Institut für Informatik
Professur Maschinelles Lernen

Discriminative Classification Models for Internet Security: Mitigating Email Spam and HTTP-Layer DDoS Attacks

Kumulative Dissertation

zur Erlangung des akademischen Grades
„Doctor rerum naturalium“
(Dr. rer. nat.)
in der Wissenschaftsdisziplin
„Maschinelles Lernen“

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Universität Potsdam

von
Uwe Dick

Potsdam, den 09. Mai 2016

Abstract

Services that operate over the Internet are under constant threat of being exposed to fraudulent use. Maintaining good user experience for legitimate users often requires the classification of entities as malicious or legitimate in order to initiate countermeasures. As an example, inbound email spam filters decide for spam or non-spam. They can base their decision on both the content of each email as well as on features that summarize prior emails received from the sending server. In general, discriminative classification methods learn to distinguish positive from negative entities. Each decision for a label may be based on features of the entity and related entities. When labels of related entities have strong interdependencies—as can be assumed e.g. for emails being delivered by the same user—classification decisions should not be made independently and dependencies should be modeled in the decision function. This thesis addresses the formulation of discriminative classification problems that are tailored for the specific demands of the following three Internet security applications. Theoretical and algorithmic solutions are devised to protect an email service against flooding of user inboxes, to mitigate abusive usage of outbound email servers, and to protect web servers against distributed denial of service attacks.

In the application of filtering an inbound email stream for unsolicited emails, utilizing features that go beyond each individual email's content can be valuable. Information about each sending mail server can be aggregated over time and may help in identifying unwanted emails. However, while this information will be available to the deployed email filter, some parts of the training data that are compiled by third party providers may not contain this information. The missing features have to be estimated at training time in order to learn a classification model. In this thesis an algorithm is derived that learns a decision function that integrates over a distribution of values for each missing entry. The distribution of missing values is a free parameter that is optimized to learn an optimal decision function.

The outbound stream of emails of an email service provider can be separated by the customer IDs that ask for delivery. All emails that are sent by the same ID in the same period of time are related, both in content and in label. Hijacked customer accounts may send batches of unsolicited emails to other email providers, which in turn might blacklist the sender's email servers after detection of incoming spam emails. The risk of being blocked from further delivery depends on the rate of outgoing unwanted emails and the duration of high spam sending rates. An optimization problem is developed that minimizes the expected cost for the email provider by learning a decision function that assigns a limit on the sending rate to customers based on the each customer's email stream.

Identifying attacking IPs during HTTP-level DDoS attacks allows to block those IPs from further accessing the web servers. DDoS attacks are usually carried out by infected clients that are members of the same botnet and show similar traffic patterns. HTTP-level attacks aim at exhausting one or more resources of the web server infrastructure, such as CPU time. If the joint set of attackers cannot increase resource usage close to the maximum capacity, no effect will be experienced by legitimate users of hosted web sites. However, if the additional load raises the computational burden towards the critical range, user experience will degrade until service may be unavailable altogether. As the loss of missing one attacker depends on block decisions for other attackers—if most other attackers are detected, not blocking one client will likely not be harmful—a structured output model has to be learned. In this thesis an algorithm is developed that learns a structured prediction decoder that searches the space of label assignments, guided by a policy.

Each model is evaluated on real-world data and is compared to reference methods. The results show that modeling each classification problem according to the specific demands of the task improves performance over solutions that do not consider the constraints inherent to an application.

Zusammenfassung

Viele Dienste im Internet benötigen zur Gewährleistung ihrer Erreichbarkeit die Möglichkeit, Entitäten als entweder gefährlich oder harmlos zu klassifizieren. Diskriminative Methoden des maschinellen Lernens verwenden Features von Entitäten oder Entitätengruppen, um zwischen positiven und negativen Labels zu unterscheiden. So können beispielsweise Email-Spamfilter Entscheidungen aufgrund sowohl des Inhalts der Email als auch von Informationen treffen, die vorherige Emails des gleichen versendenden Servers zusammenfassen. Darüber hinaus sind Labels zueinander in Verbindung stehender Entitäten, wie z.B. Emails des gleichen Nutzers, oftmals nicht unabhängig, so dass auch Klassifikationsentscheidungen nicht unabhängig getroffen werden sollten. Diese Arbeit beschäftigt sich mit der Formulierung diskriminativer Klassifikationsprobleme, die den speziellen Anforderungen von drei Internetsicherheitsanwendungen Rechnung tragen. Theoretische und algorithmische Lösungen zum Spamschutz von Nutzer-Inboxen eines Emailanbieters, zum Schutz von ausgehenden Emailservern gegen Missbrauch und zur Abwehr von *Distributed Denial of Service*-Attacken auf Webserver werden entwickelt.

Beim Säubern der bei einem Emailanbieter eingehenden Menge von Emails von ungewollten Emails wie Spam können Informationen, die über den Inhalt einzelner Emails hinausgehen, von großem Nutzen sein. Etwa können Informationen über einen Mailserver zeitlich aggregiert und zum Klassifizieren neuer Emails des gleichen Servers verwendet werden. Diese Informationen sind in der Regel nur für Emails verfügbar, die vom Emailanbieter selbst empfangen werden, und fehlen bei Datensätzen, die extern gesammelte Emails beinhalten. Während des Trainings eines Spamklassifikators müssen diese Features entsprechend geschätzt werden. In dieser Arbeit wird ein Algorithmus entwickelt, der eine Entscheidungsfunktion lernt, die über eine Verteilung von fehlenden Werten integriert. Die Verteilung ist ein freier Parameter, der während des Lernens der Entscheidungsfunktion optimiert wird.

Der Strom ausgehender Emails eines Emailanbieters setzt sich zusammen aus Emails einzelner Kunden. Alle Emails, die vom gleichen Kunden im gleichen Zeitraum gesendet werden, sind sowohl bzgl. Inhalt als auch Label abhängig. Kompromittierte Kundenaccounts können beispielsweise Batches von Spams an andere Emailanbieter schicken. Nach erfolgter Spamerkennung könnten diese Anbieter die Mailserver des sendenden Anbieters auf eine Blacklist setzen und somit am Versand weiterer Emails hindern. Das Risiko einer solchen Blockierung ist abhängig von der Rate ausgehender ungewollter Emails und der Dauer hoher Senderaten. Es wird ein Optimierungsproblem entwickelt, das die erwarteten Kosten des Emailproviders minimiert, indem eine Entscheidungsfunktion gelernt wird, die die erlaubte Versenderate von Kunden aufgrund der gesendeten Emails dynamisch einstellt.

Um angreifende IPs während einer HTTP-Level-DDoS-Attacke zu blockieren, müssen sie als solche erkannt werden. DDoS-Angriffe werden üblicherweise von Clients durchgeführt, die dem gleichen Botnet angehören und ähnliche Traffic-Muster aufweisen. HTTP-Level-Angriffe zielen darauf, eine oder mehrere Ressourcen der Webserverinfrastruktur, wie etwa CPU-Zeit, aufzubrauchen. Für legitime Besucher ergeben sich erst dann Einschränkungen der User Experience, bis hin zur Unerreichbarkeit der Webseite, wenn Angreifer den Ressourcenverbrauch in die Nähe oder über die Maximalkapazität steigern können. Dieser durch einen Angreifer verursachte Verlust hängt von Entscheidungen für andere Angreifer ab; werden z.B. die meisten anderen Angreifer erkannt, wird ein nicht geblockter Angreifer kaum Schaden anrichten. Es wird deshalb ein Algorithmus entwickelt, der einen Dekodierer für strukturierte Vorhersagen trainiert, der, geleitet durch eine Policy, den Raum der gemeinsamen Labelzuweisungen durchsucht.

Alle Modelle werden auf industriellen Daten evaluiert und mit Referenzmethoden verglichen. Die Ergebnisse zeigen, dass anforderungsspezifische Modellierung der Klassifikationsprobleme die Performance gegenüber den Vergleichsmethoden verbessert.

Acknowledgements

First, I would like to thank my supervisor Tobias Scheffer for giving me the opportunity to pursue my PhD in his research group and for his support and guidance. I am also grateful to STRATO AG for helping to finance my studies via joint research projects with the University of Potsdam and for giving me the opportunity to translate theory to practice. I would like to thank all members, past and current, of our research group for an enjoyable working environment. Finally, many thanks to friends, family, and L for all the rest.

Contents

1 Introduction **1**
1.1 Inbound Spam Classification with Missing Attributes 2
1.2 Outbound Spam Filtering 3
1.3 DDoS Attacker Detection 5

2 Learning from Incomplete Data with Infinite Imputations **7**
2.1 Introduction 7
2.2 Problem Setting 8
2.3 Learning from Incomplete Data in One Step 9
2.4 Solving the Optimization Problem 9
2.4.1 Optimal Solution with Finite Combination 9
2.4.2 Iterative Optimization Algorithm 10
2.5 Example Learners 11
2.5.1 Two Standard Learning Algorithms 11
2.5.2 Regularizing towards Prior Belief in Feature Space 11
2.5.3 Imputing the Mean in Feature Space 11
2.6 Empirical Evaluation 12
2.6.1 Classification 12
2.6.2 Regression 14
2.7 Conclusion 14
2.8 Acknowledgments 14
2.9 References 14

3 Throttling Poisson Processes **15**
3.1 Introduction 15
3.2 Poisson Process Model 16
3.2.1 Derivation of Empirical Loss 16
3.3 Erlang Learning Model 17
3.4 Prior Work and Reference Methods 19
3.5 Application 20
3.5.1 Results 21
3.6 Conclusion 22
3.7 Acknowledgments 22
3.8 References 22

4 Learning to Control a Structured-Prediction Decoder for Detection of HTTP-Layer DDoS Attackers **24**
4.1 Introduction 24

4.2 Problem Setting, Motivating Appliation 26
4.3 Anomaly Detection 27
4.3.1 Problem Setting for Anomaly Detection 27
4.3.2 Support Vector Data Description 27
4.4 Independent Classification 28
4.4.1 Problem Setting for Independent Classification 28
4.4.2 Logistic Regression 28
4.5 Structured Prediction with Approximate Inference 28
4.5.1 Problem Setting for Structured Prediction with Approximate Inference 29
4.5.2 Iterative Classification Algorithm 29
4.6 Structured Prediction with a Parametric Decoder 29
4.6.1 Problem Setting for Structured Prediction with Parametric Decoder 30
4.6.2 HC Search 30
4.6.3 Online Policy-Gradient Decoder 31
4.7 Identification of DDoS Attackers 36
4.7.1 Cost Function 36
4.7.2 Loss Function 37
4.7.3 Action Space and Stochastic Policy 38
4.7.4 Feature Representations 38
4.8 Experimental Study 40
4.8.1 Data Collection 40
4.8.2 Experimental Setting 41
4.8.3 Reference Methods 41
4.8.4 Results 42
4.8.5 Analysis 43
4.8.6 Feature Relevance 44
4.8.7 Execution Time 44
4.9 Discussion and Related Work 45
4.10 Conclusion 46
4.11 References 46

5 Discussion **49**
5.1 Application-Specific Related Work 49
5.2 Application-Oriented Modeling and Algorithms 50
5.2.1 Inbound Spam Filtering with Incomplete Data 51
5.2.2 Outbound Spam Filtering with Non-Separable Loss 51
5.2.3 DDoS Filtering with Structured Prediction Models 52
5.3 Empirical Evaluation 53

Chapter 1

Introduction

Email and web hosting are both examples of services that operate over the Internet and are provided to customers by a variety of companies and organizations. Each service is intended to essentially serve or send data from legitimate customers—e.g. business web sites hosted at a web hosting company or private emails sent via a free email service—to eligible recipients—the web browser of a potential customer of the business web site or the mail server of a friend's email service provider. Reliability of provided services is key to the success of any service provider. However, each service is under constant threat of being exposed to fraudulent use that might render it unusable to legitimate customers. Web sites can be exposed to high usage—often initiated by distributed sources corresponding to botnets—that might lead to reduced availability and suboptimal user experience. In extreme cases, web sites will not be reachable for legitimate users, leading to potentially lost business for the customers of the web hosting firm and, ultimately, for the hosting company itself. Email services can be exploited by flooding it with unwanted and illegitimate emails such as spam or phishing emails. Without appropriate countermeasures, such as employing content based spam filters, and thereby cleaning the email inbox from those unwanted emails, the service would often become unusable. On the other hand, the email infrastructure of an email service provider can also be exploited by senders of unsolicited bulk email to deliver their messages. Customer accounts can be hijacked and used for delivering large amounts of emails in short time spans, or wrongly configured scripts on web sites can be utilized by spam bots. Sending large amounts of unwanted emails can lead to IP addresses of outgoing email servers to be blacklisted by other email providers, thus effectively disrupting parts of the service.

