| $x$ | Object with $x \in \mathcal{X}$, page 11 |
| $\mathcal{Y}$ | Output space, page 11 |
| $y$ | Target variable with $y \in \mathcal{Y}$, page 11 |
| $z$ | Selector variable with $z \in \{-1, +1\}$ which indicates whether an object was drawn from the training distribution ($z = +1$) or the test distribution ($z = -1$); only used in Chapter 6, page 86 |
| $\kappa$ | Resampling-weight function $\kappa : \mathcal{X} \to \mathbb{R}^+$, page 84 |
| L | Theoretical label likelihood, page 83 |
| $\boldsymbol{\phi}$ | Feature mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{H}$, for instance, $\boldsymbol{\phi}(x) = [\phi_1(x), \ldots, \phi_m(x)]^\mathsf{T}$ where $\phi_i(x) \in \mathbb{R}$ are referred as features, page 17 |
| $\mathbb{N}$ | Set of natural numbers, page 17 |
| $\nabla_{\mathbf{x}}$ | Gradient operator to compute the partial gradient of some function with respect to $\mathbf{x}$, page 22 |
| $\nabla^2_{\mathbf{x},\mathbf{y}}$ | Hessian operator to compute the partial Hessian of some function with respect to $\mathbf{x}$ and $\mathbf{y}$ where $\nabla^2_{\mathbf{x},\mathbf{y}} f = \nabla_{\mathbf{y}}(\nabla_{\mathbf{x}} f)$, page 45 |
| $\Omega$ | Regularizer of model parameters $\mathbf{w}$ and data transformation $\dot{D}$, respectively, page 13 |
| $\mathbb{R}$ | Set of real numbers, page 12 |
| $\mathbb{R}^+$ | Set of positive real numbers, page 12 |
| $\rho$ | Regularization parameter $\rho \in \mathbb{R}^+$ which determines the amount of regularization, page 13 |
| $\theta$ | Generalization error with $\theta[f, p_{\boldsymbol{XY}}] = \mathrm{E}_{\boldsymbol{XY}}[\Delta[f, x, y]]$, page 12 |
| $c$ | Object- and target-specific cost factors $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$, page 18 |
| $D$ | Set of data points $(x_i, y_i)$ which are pairs of objects $x_i \in \mathcal{X}$ and targets $y_i \in \mathcal{Y}$, page 11 |
| $F$ | Decision function space, page 13 |

$f_{\mathbf{w}}$ Decision function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ parameterized by weight vector $\mathbf{w}$ which is used to restrict prediction models to $h_{\mathbf{w}}(x) = \operatorname{argmax}_{y \in \mathcal{Y}} f_{\mathbf{w}}(x, y)$; for binary classification where $\mathcal{Y} = \{-1, +1\}$, we typically employ a linear decision function $f_{\mathbf{w}}(x, y) := y g_{\mathbf{w}}(x)$ with $h_{\mathbf{w}}(x) = \operatorname{sign} g_{\mathbf{w}}(x)$ and $g_{\mathbf{w}}(x) = \mathbf{w}^{\mathsf{T}} \boldsymbol{\phi}(x)$, page 12

$H$ Hypothesis space, page 14

$h_{\mathbf{w}}$ Prediction model $h : \mathcal{X} \to \mathcal{Y}$ parameterized by weight vector $\mathbf{w}$, which assigns any object $x \in \mathcal{X}$ to target $y \in \mathcal{Y}$, page 11

$k$ Positive semi-definite kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which measures the "similarity" of any two objects from the input space $\mathcal{X}$, page 24

$P_{\boldsymbol{X}}(x)$ *Cumulative distribution function* (CDF) of random variable $\boldsymbol{X}$ at $x$; abbreviated form of $P(\boldsymbol{X} \leq x)$, page 11

$p_{\boldsymbol{X}}(x)$ Density of random variable $\boldsymbol{X}$ at $x$, if $\boldsymbol{X}$ is a continuous random variable, the term density refers to the *probability density function* (PDF) which equals $P(x < \boldsymbol{X} < x + \mathrm{d}x)$, whereas otherwise it refers to the *probability mass function* (PMF) with $P(\boldsymbol{X} = x)$, page 11

$s$ Selector function $s : \mathcal{X} \to \{-1, +1\}$ which assigns object $x$ to selector $z$, page 86

$L^2$ Euclidean norm of a function, page 22

$\ell^2$ Euclidean norm of a vector, page 23

A main assumption in machine learning is that the data which are used to build a predictive model are governed by the same distribution as the data which the predictive model will be exposed to at application time. This condition is violated when future data are generated in response to the presence of a predictive model which is the case, for instance, in email spam filtering.

In this thesis, we establish the concept of prediction games to handle such tasks: We model the interaction between a learner, who builds the predictive model, and a data generator, who controls the process of data generation, as an one-shot game. The game-theoretic framework enables us to explicitly model the players' interests, their possible actions, and their level of knowledge about each other. We study three instances of prediction games which differ regarding the order in which the players decide for their action.

In case studies on email spam filtering we empirically explore properties of all derived models. We show that spam filters resulting from prediction games in the majority of cases outperform other existing baseline methods.

ISBN 978-3-86956-203-2