In order to maintain service and good customer experience in all those cases, countermeasures have to be taken by service providers. Much work has been done on protecting web server infrastructure against denial of service (DoS) attacks, especially when multiple attacking IPs are involved. The effects of such distributed denial of service (DDoS) attacks can often be mitigated by allowing servers to dynamically adjust TCP timeouts or dynamically provide additional resources. In certain types of TCP layer attacks, IP addresses can be spoofed. However, HTTP layer attacks need a valid TCP connection to be established and a corresponding DDoS mitigation system should aim at detecting client IPs that participate in an attack. After detection of attacking clients, those could either be blacklisted and blocked from further accessing the web servers, or a throttling mechanism could be implemented that limits the number of allowed connections of detected clients. The next section introduces a problem formulation for learning a DDoS attacker detection mechanism that decides for attackers based on HTTP traffic features of groups of clients.

Learning content based email spam filters has shown its merit for over a decade, and email inboxes that consist to a large part of spam emails are a very unusual sight nowadays. However, performance of spam filters depends to a large part on size and diversity of the training data which should be a representative sample of both spam and non-spam (ham) emails. In order to assemble a good data set, it may be necessary to collect data from different data sources, mainly from the in-house email service but also from publicly available data sets and other sources. However, assembling data from different sources can lead to problems of inconsistency of the compiled data set. In particular, certain meta-information on sender reputation will be missing in external sources. Section 1.1 introduces this problem formulation in more depth.

Content-based filters for filtering the outbound stream of emails of an email provider can be used to effectively control the number of emails a customer is allowed to send per time. Limiting the sending rate of illegitimate or hijacked customer accounts will reduce the likelihood that the mail server IPs will be blacklisted by other email providers. While the occasional spam that is delivered via an email provider will usually not lead to any problems, sending large amounts of spam emails in a short time frame will most likely catch the attention of blacklist maintainers. An email service can reduce that risk by upper bounding the rate of outgoing emails dynamically, based on attributes of the sending customer and the content of current and previously delivered emails. That way, normal costumers are allowed to send large amounts of legit emails without restrictions, whereas suspicious accounts are throttled in order to stabilize service for all customers. Section 1.2 introduces concepts that help in understanding how learning an optimal throttling strategy can reduce costs of email service providers.

We have identified three different threats that can potentially disrupt services provided by web hosting firms or email providers. Countermeasures against all three threats involve a classification function that maps inputs x—e.g. incoming emails—to binary labels $y \in \{+1, -1\}$—e.g. spam or non-spam. However, different challenges arise from the way data is collected, costs of misclassification are defined, and dependencies between sets of inputs and outputs are assumed from the formulation. This thesis handles each of the three applications differently by developing a specific problem formulation for each of the server security tasks. An optimization criterion is derived from each problem formulation and algorithmic solutions are presented that solve the optimization criteria. The following three sections introduce each application and corresponding paper in more depth and states my contributions on derivation, implementation, and evaluation of the investigated methods.

1.1 Inbound Spam Classification with Missing Attributes

The first application considers learning a spam classifier that decides for each incoming email if it is a spam email ($y_i = +1$) or not ($y_i = -1$). The learner is given a matrix $\mathbf{X}$ of n training instances x_i corresponding to the rows of $\mathbf{X}$ and a vector of real labels $\mathbf{y}$. Using content-based binary classification algorithms for learning to assign spam scores to incoming emails has a long and successful history, see e.g. [Blanzieri 09] for an overview. However, here the training data is compiled from several sources, including spam traps, in addition to emails collected from the service provider's own email service. Another classifier that was trained using only content features computes spam scores that are used as a feature in $\mathbf{X}$. In addition to the contents of each email, the service provider records auxiliary real-time information about the sending servers. We record the number of valid and invalid receiver addresses of all emails seen from the server so far, and the mean and standard deviation of the sizes and spam scores of all emails from the server. Such information is not available for emails from external sources but will be available when classifying unseen emails.

A classification learning algorithm has to estimate the values of missing attributes and learn a classification function based on such imputations. An imputation ω is a matrix whose values are restricted to $\omega_{il} = x_{il}$ if l-th feature is available in instance i. We develop a method that learns a distribution of imputations $p(\omega)$ and integrates over all imputations in order to compute spam scores. A Mercer kernel k allows for flexibility of the decision function. We develop an optimization problem for finding the optimal parameters of decision function $f_k(x_i; \mathbf{c}, p) = \sum_{j=1}^{n} c_j \int_\omega k(\omega_i, \omega_j) dp(\omega)$. Optimal parameters minimize the average of a loss function $\ell(y, f_k(x; \mathbf{c}, p))$ over all training instances plus a regularization terms for both parameters. ℓ is usually an approximation of the real loss function—often also called cost function—that measures the real cost for misclassifying a spam as ham and vice versa.

Parameters $\mathbf{c}$ and p have to be estimated by a learning algorithm that minimizes

$$\sum_{i=1}^{n} \ell \left(y_i, \sum_{j=1}^{n} c_j \int_\omega k(\omega_j, \omega_i) dp(\omega) \right) + \eta \sum_{i,j=1}^{n} c_i c_j \int_\omega k(\omega_j, \omega_i) dp(\omega) + \gamma Q(p) \ . \tag{1.1}$$

Regularization term $\gamma Q(p)$ may regularize the distribution towards a prior belief about how imputations might be distributed.

We show that the optimal distribution p consists of at most $n+2$ distinct imputations. Based on this theorem we develop an algorithm that greedily adds new imputations to a set of active imputations and computes a linear combination of elements of this set that minimize Eqn. 1.1.

Learning from incomplete data has been studied extensively over the years [Little 87, Shivaswamy 06, Chechik 08, Wang 10]. However, in [Dick 08] (Chapter 2) we develop a method that employs the flexibility of kernels and integrates over all possible imputations w.r.t. a learned distribution of imputations. No assumptions are made on the type of distribution. I developed and implemented the algorithm and helped in developing the proof of Theorem 1 in [Dick 08]. I also conducted the experiments.

1.2 Outbound Spam Filtering

The second application considers filtering the outbound stream of emails that is delivered by an email provider's infrastructure. The email infrastructure can be exploited to send large amounts of spam emails. Automated bots may screen hosted web sites for malconfigured scripts—e.g. ad hoc contact forms that allow for changing the recipient—and exploit those to send unwanted bulk emails to arbitrary email addresses. Alternatively, customer account passwords might be hijacked and spam senders could send emails from private accounts without knowledge of the customer. Protecting an email service against such misuse has been identified as a research topic [Goodman 04], but only very few work has been published on this specific task [Zhong 05].

When other email providers detect that the stream of incoming emails from the provider's mail servers contains an unusually high fraction of spam they might add its mail server IPs to an (internal or public) blacklist. This often results in emails delivered via the email service provider's infrastructure to be rejected by other email providers, thus rendering this service at least partly unusable. Note that the occasional spam email generally doesn't pose a high risk for being blacklisted. Based on experience of a large web hosting company that employs its own email infrastructure, we assume that the risk of being blacklisted grows in the rate of spam messages being sent and the duration over which a high sending rate is maintained. Let the stream of emails that one customer sends to the mail servers for delivery be defined as a sequence $\mathbf{x} = x_1, ..., x_n$ of n emails that request delivery at time points $\mathbf{t} = t_1, ...t_n$. $\mathbf{x}_k$ and $\mathbf{t}_k$ denote the initial sequence of the first k elements of $\mathbf{x}$ and $\mathbf{t}$, resp. An outbound email filter $\pi(\mathbf{x}_k, \mathbf{t}_k)$ decides at time t_k if email x_k will be sent or discarded, based on previous emails and decisions in that sequence. The outbound rate $r^\pi(t'|\mathbf{x}, \mathbf{t})$ describes the number of emails of $\mathbf{x}$ that were delivered over the time interval of length τ—5 minutes in our application—that ends at t'. We assume that all emails that are send in a batch $(\mathbf{x}, \mathbf{t})$ by one customer have the same label y, i.e. all emails are either spam or non-spam.

In contrast to the problem of inbound spam classification, as described in section 1.1, here we assume to learn only from labeled email batches that were recorded from the internal email infrastructure. Furthermore, as mentioned above, the loss for misclassifying one email is not independent of previous decisions. Instead, in addition to the loss $\ell(y, \pi(\mathbf{x}_k, \mathbf{t}_k))$ that is incurred for the classification of the k-th email of the batch, the loss that depends on the duration and rate that spams are delivered depends on the sequence of decisions. The rate-dependent per-time loss $\lambda(y, r^\pi)$ is integrated over the whole batch $\mathbf{x}, \mathbf{t}, y$ to determine the overall loss that a filter π incurs on this batch. The loss of π on batch $\mathbf{x}, \mathbf{t}$ with label y is

$$L(\pi; \mathbf{x}, \mathbf{t}, y) = \int_{t_1}^{t_n+\tau} \lambda(y, r^\pi(t'|\mathbf{x}, \mathbf{t}))dt' + \sum_{i=1}^{n} \ell(y, \pi(\mathbf{x}_i, \mathbf{t}_i)) \tag{1.2}$$

Learning sequences of decisions can generally be done using reinforcement learning methods [Busoniu 10], an approach we also compare against in [Dick 10]. Instead, in [Dick 10][1] (Chapter 3) we develop a specially tailored method that minimizes the expected loss $\mathbb{E}_{\mathbf{x},\mathbf{t},y}[L(\pi, \mathbf{x}, \mathbf{t}, y)]$ over all batches $\mathbf{x}, \mathbf{t}, y$ by learning a throttling mechanism. The algorithm learns a rate-limit function $f_\theta(\mathbf{x}_i, \mathbf{t}_i)$ that represents the number of emails that a customer is allowed to send in the interval $[t_i - \tau, t_i)$. The filter π is defined as

$$\pi(\mathbf{x}_i, \mathbf{t}_i) = \begin{cases} -1(\text{“allow”}) & \text{if } r^\pi(t_i|\mathbf{x}_i, \mathbf{t}_i) + 1 < f_\theta(\mathbf{x}_i, \mathbf{t}_i) \\ +1(\text{“suppress”}) & \text{otherwise} \end{cases} \tag{1.3}$$

We develop a problem formulation that assumes that the sequence of delivery time points $\mathbf{t}$ is drawn from a Poisson process. We proof that the corresponding opti-

[1]Published at Conference on Neural Information Processing Systems. https://nips.cc

mization problem is convex and show experimentally that the learned filter indeed performs better in reducing the expected loss than baseline methods.

I developed the problem formulation, the optimization problem, implemented the algorithm and performed the experiments in [Dick 10]. Peter Haider helped in writing the paper and the other authors helped in developing the proof of Theorem 1.

1.3 DDoS Attacker Detection

The third application considers the problem of learning a DDoS attacker detection mechanism. The filter has to decide which client IPs that currently access a web server are staging an attack on the server. While most of the time the web servers are not under attack, in case of an attack the detection mechanism should detect as many attackers as possible in order to reduce the costs that the attack imposes on the web hosting company. The attacker detection aims at identifying client IPs that are part of an HTTP-level attack [Ranjan 06, Liu 11]. HTTP-level attackers try to exhaust web server resources by querying them with a large number of HTTP-compliant requests that may each look inconspicuous but which may exhaust resources such as CPU time very quickly if no countermeasures are taken.

The loss each undetected attacker imposes on the system is dependent on other detection decisions that the filter outputs at the same time. If only a few attackers pose a reasonable number of requests to the servers, the server infrastructure will usually serve the additional load without any effect on the general user experience of other customers. However, if the additional requests increase the overall load on a server resource, such as CPU time, close to the available maximum capacity, user experience will degrade and ultimately web sites may become unavailable, rendering the attack successful. We assume negative user experience due to high response times or unavailability of the service to be the main cost inducing factor to the web hosting company. We therefore assume that the loss of not detecting attacking client IPs is super-linear—quadratic in our experiments—in the number of requests and used CPU time initiated by undetected attackers.

Because the joint loss c induced by all undetected attackers is not separable over individual decisions of the detection mechanism, we resort to learning a structured prediction model that decides for all current clients on a domain simultaneously (e.g. [Tsochantaridis 05]). In other words, we aim at finding the subset of clients that are attackers. Often, all attacking clients are part of the same botnet and their behavior follows a common pattern. We can take advantage of this knowledge by defining features for sets of clients that capture common patterns such as common user agents. Despite an intuitive advantage of this approach over independently classifying each client individually, the space of possible outcomes of the algorithm becomes exponential in the number of clients. Here, we would have to evaluate each subset of clients for its likelihood of containing all attackers but no legitimate clients.

Let the set of clients that access the server in a given time interval—10 seconds in our application—be denoted by $\mathbf{x} = \{x_1, ..., x_n\}$ with real labels $\mathbf{y} = \{y_1, ..., y_n\}$, where $y_i = +1$ identifies client x_i as an attacker. We learn a joint input-output function $f_\phi(\mathbf{x}, \mathbf{y}')$ that maps clients $\mathbf{x}$ and labels $\mathbf{y}'$ to a real-valued score. The decoder aims at finding the best scoring output from the set of all possible outputs

$\mathcal{Y}(\mathbf{x})$

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}' \in \mathcal{Y}(x)} f_\phi(\mathbf{x}, \mathbf{y}') \tag{1.4}$$

that minimizes the expected loss $\mathbb{E}_{\mathbf{x},\mathbf{y}}[c(\mathbf{x}, \mathbf{y}, \operatorname{argmax}_{\mathbf{y}' \in \mathcal{Y}(x)} f_\phi(\mathbf{x}, \mathbf{y}'))]$.

Time is very limited in the decoding phase, as decisions have to made in real time. Searching the complete output space $\mathcal{Y}(x)$ is impossible except in cases where only very few clients are present. Instead, we have to find a good search strategy that traverses the output space in such a way that potentially high scoring outputs are visited early during the search. In [Dick 16][2] (Chapter 4) we investigate a method that learns a search strategy π_ψ simultaneously to learning the decision function f_ϕ. In each time step T the search strategy $\pi_\psi(\mathbf{x}, \mathcal{Y}_T(\mathbf{x}))$ adds a new element to the set of visited outputs $\mathcal{Y}_T(\mathbf{x}) \subseteq \mathcal{Y}(\mathbf{x})$ and with the help of the decision function f_ϕ we can output the best scoring element $\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y}' \in \mathcal{Y}_{T+1}(x)} f_\phi(\mathbf{x}, \mathbf{y}')$. Such a decoder is called an anytime decoder as the search could be stopped at any time point [Doppa 14a]. If a distribution of available decoding time steps $p(T)$ can be estimated, we can optimize both parameters ϕ and ψ such that more focus is laid on more likely final time steps. The learned model minimizes the expected cost over all sets $\mathbf{x}$ and $\mathbf{y}$

$$\operatorname*{argmin}_{\boldsymbol{\phi},\boldsymbol{\psi}} \mathbb{E}_{(\mathbf{x},\mathbf{y}),T,Y_T(\mathbf{x})} \left[c(\mathbf{x}, \mathbf{y}, \operatorname*{argmax}_{\hat{\mathbf{y}} \in Y_T(\mathbf{x})} f_{\boldsymbol{\phi}}(\mathbf{x}, \hat{\mathbf{y}}) \right] \tag{1.5}$$

$$\text{with} \quad (\mathbf{x}, \mathbf{y}) \sim p(\mathbf{x}, \mathbf{y}), \quad T \sim p(T|\tau) \tag{1.6}$$

$$Y_T(\mathbf{x}) \sim p(Y_T(\mathbf{x})|\pi_{\boldsymbol{\psi}}, \mathbf{x}, T) \tag{1.7}$$

An algorithm is derived that approximately minimizes the expected costs and is shown to be convergent. I developed the algorithm and the proof of convergence and both implemented and evaluated the method in [Dick 16]. I gathered and labeled the data set from a large web hosting company.

[2]The final publication is available at Springer via http://dx.doi.org/10.1007/s10994-016-5581-9

Learning from Incomplete Data with Infinite Imputations

Uwe Dick DICK@MPI-SB.MPG.DE
Peter Haider HAIDER@MPI-SB.MPG.DE
Tobias Scheffer SCHEFFER@MPI-SB.MPG.DE
Max Planck Institute for Computer Science, Saarbrücken, Germany

Abstract

We address the problem of learning decision functions from training data in which some attribute values are unobserved. This problem can arise, for instance, when training data is aggregated from multiple sources, and some sources record only a subset of attributes. We derive a generic joint optimization problem in which the distribution governing the missing values is a free parameter. We show that the optimal solution concentrates the density mass on finitely many imputations, and provide a corresponding algorithm for learning from incomplete data. We report on empirical results on benchmark data, and on the email spam application that motivates our work.

1. Introduction

In many applications, one has to deal with training data with incompletely observed attributes. For instance, training data may be aggregated from different sources. If not all sources are capable of providing the same set of input attributes, the combined training sample contains incompletely observed data. This situation occurs in email spam detection, where it is helpful to augment the content of an email with real-time information about the sending server, such as its blacklist status. This information is available for all training emails that arrive at a mail server under one's own control, and it is also available at application time. But if one wants to utilize training emails from public archives, this information is missing.

We adress a learning setting in which values are *missing at random:* here, the presence or absence of values does not convey information about the class labels. If this condition is not met, it is informative to consider the presence or absence of values as additional input to the decision function. Techniques for learning from incomplete data typically involve a distributional model that imputes missing values, and the desired final predictive model. Prior work on learning from incomplete data is manifold in the literature, and may be grouped by the way the distributional model is used.

The first group models the distribution of missing values in a first step, and learns the decision function based on the distributional model in a second step. Shivaswamy et al. (2006) formulate a loss function that takes a fixed proportion of the probability mass of each instance into account, with respect to the estimated distribution of missing values. They derive second order cone programs which renders the method applicable only to very small problems. Other examples include Williams and Carin (2005), Williams et al. (2005), and Smola et al. (2005).

The second group estimates the parameters of a distributional model and the final predictive model jointly. As an example, recently Liao et al. (2007) propose an EM-algorithm for jointly estimating the imputation model and a logistic regression classifier with linear kernel, assuming the data arises from a mixture of multivariate Gaussians.

The third group makes no model assumption about the missing values, but learns the decision function based on the visible input alone. For example, Chechik et al. (2007) derive a geometrically motivated approach. For each example, the margin is re-scaled according to the visible attributes. This procedure specifically aims at learning from data with values that are *structurally missing*—as opposed to *missing at random.* Chechik et al. (2007) find empirically that the procedure is not adequate when values are missing at random.

Jointly learning a distributional model and a kernel predictive model relates to the problem of learning a

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008. Copyright 2008 by the author(s)/owner(s).

kernel function from a prescribed set of parameterized kernels. This problen drew a lot of attention recently; see, for example, Argyriou et al. (2005) and Micchelli and Pontil (2007).

Estimating the distributional model first and training the predictive model in a second step leaves the user free to choose any learning algorithm for this second step. However, a harder problem has to be solved than would be necessary. If one is only interested in a decision function that minimizes the desired loss, knowing the values or distribution of the missing attributes in the training set is not actually required. Furthermore, errors made in the imputation step and errors made in estimating the parameters of the predictive model can add up in a sequential procedure.

Consequently, we investigate learning the decision function and the distribution of imputations dependently. Unlike prior work on this topic, we develop a solution for a very general class of optimization criteria. Our solution covers a wide range of loss functions for classification and regression problems. It comes with all the usual benefits of kernel methods. We derive an optimization problem in which the distribution governing the missing values is a free parameter. The optimization problem searches for a decision function and a distribution governing the missing values which together minimize a regularized empirical risk.

No fixed parametric form of the distributional model is assumed. A regularizer that can be motivated by a distributional assumption may *bias* the distributional model *towards* a prior belief. However, the regularizer may be overruled by the data, and the resulting distributional model may be different from any parametric form. We are able to prove that there exists an optimal solution based on a distribution that is supported by finitely many imputations. This justifies a greedy algorithm for finding a solution. We derive manifestations of the general learning method and study them empirically.

The paper is structured as follows. After introducing the problem setting in Section 2, we derive an optimization problem in Section 3. Section 4 proves that there is an optimal solution that concentrates the density mass on finitely many imputations and presents an algorithm. Example instantiations of the general solution are presented in Section 5. We empirically evaluate the method in Section 6. Section 7 concludes.

2. Problem Setting

We address the problem of learning a decision function f from a training sample in which some attribute values are unobserved.

Let $\mathbf{X}$ be a matrix of n training instances $\mathbf{x}_i$ and let $\mathbf{y}$ be the vector of corresponding target values y_i. Instances and target values are drawn *iid* from an unknown distribution $p(\mathbf{x}, y)$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathcal{Y}$, where $\mathcal{Y}$ denotes the set of possible target values. Matrix $\mathbf{Z}$ indicates which features are observed. A value of $z_{il} = 1$ indicates that x_{il}, the l-th feature of the i-th example, is observed. Values are *missing at random:* y_i is conditionally independent of $\mathbf{z}_i$ given $\mathbf{x}_i$.

The goal is to learn a function $f : \mathbf{x} \mapsto y$ that predicts target values for *completely observed* examples. The decision function should incur only a minimal true risk $R(f) = \int L(y, f(\mathbf{x}))p(\mathbf{x}, y)d\mathbf{x}dy$, where L is a loss function for the task at hand.

As a means to minimizing the true risk, we seek a function f in the *reproducing kernel Hilbert space* $\mathcal{H}_k$ induced by a kernel k that minimizes a regularized empirical risk functional $R(f) = \sum_{i=1}^n l(y_i, f(\mathbf{x}_i)) + \eta\|f\|_k^2$. We demand k to be a Mercer kernel. Loss function l approximates the true loss L. The *representer theorem* allows us to write the minimizer as a sum over functions in $\mathcal{H}_k$ centered at training instances: $f(\mathbf{x}) = \sum_{j=1}^n c_j k(\mathbf{x}_j, \mathbf{x})$.

The learning problem from completely observed data would amount to solving Optimization Problem 1.

Optimization Problem 1 (Primal learning problem, observed data). *Over* $\mathbf{c}$, *minimize*

$$R(\mathbf{c}, k) = \sum_{i=1}^n l\Big(y_i, \sum_{j=1}^n c_j k(\mathbf{x}_j, \mathbf{x}_i)\Big) + \eta \sum_{i,j=1}^n c_i c_j k(\mathbf{x}_j, \mathbf{x}_i)$$

We require that the loss function be defined in such a way that Optimization Problem 1 can be written in the dual form of Optimization Problem 2. A wide range of loss functions satisfies this demand; we will later see that this includes hinge loss and squared loss.

Optimization Problem 2 (Dual of learning problem). *Given* $a < 0$, *over* $\mathbf{c}$, *maximize*

$$a \langle \mathbf{c}, \mathbf{K}\mathbf{c} \rangle - R^*(\mathbf{c})$$

subject to the constraints

$$\forall_{i=1}^{m_1^*} g_i^*(\mathbf{c}) \leq 0, \qquad \forall_{j=1}^{m_2^*} h_j^*(\mathbf{c}) = 0. \qquad (1)$$

$R^*(\mathbf{c})$ denotes a differentiable convex function of the dual variables $\mathbf{c}$ which we demand to be independent of the kernel matrix $\mathbf{K}$. The inequality constraints g_i^* are differentiable convex and the equality constraints h_j^* differentiable affine. We like to note that the requirement of independence between R^* and $\mathbf{K}$ is not very restrictive in practice, as we will see in chapter 5. Furthermore, we demand strong duality to hold between Optimization problems 1 and 2.

3. Learning from Incomplete Data in One Step

If any instance $\mathbf{x}_i$ has unobserved features, then $k(\mathbf{x}_i, \mathbf{x})$ and, consequently, the decision function f are not properly defined. In order to learn from incomplete data, we will marginalize the decision function and risk functional by the observable attributes and integrate over all unobserved quantities. To this end, we define $\boldsymbol{\omega} \in \Omega_{\mathbf{X}}^{\mathbf{Z}} \subset \mathbb{R}^{n\times d}$ as a matrix of imputations constrained by $\omega_{il} = x_{il}$ if $z_{il} = 1$. We demand $\Omega_{\mathbf{X}}^{\mathbf{Z}}$ to be compact for the rest of this paper. Let $\boldsymbol{\omega}_i$ denote the i-th row of $\boldsymbol{\omega}$. Then we can define a family of kernels $K(\boldsymbol{\omega})(\mathbf{x}_j, \mathbf{x}_i) = k(\boldsymbol{\omega}_j, \boldsymbol{\omega}_i)$. Any probability measure $p(\boldsymbol{\omega})$ on imputations induces a marginalization of the kernel by the observable variables. Equation 2 integrates over all imputations of unobserved values; it can be evaluated based on the observed values.

$$K(p)(\mathbf{x}_j, \mathbf{x}_i) = \int_{\boldsymbol{\omega}\in\Omega_{\mathbf{X}}^{\mathbf{Z}}} k(\boldsymbol{\omega}_j, \boldsymbol{\omega}_i) dp(\boldsymbol{\omega}) \tag{2}$$

Any probability measure $p(\boldsymbol{\omega})$ constitutes an optimization criterion $R(\mathbf{c}, K(p))$. In the absence of knowledge about the true distribution of missing values, $p(\boldsymbol{\omega})$ becomes a free parameter. Note that $p(\boldsymbol{\omega})$ is a continuous probability measure that is not constrained to any particular parametric form; the space of parameters is therefore of infinite dimensionality.

It is natural to add a regularizer $Q(p)$ that reflects prior belief on the distribution of imputations $p(\boldsymbol{\omega})$ to the optimization criterion, in addition to the empirical risk and regularizer on the predictive model. The regularizer is assumed to be continuous in p. The regularizer does not *constrain* $p(\boldsymbol{\omega})$ to any specific class of distribution, but it reflects that some distributions are believed to be more likely. Without a regularizer, the criterion can often be minimized by imputations which move instances with missing values far away from the separator, thereby removing their influence on the outcome of the learning process. This leads to Optimization Problem 3.

Optimization Problem 3 (Learning problem with infinite imputations). *Given n training examples with incomplete feature values, $\gamma > 0$, kernel function k, over all $\mathbf{c}$ and p, minimize*

$$\tilde{R}_{k,\gamma}(\mathbf{c}, p) \quad = \quad R(\mathbf{c}, K(p)) + \gamma Q(p) \tag{3}$$

subject to the constraints

$$\forall \boldsymbol{\omega} : \ p(\boldsymbol{\omega}) \geq 0, \qquad \textstyle\int_{\boldsymbol{\omega}\in\Omega_{\mathbf{X}}^{\mathbf{Z}}} p(\boldsymbol{\omega}) d\boldsymbol{\omega} = 1.$$

Each solution to Optimization Problem 3 integrates over *infinitely* many different imputations. The search space contains all continuous probability measures on imputations, the search is guided by the regularizer Q. The regularization parameter γ determines the influence of the regularization on the resulting distribution. For $\gamma \to \infty$ the solution of the optimization reduces to the solution obtained by first estimating the distribution of missing attribute values that minimizes the regularizer. For $\gamma \to 0$ the solution is constituted by the distribution minimizing the risk functional R.

4. Solving the Optimization Problem

In this section, we devise a method for efficiently finding a solution to Optimization Problem 3. Firstly, we show that there exists an optimal solution $\hat{\mathbf{c}}, \hat{p}$ with $\hat{p}$ supported on at most $n+2$ imputations $\boldsymbol{\omega} \in \Omega_{\mathbf{X}}^{\mathbf{Z}}$. Secondly, we present an algorithm that iteratively finds the optimal imputations and parameters minimizing the regularized empirical risk.

4.1. Optimal Solution with Finite Combination

In addition to the parameters $\mathbf{c}$ of the predictive models, continuous probability measure $p(\boldsymbol{\omega})$ contributes an infinite set of parameters to Optimization Problem 3. The implementation of imputations as parameters of a kernel family allows us to show that there exists an optimal probability measure $\hat{p}$ for Equation 3 such that $\hat{p}$ consists of finitely many different imputations.

Theorem 1. *Optimization Problem 3 has an optimal solution $\hat{\mathbf{c}}, \hat{p}$ in which $\hat{p}$ is supported by at most $n+2$ imputations $\boldsymbol{\omega} \in \Omega_{\mathbf{X}}^{\mathbf{Z}}$.*

Proof. The compactness of $\Omega_{\mathbf{X}}^{\mathbf{Z}}$ and the continuity of $\mathbf{K}$ immediately imply that *there exists some* solution to Optimization Problem 3. It remains to be shown that at least one of the solutions is supported by at most $n+2$ imputations. Let $\bar{\mathbf{c}}, \bar{p}$ be *any* solution and let all requirements of the previous section hold. The idea of this proof is to construct a correspondence between distributions over imputations and vectors in $\mathbb{R}^{n+1}$, where a finite support set is known to exist. Define $\mathbf{S}(\boldsymbol{\omega}) = \mathbf{K}(\boldsymbol{\omega})\bar{c} \in \mathbb{R}^n$ and $D = \{(\mathbf{S}(\boldsymbol{\omega})^\top, Q(\boldsymbol{\omega}))^\top : \boldsymbol{\omega} \in \Omega_{\mathbf{X}}^{\mathbf{Z}}\} \subset \mathbb{R}^{n+1}$. Since $\Omega_{\mathbf{X}}^{\mathbf{Z}}$ is compact and $K(\cdot)$ and $Q(\cdot)$ are continuous by definition, D is compact as well. We define a measure over D as $\mu(A \times B) = \bar{p}(\{\boldsymbol{\omega} : \mathbf{S}(\boldsymbol{\omega}) \in A \wedge Q(\boldsymbol{\omega}) \in B\})$.

Then, by Carathéodory's convex hull theorem, there exists a set of k vectors $\{(\mathbf{s}_1^\top, q_1)^\top, \ldots, (\mathbf{s}_k^\top, q_k)^\top\} \subseteq D$ with $k \leq n+2$ and nonnegative constants ν_i with

$\sum_{i=1}^{k} \nu_i = 1$, such that

$$\int_D (\mathbf{s}^\top, q)^\top d\mu((\mathbf{s}^\top, q)^\top) = \sum_{i=1}^{k} (\mathbf{s}_i^\top, q_i)^\top \nu_i.$$

For each i, select any $\boldsymbol{\omega}_i$ such that $(\mathbf{S}(\boldsymbol{\omega}_i)^\top, Q(\boldsymbol{\omega}_i)) = (\mathbf{s}_i^\top, q_i)$. We construct $\hat{p}$ by setting $\hat{p}(\boldsymbol{\omega}) = \sum_{i=1}^{k} \nu_i \delta_{\boldsymbol{\omega}_i}$, where $\delta_{\boldsymbol{\omega}_i}$ denotes the Dirac measure at $\boldsymbol{\omega}_i$. The optimal $\hat{\mathbf{c}}$ results as $\arg\min_{\mathbf{c}} R(\mathbf{c}, K(\hat{p}))$. We have

$$\begin{aligned} \int_D \mathbf{s} d\mu((\mathbf{s}^\top, q)^\top) &= \sum_{i=1}^{k} \mathbf{s}_i \nu_i, \quad \text{and} \\ \int_D q d\mu((\mathbf{s}^\top, q)^\top) &= \sum_{i=1}^{k} q_i \nu_i. \end{aligned}$$

Then

$$\begin{aligned} \mathbf{K}(\bar{p})\bar{\mathbf{c}} &= \left(\int_{\Omega_{\mathbf{X}}^{\mathbf{Z}}} \mathbf{K}(\boldsymbol{\omega}) d\bar{p}(\boldsymbol{\omega})\right) \bar{\mathbf{c}} = \int_{\Omega_{\mathbf{X}}^{\mathbf{Z}}} \mathbf{S}(\boldsymbol{\omega}) d\bar{p}(\boldsymbol{\omega}) \\ &= \int_D \mathbf{S}(\boldsymbol{\omega}) d\mu \left((\mathbf{S}(\boldsymbol{\omega})^\top, Q(\boldsymbol{\omega}))^\top\right) \\ &= \sum_{i=1}^{k} \mathbf{s}_i \nu_i = \int_{\Omega_{\mathbf{X}}^{\mathbf{Z}}} \mathbf{S}(\boldsymbol{\omega}) d\hat{p}(\boldsymbol{\omega}) \\ &= \int_{\Omega_{\mathbf{X}}^{\mathbf{Z}}} \mathbf{K}(\boldsymbol{\omega}) d\hat{p}(\boldsymbol{\omega}) \bar{\mathbf{c}} = \mathbf{K}(\hat{p})\bar{\mathbf{c}}. \end{aligned}$$

Likewise,

$$\begin{aligned} Q(\bar{p}) &= \int_D Q(\boldsymbol{\omega}) d\mu \left((\mathbf{S}(\boldsymbol{\omega})^\top, Q(\boldsymbol{\omega}))^\top\right) \\ &= \sum_{i=1}^{k} q_i \nu_i = Q(\hat{p}). \end{aligned}$$

Since $Q(p)$ does not depend on $\mathbf{c}$, $\bar{\mathbf{c}} = \arg\min_{\mathbf{c}} R(\mathbf{c}, \mathbf{K}(\bar{p}))$, and by strong duality, $\bar{\mathbf{c}} = \arg\max_{\mathbf{c}} a \langle \mathbf{c}, \mathbf{K}(\bar{p})\mathbf{c}\rangle - R^*(\mathbf{c})$. This implies that the Karush-Kuhn-Tucker conditions hold for $\bar{\mathbf{c}}$, namely there exist constants $\kappa_i \geq 0$ and λ_j such that

$$a\mathbf{K}(\bar{p})\bar{\mathbf{c}} - \nabla R^*(\bar{\mathbf{c}}) + \sum_i \kappa_i \nabla g_i^*(\bar{\mathbf{c}}) + \sum_j \lambda_j \nabla h_j^*(\bar{\mathbf{c}}) = 0$$

$$\forall_i \; g_i^*(\bar{\mathbf{c}}) \leq 0, \quad \forall_j \; h_i^*(\bar{\mathbf{c}}) = 0, \quad \forall_i \; \kappa_i g_i^*(\bar{\mathbf{c}}) = 0$$

It is easy to see that therefore $\bar{\mathbf{c}}$ is also a maximizer of $a \langle \mathbf{c}, \mathbf{K}(\hat{p})\mathbf{c}\rangle - R^*(\mathbf{c})$, because $\mathbf{K}(\bar{p})\bar{\mathbf{c}} = \mathbf{K}(\hat{p})\bar{\mathbf{c}}$ and the Karush-Kuhn-Tucker conditions still hold. Their sufficiency follows from the fact that $\mathbf{K}(p)$ is positive semi-definite for any p, and the convexity and affinity premises. Thus,

$$\begin{aligned} & R(\bar{\mathbf{c}}, K(\bar{p})) + \gamma Q(\bar{p}) \\ &= \left[\min_{\mathbf{c}} R(\mathbf{c}, K(\bar{p}))\right] + \gamma Q(\bar{p}) \\ &= \left[\max_{\mathbf{c}} a \langle \mathbf{c}, \mathbf{K}(\bar{p})\mathbf{c}\rangle - R^*(\mathbf{c})\right] + \gamma Q(\bar{p}) \\ &= \left[a \langle \bar{\mathbf{c}}, \mathbf{K}(\bar{p})\bar{\mathbf{c}}\rangle - R^*(\bar{\mathbf{c}})\right] + \gamma Q(\bar{p}) \\ &= \left[a \langle \bar{\mathbf{c}}, \mathbf{K}(\hat{p})\bar{\mathbf{c}}\rangle - R^*(\bar{\mathbf{c}})\right] + \gamma Q(\hat{p}) \\ &= \left[\max_{\mathbf{c}} a \langle \mathbf{c}, \mathbf{K}(\hat{p})\mathbf{c}\rangle - R^*(\mathbf{c})\right] + \gamma Q(\hat{p}) \\ &= \left[\min_{\mathbf{c}} R(\mathbf{c}, K(\hat{p}))\right] + \gamma Q(\hat{p}) \\ &= R(\hat{\mathbf{c}}, K(\hat{p})) + \gamma Q(\hat{p}). \end{aligned}$$

We have now established that there exists a solution with at most $n + 2$ imputations. □

4.2. Iterative Optimization Algorithm

This result justifies the following greedy algorithm to find an optimal solution to Optimization Problem 3. The algorithm works by iteratively optimizing Problem 1 (or, equivalently, 2), and updating the distribution over the missing attribute values. Let $p_{\bar{\boldsymbol{\omega}}}$ denote the distribution $p(\boldsymbol{\omega}) = \delta_{\bar{\boldsymbol{\omega}}}$. Algorithm 1 shows the steps.

Algorithm 1 Compute optimal distribution of imputations on $\Omega_{\mathbf{X}}^{\mathbf{Z}}$

```
Initialization: Choose $p^{(1)} = p_{\boldsymbol{\omega}^{(1)}}$; e.g., $\boldsymbol{\omega}_{il}^{(1)} = 0$ for all $z_{il} \neq 1$
for $t = 1 \ldots$ do
    1. $\hat{\mathbf{c}} \leftarrow \arg\min_{\mathbf{c}} R(\mathbf{c}, K(p^{(t)}))$
    2. Find $\boldsymbol{\omega}^{(t+1)} \in \Omega_{\mathbf{X}}^{\mathbf{Z}} : \tilde{R}_{k,\gamma}(\hat{\mathbf{c}}, p_{\boldsymbol{\omega}^{(t+1)}}) < \tilde{R}_{k,\gamma}(\hat{\mathbf{c}}, p^{(t)})$. If no such $\boldsymbol{\omega}^{(t+1)}$ exists, terminate.
    3. $\beta_t \leftarrow \arg\min_{\beta \in (0,1]} \left[\min_{\mathbf{c}} \tilde{R}_{k,\gamma}(\mathbf{c}, \beta p_{\boldsymbol{\omega}^{(t+1)}} + (1-\beta)p^{(t)})\right]$
    4. $p^{(t+1)} \leftarrow \beta_t p_{\boldsymbol{\omega}^{(t+1)}} + (1-\beta_t)p^{(t)}$
    5. $\forall j < t : \beta_j \leftarrow \beta_j(1-\beta_t)$
end for
```

Step 1 consists of minimizing the regularized empirical risk functional R, given the current distribution. In step 2 a new imputation is constructed which improves on the current objective value. Since in general $\tilde{R}_{k,\gamma}(\mathbf{c}, p_{\boldsymbol{\omega}})$ is not convex in $\boldsymbol{\omega}$, one cannot find the *optimal* $\boldsymbol{\omega}$ efficiently. But the algorithm only requires to find *any* better $\boldsymbol{\omega}$. Thus it is reasonable to perform gradient ascent on $\boldsymbol{\omega}$, with random restarts in case the found local optimum does not satisfy the inequality of step 2. In step 3 and 4 the optimal distribution consisting of the weighted sum of currently used Dirac impulses $\sum_{i=1}^{t} \beta_i \delta_{\boldsymbol{\omega}_i}$ and the new imputation $\delta_{\boldsymbol{\omega}^{(t+1)}}$ is computed. This step is convex in β if

$\tilde{R}_{k,\gamma}(\mathbf{c}, \beta p_{\boldsymbol{\omega}^{(t+1)}} + (1-\beta)p^{(t)})$ is linear in β. By looking at Optimization Problem 2, we see that this is the case for R. Thus the convexity depends on the choice for Q (see Sect. 5.2). Step 5 updates the weights of the previous imputations.

The algorithm finds t imputations $\boldsymbol{\omega}^{(j)}$ and their weights β_j, as well as the optimal example coefficients $\mathbf{c}$. We can construct the classification function f as

$$f(\mathbf{x}) \;=\; \sum_{j=1}^{t}\sum_{i=1}^{n} \beta_j \mathbf{c}_i k(\boldsymbol{\omega}_i^{(j)}, \mathbf{x}). \qquad (4)$$

Note that the value $n+2$ is an upper bound for the number of basic kernels which constitute the optimal solution. The algorithm is not guaranteed to terminate after $n+2$ iterations, because the calculated imputations are not necessarily optimal. In practice, however, the number of iterations is usually much lower. In our experiments, the objective value of the optimization problem converges in less than 50 iterations.

5. Example Learners

In this chapter we present manifestations of the generic method, which we call *weighted infinite imputations*, for learning from incomplete data that we use in the experimental evaluation.

Recall from Section 3 the goal to learn a decision function f from incomplete data that minimizes the expected risk $R(f) = \int L(y, f(\mathbf{x}))p(\mathbf{x}, y)d\mathbf{x}dy$. In classification problems the natural loss function L becomes the *zero-one loss*, whereas in regression problems the loss depends on the specific application; common choices are the *squared error* or the *ϵ-insensitive* loss. The considerations in the previous chapters show that, in order to learn regression or classification functions from training instances with missing attribute values, we only have to specify the dual formulation of the preferred learning algorithm on complete data and a regularizer on the distribution of imputations p.

5.1. Two Standard Learning Algorithms

For binary classification problems, we choose to approximate the zero-one by the *hinge* loss and perform *support vector machine* learning. The dual formulation of the SVM is given by $R^{SVM}(\mathbf{c}, k) = \sum_{i=1}^{n} \frac{c_i}{y_i} - \frac{1}{2}\sum_{i,j=1}^{n} c_i c_j k(\mathbf{x}_j, \mathbf{x}_i)$ subject to the constraints $0 \leq \frac{c_i}{y_i} \leq \frac{1}{\eta}$ and $\sum_{i=1}^{n} c_i = 0$. We see that the demands of Optimization Problem 2 are met and a finite solution can be found. Taking the SVM formulation as the dual Optimization Problem 2 gives us the means – in conjunction with an appropriate regularizer Q – to learn a classification function f from incomplete data.

For regression problems, the loss depends on the task at hand, as noted above. We focus on penalizing the *squared error*, though we like to mention that the approach works for other losses likewise. One widely used learning algorithm for solving the problem is *kernel ridge regression.* Again, we can learn the regression function f from incomplete data by using the same principles as described above. Kernel ridge regression minimizes the regularized empirical risk $\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2 + \eta\|f\|^2$. The dual formulation $R^{KRR}(\mathbf{c}, k) = \sum_{i=1}^{n} c_i y_i - \frac{1}{4}\sum_{i=1}^{n} c_i^2 + \frac{1}{4\eta}\sum_{i,j=1}^{n} c_i c_j k(x_i, x_j)$ again meets the demands of the dual optimization problem 2. Substituting its primal formulation for R in step 1 of Algorithm 1 and in Eqn. 3 solves the problem of learning the regression function from incomplete data after specifying a regularizer Q.

5.2. Regularizing towards Prior Belief in Feature Space

A regularizer on the distribution of missing values can guide the search towards distributions $\hat{\boldsymbol{\omega}}$ that we believe to be likely. We introduce a regularization term which penalizes imputations that are different from our prior belief $\hat{\boldsymbol{\omega}}$. We choose to penalize the sum of squared distances between instances $\mathbf{x}_i$ and $\hat{\boldsymbol{\omega}}_i$ in *feature space* $\mathcal{H}_k$ induced by kernel k. We define the squared distance regularization term Q^{sq} as

$$\begin{aligned} Q^{sq}(k, \hat{\boldsymbol{\omega}}) \;&=\; \sum_{i=1}^{n} \|\phi_k(\mathbf{x}_i) - \phi_k(\hat{\boldsymbol{\omega}}_i)\|_2^2 \\ &=\; \sum_{i=1}^{n} k(\mathbf{x}_i, \mathbf{x}_i) - 2k(\mathbf{x}_i, \hat{\boldsymbol{\omega}}_i) + k(\hat{\boldsymbol{\omega}}_i, \hat{\boldsymbol{\omega}}_i). \end{aligned}$$

Note that when using Q^{sq}, step 3 of Algorithm 1 becomes a convex minimization procedure.

5.3. Imputing the Mean in Feature Space

In principle any imputation we believe is useful for learning a good classifier can be used as $\hat{\boldsymbol{\omega}}$. Several models of the data can be assumed to compute corresponding optimal imputations. We like to mention one interesting model, namely the class-based mean imputation in the *feature space* $\mathcal{H}_k$ induced by kernel k. This model imputes missing values such that the sum of squared distances between completed instances to the class-dependent mean in feature space is minimal over all possible imputations. $\hat{\boldsymbol{\omega}} = \arg\min_{\boldsymbol{\omega}} \sum_{i=1}^{n} \|\phi_k(\boldsymbol{\omega}_i) - \frac{1}{n_{y_i}}\sum_{j:y_j=y_i} \phi_k(\boldsymbol{\omega}_j)\|_2^2$, where n_y denotes the number of instances with label y. Simple algebraic manipulations show that this is equivalent to minimizing the sum of squared distances between all instances $\sum_{v\in\{-1,1\}} \frac{1}{n_v} \sum_{i,j:y_i=y_j=v} \|\phi_k(\boldsymbol{\omega}_i) - \phi_k(\boldsymbol{\omega}_j)\|_2^2 = \sum_{v\in\{-1,1\}} \frac{1}{n_v} \sum_{i,j:y_i=y_j=v} \big[k(\boldsymbol{\omega}_i, \boldsymbol{\omega}_i) - 2k(\boldsymbol{\omega}_i, \boldsymbol{\omega}_j) + k(\boldsymbol{\omega}_j, \boldsymbol{\omega}_j)\big]$

Definition 1 (Mean in Feature Space). *The class-based* mean in feature space *imputation method imputes missing values $\hat{\boldsymbol{\omega}}$ which optimize*

$$\hat{\boldsymbol{\omega}} = \arg\min_{\boldsymbol{\omega}} \sum_{v\in\{-1,+1\}} \frac{1}{n_v} \sum_{i,j:y_i=y_j=v} \big[k(\boldsymbol{\omega}_i, \boldsymbol{\omega}_i) - 2k(\boldsymbol{\omega}_i, \boldsymbol{\omega}_j) + k(\boldsymbol{\omega}_j, \boldsymbol{\omega}_j)\big]$$

Note that this model reduces to the standard mean in input space when using the linear kernel.

6. Empirical Evaluation

We evaluate the performance of our generic approach *weighted infinite imputations* for two example realizations. We test for classification performance on the email spam data set which motivates our investigation. Furthermore, we test on seven additional binary classification problems and three regression problems.

6.1. Classification

We choose to learn the decision function for the binary classification task by substituting the risk functional of the *support vector machine*, $-R^{SVM}$, as presented in section 5.1 for R and the squared distance regularizer Q^{sq} (Section 5.2) for Q in Optimization Problem 3.

For the motivating problem setting, we assemble a data set of 2509 spam and non-spam emails, which are preprocessed by a linear text classifier which is currently in use at a large webspace hosting company. This classifier discriminates reasonably well between spam and non-spam, but there is still a small fraction of misclassified emails. The classifier has been trained on about 1 million emails from a variety of sources, including spam-traps as well as emails from the hosting company itself, recognizing more than 10 million distinct text features. On this scale, training a support vector machine with Gaussian kernel is impractical, therefore we employ a two-step procedure. We discard the contents of the emails and retain only their spam score from the text classifier and their size in bytes as content features in the second-step classifier. At the time of collection of the emails, we record auxiliary real-time information about the sending servers. This includes the number of valid and invalid receiver addresses of all emails seen from the server so far, and the mean and standard deviation of the sizes and spam scores of all emails from the server. Such information is not available for emails from external sources, but will be available when classifying unseen emails. We randomly draw 1259 emails, both spam and non-spam, with server information, whereas half of those were drawn from a set of misclassified spam-emails. We augment this set with 1250 emails drawn randomly from a source without server information for which only 2 of the 8 attributes are observed.

To evaluate the common odd versus even digits discrimination, random subsets of 1000 training examples from the USPS handwritten digit recognition set are used. We test on the remaining 6291 examples. Additionally, we test on KDD Cup 2004 Physics (1000 train, 5179 test, 78 attributes) data set and on the 4-view land mine detection data (500, 213, 41) as used by Williams and Carin (2005). In the latter, instances consist of 4 views on the data, each from a separate sensor. Consequently, we randomly select complete views as missing. From the UCI machine learning repository we take the Breast (277 instances, 9 features), Diabetes (768, 8), German (1000, 20), and Waveform (5000, 21) data sets. Selection criteria for this subset of the repository were minimum requirements on sample size and number of attributes.

On each data set we test the performance of *weighted infinite imputation* using four different regularization imputations $\hat{\boldsymbol{\omega}}$ for the regularizer $Q^{sq}(K(p), \hat{\boldsymbol{\omega}})$. These imputations are computed by *mean imputation in input space* (**MeanInput**) and *mean imputation in feature space* (**MeanFeat**) as by Definition 1. Additionally we use the *EM* algorithm to compute the attributes imputed by the maximum likelihood parameters of an assumed multivariate Gaussian distribution with no restrictions on the covariate matrix (**Gauss**), and a Gaussian Mixture Model with 10 Gauss centers and spherical covariances (**GMM**).

Four learning procedures based on single imputations serve as reference methods: the **MeanInput**, **MeanFeat**, **Gauss**, and **GMM** reference methods first determine a single imputation, and then invoke the learning algorithm.

All experiments use a spheric Gaussian kernel. Its variance parameter σ as well as the SVM-parameter η are adjusted using the regular SVM with a training and test split on fully observed data. All experiments on the same data set use this resulting parameter setting. Results are averaged over 100 runs were in each run training and test split as well as missing attributes are chosen randomly. If not stated otherwise, 85% of attributes are marked missing on all data sets. In order to evaluate our method on the email data set, we perform 20-fold cross-validation. Since the emails with

Table 1. Classification accuracies and standard errors for all data sets. Higher accuracy values are written in bold face, "$*$" denotes significant classification improvement.

		MeanInput	Gauss	GMM	MeanFeat
Email	Single imp	0.9571 ± 0.0022	0.9412 ± 0.0037	0.9505 ± 0.0030	0.9570 ± 0.0022
	WII	0.9571 ± 0.0022	$\mathbf{0.9536 \pm 0.0022}$ $*$	$\mathbf{0.9527 \pm 0.0024}$	$\mathbf{0.9600 \pm 0.0019}$ $*$
USPS	Single imp	0.8581 ± 0.0027	0.8688 ± 0.0022	0.9063 ± 0.0012	0.8581 ± 0.0027
	WII	$\mathbf{0.8641 \pm 0.0027}$ $*$	$\mathbf{0.8824 \pm 0.0024}$ $*$	$\mathbf{0.9105 \pm 0.0015}$ $*$	$\mathbf{0.8687 \pm 0.0027}$ $*$
Physics	Single imp	0.6957 ± 0.0035	0.5575 ± 0.0038	0.6137 ± 0.0050	0.6935 ± 0.0028
	WII	$\mathbf{0.7084 \pm 0.0039}$ $*$	$\mathbf{0.6543 \pm 0.0055}$ $*$	$\mathbf{0.6881 \pm 0.0049}$ $*$	$\mathbf{0.7036 \pm 0.0032}$ $*$
Mine	Single imp	0.8650 ± 0.0025	0.8887 ± 0.0023	0.8916 ± 0.0023	0.8660 ± 0.0026
	WII	$\mathbf{0.8833 \pm 0.0026}$ $*$	$\mathbf{0.8921 \pm 0.0021}$	$\mathbf{0.8946 \pm 0.0022}$ $*$	$\mathbf{0.8844 \pm 0.0026}$ $*$
Breast	Single imp	0.7170 ± 0.0055	0.7200 ± 0.0048	0.7164 ± 0.0048	0.7085 ± 0.0057
	WII	$\mathbf{0.7184 \pm 0.0056}$	$\mathbf{0.7243 \pm 0.0048}$ $*$	$\mathbf{0.7212 \pm 0.0050}$ $*$	$\mathbf{0.7152 \pm 0.0057}$ $*$
Diabetes	Single imp	0.7448 ± 0.0025	0.7053 ± 0.0036	0.7154 ± 0.0043	0.7438 ± 0.0026
	WII	$\mathbf{0.7455 \pm 0.0025}$	$\mathbf{0.7234 \pm 0.0036}$ $*$	$\mathbf{0.7389 \pm 0.0031}$ $*$	$\mathbf{0.7439 \pm 0.0024}$
German	Single imp	0.7331 ± 0.0029	0.7058 ± 0.0029	0.7056 ± 0.0028	$\mathbf{0.7364 \pm 0.0029}$
	WII	$\mathbf{0.7368 \pm 0.0025}$ $*$	$\mathbf{0.7118 \pm 0.0030}$ $*$	$\mathbf{0.7120 \pm 0.0028}$ $*$	0.7357 ± 0.0027
Waveform	Single imp	0.8700 ± 0.0019	0.8241 ± 0.0031	0.7827 ± 0.0049	0.8679 ± 0.0020
	WII	0.8700 ± 0.0019	$\mathbf{0.8612 \pm 0.0019}$ $*$	$\mathbf{0.8583 \pm 0.0020}$ $*$	$\mathbf{0.8686 \pm 0.0020}$ $*$

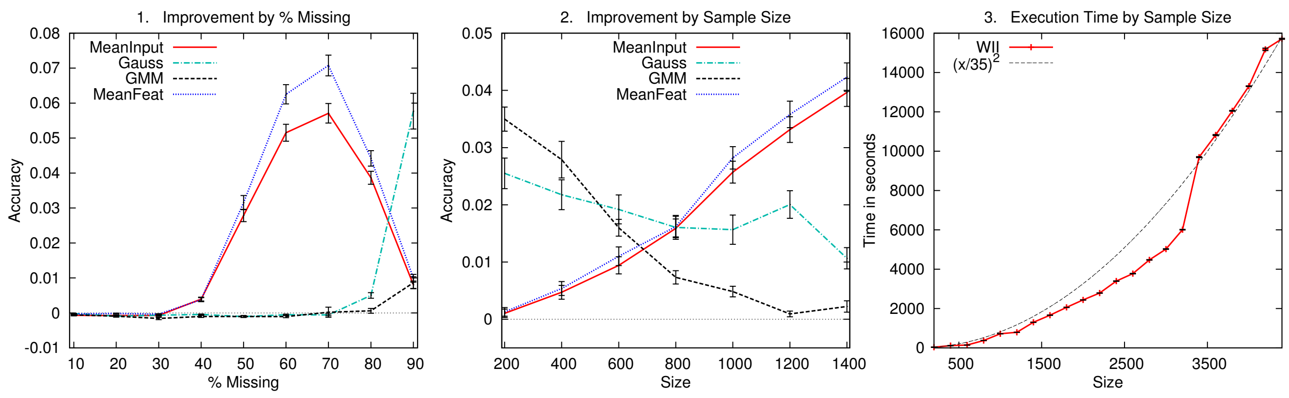


Figure 1. Detailed results on USPS classification task.

missing attributes cannot be used as test examples, the test sets are only taken from the fully observed part of the data set.

Table 6.1 shows accuracies and standard errors for the *weighted infinite imputations* (WII) method with squared distance regularization compared to all single imputations $\hat{\omega}$ on each data set. Regularization parameter γ is automatically chosen for each run based on the performance on a separate tuning set. Baselines are obtained by first imputing $\hat{\omega}$ and learning the classifier in a second step. The *weighted infinite imputations* method outperforms the single imputation in virtually all settings. We test for significant improvements with a paired t-test on the 5% significance level. Significant improvements are marked with a "$*$" in the table.

We explore the dependence of classification performance on training sample size and the percentage of missing attribute values in more detail. The first graph in Figure 1 shows improvements in classification accuracy of our method over the single imputations depending on the percentage of missing values. Graph 2 shows classification accuracy improvements depending on the size of the labeled training set. Both experiments are performed on USPS data set and we again adjust γ separately for each run based on the performance on the tuning set. We note that similar results are obtained for the other classification problems. The *weighted infinite imputation* method can improve classification accuracy even when only 30% of the attribute values are missing. It shows, though, that it works best if at least 60% are missing, depending on $\hat{\omega}$. On the other hand, we see that it works for all training set sizes, again depending on $\hat{\omega}$. Similar results are obtained for the other data sets.

Table 2. Mean squared error results and standard errors for regression data sets. Smaller mean squared errors are written in bold face, "$*$" denotes significant improvement.

		MeanInput	Gauss	GMM	MeanFeat
Housing	Single imp	193.0908 ± 19.9408	288.6192 ± 41.5954	160.4940 ± 16.2004	1134.5635 ± 101.9452
	WII	$\mathbf{66.5144 \pm 0.8958} *$	$\mathbf{62.3073 \pm 0.8479} *$	$\mathbf{66.7959 \pm 0.9173} *$	$\mathbf{64.7926 \pm 0.9619} *$
Ailerons	Single imp	81.7671 ± 4.5862	172.5037 ± 8.6705	79.8924 ± 4.0297	193.5790 ± 10.4899
	WII	$\mathbf{11.8034 \pm 0.1494} *$	$\mathbf{8.7505 \pm 0.0932} *$	$\mathbf{11.7595 \pm 0.1530} *$	$\mathbf{11.8220 \pm 0.1387} *$
Cpu_act	Single imp	10454.176 ± 962.598	15000.380 ± 973.100	10123.172 ± 933.143	15710.812 ± 1099.603
	WII	$\mathbf{306.257 \pm 12.500} *$	$\mathbf{204.180 \pm 5.058} *$	$\mathbf{305.651 \pm 13.627} *$	$\mathbf{247.988 \pm 8.010} *$

To evaluate the convergence of our method, we measure classification accuracy after each iteration of the learning algorithm. It shows that classification accuracy does not change significantly after about 5 iterations for a typical γ, in this case $\gamma = 10^5$ for the USPS data set. On average the algorithm terminates after about 30-40 iterations. The computational demands of the *weighted infinite imputation* method are approximately quadratic in the training set size for the classification task, as can be seen in Graph 3 of Figure 1. This result depends on the specific risk functional R and its optimization implementation. Nevertheless, it shows that risk functionals which are solvable in quadratic time do not change their computational complexity class when learned with incomplete data.

6.2. Regression

We evaluate the *weighted infinite imputations* method on regression problems using the squared error as loss function. Consequently, risk functional R^{KRR} (Sect. 5.1) is used as R and again the squared distance regularizer Q^{sq} for Q in Optimization Problem 3. From UCI we take the Housing data (506, 14), and from the Weka homepage cpu_act (1500, 21) and ailerons (2000, 40). Ridge parameter η and RBF-kernel parameter σ were again chosen such that they lead to best results on the completely observed data. Regularization parameter γ was chosen based on the performance on a tuning set consisting of 150 examples. Results are shown in Table 2. We can see that our method outperforms the results obtained with the single imputations significantly for all settings.

7. Conclusion

We devised an optimization problem for learning decision functions from incomplete data, where the distribution p of the missing attribute values is a free parameter. The investigated method makes only minor assumptions on the distribution by the means of a regularizer on p that can be chosen freely. By simultaneously optimizing the function and the distribution of imputations, their dependency is taken into account properly. We presented a proof that the optimal solution for the joint learning problem concentrates the density mass of the distribution on finitely many imputations. This justifies the presented iterative algorithm that finds a solution. We showed that instantiations of the general learning method consistently outperform single imputations.

Acknowledgments

We gratefully acknowledge support from STRATO Rechenzentrum AG.

References

Argyriou, A., Micchelli, C., & Pontil, M. (2005). Learning convex combinations of continuously parameterized basic kernels. *Proceedings of the 18th Conference on Learning Theory.*

Chechik, G., Heitz, G., Elidan, G., Abbeel, P., & Koller, D. (2007). Max-margin classification of incomplete data. *Advances in Neural Information Processing Systems 19.*

Liao, X., Li, H., & Carin, L. (2007). Quadratically gated mixture of experts for incomplete data classification. *Proceedings of the 24th International Conference on Machine learning.*

Micchelli, C., & Pontil, M. (2007). Feature space perspectives for learning the kernel. *Machine Learning, 66.*

Shivaswamy, P. K., Bhattacharyya, C., & Smola, A. J. (2006). Second order cone programming approaches for handling missing and uncertain data. *Journal of Machine Learning Research, 7.*

Smola, A., Vishwanathan, S., & Hofmann, T. (2005). Kernel methods for missing variables. *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics.*

Williams, D., & Carin, L. (2005). Analytical kernel matrix completion with incomplete multi-view data. *Proceedings of the ICML 2005 Workshop on Learning With Multiple Views.*

Williams, D., Liao, X., Xue, Y., & Carin, L. (2005). Incomplete-data classification using logistic regression. *Proceedings of the 22nd International Conference on Machine learning.*

Throttling Poisson Processes

Uwe Dick   Peter Haider   Thomas Vanck   Michael Brückner   Tobias Scheffer
University of Potsdam
Department of Computer Science
August-Bebel-Strasse 89, 14482 Potsdam, Germany
{uwedick,haider,vanck,mibrueck,scheffer}@cs.uni-potsdam.de

Abstract

We study a setting in which Poisson processes generate sequences of decision-making events. The optimization goal is allowed to depend on the *rate* of decision outcomes; the rate may depend on a potentially long backlog of events and decisions. We model the problem as a Poisson process with a throttling policy that enforces a data-dependent rate limit and reduce the learning problem to a convex optimization problem that can be solved efficiently. This problem setting matches applications in which damage caused by an attacker grows as a function of the rate of unsuppressed hostile events. We report on experiments on abuse detection for an email service.

1 Introduction

This paper studies a family of decision-making problems in which discrete events occur on a continuous time scale. The time intervals between events are governed by a Poisson process. Each event has to be met by a decision to either suppress or allow it. The optimization criterion is allowed to depend on the *rate* of decision outcomes within a time interval; the criterion is not necessarily a sum of a loss function over individual decisions.

The problems that we study cannot adequately be modeled as Mavkov or semi-Markov decision problems because the probability of transitioning from any value of decision rates to any other value depends on the exact points in time at which each event occurred in the past. Encoding the entire backlog of time stamps in the state of a Markov process would lead to an unwieldy formalism. The learning formalism which we explore in this paper models the problem directly as a Poisson process with a throttling policy that depends on an explicit data-dependent rate limit, which allows us to refer to a result from queuing theory and derive a convex optimization problem that can be solved efficiently.

Consider the following two scenarios as motivating applications. In order to stage a successful denial-of-service attack, an assailant has to post requests at a rate that exceeds the capacity of the service. A prevention system has to meet each request by a decision to suppress it, or allow it to be processed by the service provider. Suppressing legitimate requests runs up costs. Passing few abusive requests to be processed runs up virtually no costs. Only when the rate of passed abusive requests exceeds a certain capacity, the service becomes unavailable and costs incur. The following second application scenario will serve as a running example throughout this paper. Any email service provider has to deal with a certain fraction of accounts that are set up to disseminate phishing messages and email spam. Serving the occasional spam message causes no harm other than consuming computational ressources. But if the rate of spam messages that an outbound email server discharges triggers alerting mechanisms of other providers, then that outbound server will become blacklisted and the service is disrupted. Naturally, suppressing any legitimate message is a disruption to the service, too.

Let $\mathbf{x}$ denote a sequence of decision events $x_1, \ldots, x_n$; each event is a point $x_i \in \mathcal{X}$ in an instance space. Sequence $\mathbf{t}$ denotes the time stamps $t_i \in \mathbb{R}_+$ of the decision events with $t_i < t_{i+1}$. We define an episode e by the tuple $e = (\mathbf{x}, \mathbf{t}, y)$ which includes a label $y \in \{-1, +1\}$. In our application, an episode corresponds to the sequence of emails sent within an observation interval from a legitimate ($y = -1$) or abusive ($y = +1$) account e. We write $\mathbf{x}_i$ and $\mathbf{t}_i$ to denote the initial sequence of the first i elements of $\mathbf{x}$ and $\mathbf{t}$, respectively. Note that the length n of the sequences can be different for different episodes.

Let $\mathcal{A} = \{-1, +1\}$ be a binary decision set, where $+1$ corresponds to suppressing an event and -1 corresponds to passing it. The decision model π gets to make a decision $\pi(\mathbf{x}_i, \mathbf{t}_i) \in \mathcal{A}$ at each point in time t_i at which an event occurs.

The *outbound rate* $r^\pi(t'|\mathbf{x}, \mathbf{t})$ at time t' for episode e and decision model π is a crucial concept. It counts the number of events that were let pass during a time interval of lengh τ ending *before* t'. It is therefore defined as $r^\pi(t'|\mathbf{x}, \mathbf{t}) = |\{i : \pi(\mathbf{x}_i, \mathbf{t}_i) = -1 \wedge t_i \in [t' - \tau, t')\}|$. In outbound spam throttling, τ corresponds to the time interval that is used by other providers to estimate the incoming spam rate.

We define an immediate loss function $\ell : Y \times \mathcal{A} \rightarrow \mathbb{R}_+$ that specifies the immediate loss of deciding $a \in \mathcal{A}$ for an event with label $y \in Y$ as

$$\ell(y, a) \quad = \quad \begin{cases} c_+ & y = +1 \wedge a = -1 \\ c_- & y = -1 \wedge a = +1 \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where c_+ and c_- are positive constants, corresponding to costs of false positive and false negative decisions. Additionally, the rate-based loss $\lambda : Y \times \mathbb{R}_+ \rightarrow \mathbb{R}_+$ is the loss that runs up per unit of time. We require λ to be a convex, monotonically increasing function in the outbound rate for $y = +1$ and to be 0 otherwise. The rate-based loss reflects the risk of the service getting blacklisted based on the current sending behaviour. This risk grows in the rate of spam messages discharged and the duration over which a high sending rate of spam messages is maintained.

The total loss of a model π for an episode $e = (\mathbf{x}, \mathbf{t}, y)$ is therefore defined as

$$L(\pi; \mathbf{x}, \mathbf{t}, y) = \int_{t_1}^{t_n + \tau} \lambda\left(y, r^\pi(t'|\mathbf{x}, \mathbf{t})\right) dt' + \sum_{i=1}^{n} \ell\left(y, \pi(\mathbf{x}_i, \mathbf{t}_i)\right) \tag{2}$$

The first term penalizes a high rate of unsuppressed events with label $+1$—in our example, a high rate of unsuppressed spam messages—whereas the second term penalizes each decision individually. For the special case of $\lambda = 0$, the optimization criterion resolves to a risk, and the problem becomes a standard binary classification problem.

An unknown target distribution over $p(\mathbf{x}, \mathbf{t}, y)$ induces the overall optimization goal $\mathbb{E}_{\mathbf{x},\mathbf{t},y}[L(\pi; \mathbf{x}, \mathbf{t}, y)]$. The learning problem consists in finding $\pi^* = \text{argmin}_\pi \mathbb{E}_{\mathbf{x},\mathbf{t},y}[L(\pi; \mathbf{x}, \mathbf{t}, y)]$ from a training sample of tuples $D = \{(\mathbf{x}^1_{n^1}, \mathbf{t}^1_{n^1}, y^1), \ldots, (\mathbf{x}^m_{n^m}, \mathbf{t}^m_{n^m}, y^m)\}$.

2 Poisson Process Model

We assume the following data generation process for episodes $e = (\mathbf{x}, \mathbf{t}, y)$ that will allow us to derive an optimization problem to be solved by the learning procedure. First, a rate parameter ρ, label y, and the sequence of instances $\mathbf{x}$, are drawn from a joint distribution $p(\mathbf{x}, \rho, y)$. Rate ρ is the parameter of a Poisson process $p(\mathbf{t}|\rho)$ which now generates time sequence $\mathbf{t}$. The expected loss of decision model π is taken over all input sequences $\mathbf{x}$, rate parameter ρ, label y, and over all possible sequences of time stamps $\mathbf{t}$ that can be generated according to the Poisson process.

$$\mathbb{E}_{\mathbf{x},\mathbf{t},y}[L(\pi; \mathbf{x}, \mathbf{t}, y)] \quad = \quad \int_{\mathbf{x}} \int_{\mathbf{t}} \int_{\rho} \sum_{y} L(\pi; \mathbf{x}, \mathbf{t}, y) p(\mathbf{t}|\rho) p(\mathbf{x}, \rho, y) d\rho d\mathbf{t} d\mathbf{x} \tag{3}$$

2.1 Derivation of Empirical Loss

In deriving the empirical counterpart of the expected loss, we want to exploit our assumption that time stamps are generated by a Poisson process with unknown but fixed rate parameter. For each input episode $(\mathbf{x}, \mathbf{t}, y)$, instead of minimizing the expected loss over the single observed sequence of time stamps, we would therefore like to minimize the expected loss over all sequences of time stamps generated by a Poisson process with the rate parameter that has most likely generated the observed sequence of time stamps. Equation 4 introduces the observed time sequence of time stamps $\mathbf{t}'$ into Equation 3 and uses the fact that the rate parameter ρ is independent of $\mathbf{x}$ and y given $\mathbf{t}'$. Equation 5 rearranges the terms, and Equation 6 writes the central integral as a conditional expected value of the loss given the rate ρ. Finally, Equation 7 approximates the integral over all values of ρ by a single summand with value ρ^* for each episode.

$$\begin{aligned}
\mathbb{E}_{\mathbf{x},\mathbf{t},y}[L(\pi;\mathbf{x},\mathbf{t},y)] &= \int_{\mathbf{t}'}\int_{\mathbf{x}}\int_{\mathbf{t}}\int_{\rho}\sum_{y} L(\pi;\mathbf{x},\mathbf{t},y)p(\mathbf{t}|\rho)p(\rho|\mathbf{t}')p(\mathbf{x},\mathbf{t}',y)d\rho d\mathbf{t}d\mathbf{x}d\mathbf{t}' && (4)\\
&= \int_{\mathbf{t}'}\int_{\mathbf{x}}\sum_{y}\left(\int_{\rho}\left(\int_{\mathbf{t}} L(\pi;\mathbf{x},\mathbf{t},y)p(\mathbf{t}|\rho)d\mathbf{t}\right)p(\rho|\mathbf{t}')d\rho\right)p(\mathbf{x},\mathbf{t}',y)d\mathbf{x}d\mathbf{t}' && (5)\\
&= \int_{\mathbf{t}'}\int_{\mathbf{x}}\sum_{y}\left(\int_{\rho}(\mathbb{E}_{\mathbf{t}}\left[L(\pi;\mathbf{x},\mathbf{t},y)\mid\rho\right]p(\rho|\mathbf{t}')d\rho\right)p(\mathbf{x},\mathbf{t}',y)d\mathbf{x}d\mathbf{t}' && (6)\\
&\approx \int_{\mathbf{t}'}\int_{\mathbf{x}}\sum_{y}\mathbb{E}_{\mathbf{t}}\left[L(\pi;\mathbf{x},\mathbf{t},y)\mid\rho^*\right]p(\mathbf{x},\mathbf{t}',y)d\mathbf{x}d\mathbf{t}' && (7)
\end{aligned}$$

We arrive at the regularized risk functional in Equation 8 by replacing $p(\mathbf{x}, \mathbf{t}', y)$ by $\frac{1}{m}$ for all observations in D and inserting MAP estimate ρ_e^* as parameter that generated time stamps $\mathbf{t}^e$. The influence of the convex regularizer Ω is determined by regularization parameter $\eta > 0$.

$$\begin{aligned}
\hat{\mathbb{E}}_{\mathbf{x},\mathbf{t},y}[L(\pi;\mathbf{x},\mathbf{t},y)] &= \frac{1}{m}\sum_{e=1}^{m}\mathbb{E}_{\mathbf{t}}\left[L(\pi;\mathbf{x}^e,\mathbf{t},y^e)\mid\rho_e^*\right]+\eta\Omega(\pi) && (8)\\
\text{with} \quad & \rho_e^* = \operatorname{argmax}_{\rho} p(\rho|\mathbf{t}^e)
\end{aligned}$$

Minimizing this risk functional is the basis of the learning procedure in the next section. As noted in Section 1, for the special case when the rate-based loss λ is zero, the problem reduces to a standard weighted binary classification problem and would be easy to solve with standard learning algorithms. However, as we will see in Section 4, the λ-dependent loss makes the task of learning a decision function hard to solve; attributing individual decisions with their "fair share" of the rate loss—and thus estimating the cost of the decision—is problematic. The Erlang learning model of Section 3 employs a decision function that allows to factorize the rate loss naturally.

3 Erlang Learning Model

In the following we derive an optimization problem that is based on modeling the policy as a data-dependent rate limit. This allows us to apply a result from queuing theory and approximate the empirical risk functional of Equation (8) with a convex upper bound. We define decision model π in terms of the function $f_\theta(\mathbf{x}_i, \mathbf{t}_i) = \exp(\theta^{\mathsf{T}}\phi(\mathbf{x}_i, \mathbf{t}_i))$ which sets a limit on the admissible rate of events, where ϕ is some feature mapping of the initial sequence $(\mathbf{x}_i, \mathbf{t}_i)$ and θ is a parameter vector. The throttling model is defined as

$$\pi(\mathbf{x}_i,\mathbf{t}_i) = \begin{cases} -1 \text{ (``allow'')} & \text{if } r^{\pi}(t_i|\mathbf{x}_i,\mathbf{t}_i)+1 \le f_\theta(\mathbf{x}_i,\mathbf{t}_i) \\ +1 \text{ (``suppress'')} & \text{otherwise.}\end{cases} \qquad (9)$$

The decision model blocks event x_i, if the number of instances that were sent within $[t_i - \tau, t_i)$, plus the current instance, would exceed rate limit $f_\theta(\mathbf{x}_i, \mathbf{t}_i)$. We will now transform the optimization goal of Equation 8 into an optimization problem that can be solved by standard convex optimization tools. To this end, we first decompose the expected loss of an input sequence given the rate parameter in Equation 8 into immediate and rate-dependent loss terms. Note that $\mathbf{t}^e$ denotes the observed training sequence whereas $\mathbf{t}$ serves as expectation variable for the expectation $\mathbb{E}_{\mathbf{t}}[\cdot|{\rho_e}^*]$ over all sequences conditional on the Poisson process rate parameter ${\rho_e}^*$ as in Equation 8.

$$\begin{aligned}
&\mathbb{E}_{\mathbf{t}}\left[L(\pi;\mathbf{x}^e,\mathbf{t},y^e)\mid\rho_e^*\right]\\
&= \mathbb{E}_{\mathbf{t}}\left[\int_{t_1}^{t_{n^e}+\tau}\lambda\left(y^e,r^\pi(t'|\mathbf{x}^e,\mathbf{t})\right)dt'\mid\rho_e^*\right]+\sum_{i=1}^{n^e}\mathbb{E}_{\mathbf{t}}[\ell(y^e,\pi(\mathbf{x}_i^e,\mathbf{t}_i))\mid\rho_e^*] && (10)\\
&= \mathbb{E}_{\mathbf{t}}\Big[\int_{t_1}^{t_{n^e}+\tau}\lambda\left(y^e,r^\pi(t'|\mathbf{x}^e,\mathbf{t})\right)dt'\mid\rho_e^*\Big]+\sum_{i=1}^{n^e}\mathbb{E}_{\mathbf{t}}\Big[\delta\big(\pi(\mathbf{x}_i^e,\mathbf{t}_i)\neq y^e\big)\mid\rho_e^*\Big]\ell(y^e,-y^e) && (11)
\end{aligned}$$

Equation 10 uses the definition of the loss function in Equation 2. Equation 11 exploits that only decisions against the correct label, $\pi(\mathbf{x}_i^e,\mathbf{t}_i)\neq y^e$, incur a positive loss $\ell(y,\pi(\mathbf{x}_i^e,\mathbf{t}_i))$.

We will first derive a convex approximation of the expected rate-based loss $\mathbb{E}_{\mathbf{t}}[\int_{t_1}^{t_{n^e}+\tau}\lambda\left(y^e,r^\pi(t'|\mathbf{x}^e,\mathbf{t})\right)dt'|\rho_e^*]$ (left side of Equation 11). Our definition of the decision model allows us to factorize the expected rate-based loss into contributions of individual rate limit decisions. The convexity will be addressed by Theorem 1.

Since the outbound rate r^π increases only at decision points t_i, we can upper-bound its value with the value immediately after the most recent decision in Equation 12. Equation 13 approximates the actual outbound rate with the rate limit given by $f_\theta(\mathbf{x}_i^e,\mathbf{t}_i^e)$. This is reasonable because the outbound rate depends on the policy decisions which are defined in terms of the rate limit. Because $\mathbf{t}$ is generated by a Poisson process, $\mathbb{E}_{\mathbf{t}}[t_{i+1}-t_i\mid\rho_e^*]=\frac{1}{\rho_e^*}$ (Equation 14).

$$\begin{aligned}
&\mathbb{E}_{\mathbf{t}}\left[\int_{t_1}^{t_{n^e}+\tau}\lambda\left(y^e,r^\pi(t'|\mathbf{x}^e,\mathbf{t})\right)dt'\mid\rho_e^*\right]\\
&\leq \sum_{i=1}^{n^e-1}\mathbb{E}_{\mathbf{t}}[t_{i+1}-t_i\mid\rho_e^*]\lambda(y^e,r^\pi(t_i|\mathbf{x}^e,\mathbf{t}))+\tau\lambda(y^e,r^\pi(t_{n^e}|\mathbf{x}^e,\mathbf{t})) && (12)\\
&\approx \sum_{i=1}^{n^e-1}\mathbb{E}_{\mathbf{t}}[t_{i+1}-t_i\mid\rho_e^*]\lambda\big(y^e,f_\theta(\mathbf{x}_i^e,\mathbf{t}_i^e)\big)+\tau\lambda\big(y^e,f_\theta(\mathbf{x}_{n^e}^e,\mathbf{t}_{n^e}^e)\big) && (13)\\
&= \sum_{i=1}^{n^e-1}\frac{1}{{\rho_e}^*}\lambda\big(y^e,f_\theta(\mathbf{x}_i^e,\mathbf{t}_i^e)\big)+\tau\lambda\big(y^e,f_\theta(\mathbf{x}_{n^e}^e,\mathbf{t}_{n^e}^e)\big) && (14)
\end{aligned}$$

We have thus established a convex approximation of the left side of Equation 11.

We will now derive a closed form approximation of $\mathbb{E}_{\mathbf{t}}[\delta\left(\pi(\mathbf{x}_i^e,\mathbf{t}_i)\neq y^e\right)\mid\rho_e^*]$, the second part of the loss functional in Equation 11. Queuing theory provides a convex approximation: The *Erlang-B* formula [5] gives the probability that a queuing process which maintains a constant rate limit of f within a time interval of τ will block an event when events are generated by a Poisson process with given rate parameter ρ. Fortet's formula (Equation 15) generalizes the Erlang-B formula for non-integer rate limits.

$$B(f,\rho\tau)=\frac{1}{\int_0^\infty e^{-z}(1+\frac{z}{\rho\tau})^f dz} \qquad (15)$$

The integral can be computed efficiently using a rapidly converging series, c.f. [5]. The formula requires a constant rate limit, so that the process can reach an equilibrium. In our model, the rate limit $f_\theta(\mathbf{x}_i,\mathbf{t}_i)$ is a function of the sequences $\mathbf{x}_i$ and $\mathbf{t}_i$ until instance x_i, and Fortet's formula therefore serves as an approximation.

$$\begin{aligned}
\mathbb{E}_{\mathbf{t}}\left[\delta(\pi(\mathbf{x}_i^e,\mathbf{t}_i)=1)|\rho_e^*\right] &\approx B(f_\theta(\mathbf{x}_i^e,\mathbf{t}_i^e),\rho_e^*\tau) && (16)\\
&= \left[\int_0^\infty e^{-z}(1+\frac{z}{\rho_e^*\tau})^{f_\theta(\mathbf{x}_i^e,\mathbf{t}_i^e)}dz\right]^{-1} && (17)
\end{aligned}$$

Unfortunately, Equation 17 is not convex in θ. We approximate it with the convex upper bound $-\log\left(1-B(f_\theta(\mathbf{x}_i^e,\mathbf{t}_i^e),\rho_e^*\tau)\right)$ (cf. the dashed green line in the left panel of Figure 2(b) for an illustration). This is an upper bound, because $-\log p\geq 1-p$ for $0\leq p\leq 1$; its convexity is addressed by Theorem 1. Likewise, $\mathbb{E}_{\mathbf{t}}\left[\delta(\pi(\mathbf{x}_i^e,\mathbf{t}_i)=-1)|\rho_e^*\right]$ is approximated by upper bound $\log\left(B(f_\theta(\mathbf{x}_i^e,\mathbf{t}_i^e),\rho_e^*\tau)\right)$. We have thus derived a convex upper bound of $\mathbb{E}_{\mathbf{t}}[\delta\left(\pi(\mathbf{x}_i^e,\mathbf{t}_i)\neq y^e\right)|\rho_e^*]$.

Combining the two components of the optimization goal (Equation 11) and adding convex regularizer $\Omega(\theta)$ and regularization parameter $\eta > 0$ (Equation 8), we arrive at an optimization problem for finding the optimal policy parameters θ.

Optimization Problem 1 (Erlang Learning Model)**.** *Over θ, minimize*

$$\begin{aligned} R(\theta) = \frac{1}{m}\sum_{e=1}^{m}\Bigg\{ & \sum_{i=1}^{n^e-1}\frac{1}{{\rho_e}^*}\lambda\big(y^e, f_\theta(\mathbf{x}_i^e, \mathbf{t}_i^e)\big) + \tau\lambda\big(y^e, f_\theta(\mathbf{x}_{n^e}^e, \mathbf{t}_{n^e}^e)\big) \\ & + \sum_{i=1}^{n^e} -\log\left[\delta(y^e{=}1) - y^e B\big(f_\theta(\mathbf{x}_i^e, \mathbf{t}_i^e), \rho_e^*\tau\big)\right]\ell(y^e, -y^e)\Bigg\} + \eta\Omega(\theta) \end{aligned} \tag{18}$$

Next we show that minimizing risk functional R amounts to solving a convex optimization problem.

Theorem 1 (Convexity of R)**.** *$R(\theta)$ is a convex risk functional in θ for any $\rho_e^* > 0$ and $\tau > 0$.*

Proof. The convexity of λ and Ω follows from their definitions. It remains to be shown that both $-\log B(f_\theta(\cdot), \rho_e^*\tau))$ and $-\log(1 - B(f_\theta(\cdot), \rho_e^*)$ are convex in θ. Component $\ell(y^e, -y^e)$ of Equation 18 is independent of θ. It is known that Fortet's formula $B(f, {\rho_e}^*\tau))$ is convex, monotically decreasing, and positive in f for $\rho_e^*\tau > 0$ [5]. Furthermore $-\log(B(f, \rho_e^*\tau)))$ is convex and monotonically increasing. Since $f_\theta(\cdot)$ is convex in θ, it follows that $-\log(B(f_\theta(\cdot), \rho_e^*))$ is also convex. Next, we show that $-\log(1 - B(f_\theta(\cdot), \rho_e^*\tau)))$ is convex and monotonically decreasing. From the above it follows that $b(f) = 1 - B(f, \rho_e^*\tau))$ is monotonically increasing, concave and positive. Therefore, $\frac{d^2}{df^2} - \ln(b(f)) = \frac{1}{b^2(f)}b'(f) + b''(f)\frac{-1}{b(f)} \geq 0$ as both summands are positive. Again, it follows that $-\log(1 - B(f_\theta(\cdot), \rho_e^*\tau)))$ is convex in θ due to the definition of f_θ. □

4 Prior Work and Reference Methods

We will now discuss how the problem of minimizing the expected loss, $\pi^* = \text{argmin}_\pi \mathbb{E}_{\mathbf{x},\mathbf{t},y}[L(\pi; \mathbf{x}, \mathbf{t}, y)]$, from a sample of sequences $\mathbf{x}$ of events with labels y and observed rate parameters ρ^* relates to previously studied methods. Sequential decision-making problems are commonly solved by reinforcement learning approaches, which have to attribute the loss of an episode (Equation 2) to individual decisions in order to learn to decide optimally in each state. Thus, a crucial part of defining an appropriate procedure for learning the optimal policy consists in defining an appropriate state-action loss function. $Q^\pi(s, a)$ estimates the loss of performing action a in state s when following policy π for the rest of the episode.

Several different state-action loss functions for related problems have been investigated in the literature. For example, policy gradient methods such as in [4] assign the loss of an episode to individual decisions proportional to the log-probabilities of the decisions. Other approaches use sampled estimates of the rest of the episode $Q(s_i, a_i) = L(\pi, \mathbf{s}) - L(\pi, \mathbf{s}_i)$ or the expected loss if a distribution of states of the episode is known [7]. Such general purpose methods, however, are not the optimal choice for the particular problem instance at hand. Consider the special case $\lambda = 0$, where the problem reduces to a sequence of independent binary decisions. Assigning the cumulative loss of the episode to all instances leads to a grave distortion of the optimization criterion.

As reference in our experiments we use a state-action loss function that assigns the immediate loss $\ell(y, a_i)$ to state s_i only. Decision a_i determines the loss incurred by λ only for τ time units, in the interval $[t_i, t_i + \tau)$. The corresponding rate loss is $\int_{t_i}^{t_i+\tau}\lambda(y, r^\pi(t'|\mathbf{x}, \mathbf{t}))dt'$. Thus, the loss of deciding $a_i = -1$ instead of $a_i = +1$ is the difference in the corresponding λ-induced loss. Let $\mathbf{x}^{-i}, \mathbf{t}^{-i}$ denote the sequence $\mathbf{x}, \mathbf{t}$ without instance x_i. This leads to a state-action loss function that is the sum of immediate loss and λ-induced loss; it serves as our first baseline.

$$Q_{it}^\pi(s_i, a) = \ell(y, a) + \delta(a{=}-1)\int_{t_i}^{t_i+\tau}\lambda(y, r^\pi(t'|\mathbf{x}^{-i}, \mathbf{t}^{-i}) + 1) - \lambda(y, r^\pi(t'|\mathbf{x}^{-i}, \mathbf{t}^{-i}))dt' \tag{19}$$

By approximating $\int_{t_i}^{t_i+\tau}\lambda(y, r^\pi(t'|\mathbf{x}, \mathbf{t}))$ with $\tau\lambda(y, r^\pi(t_i|\mathbf{x}, \mathbf{t}))$, we define the state-action loss function of a second plausible state-action loss that, instead of using the observed loss to estimate the loss of an action, approximates it with the loss that would be incurred by the current outbound rate $r^\pi(t_i|\mathbf{x}^{-i}, \mathbf{t}^{-i})$ for τ time units.

$$Q^\pi_{ub}(s_i, a) = \ell(y, a) + \delta(a{=}-1)\left[\tau\left(\lambda(y, r^\pi(t_i|\mathbf{x}^{-i}, \mathbf{t}^{-i}) + 1) - \lambda(y, r^\pi(t_i|\mathbf{x}^{-i}, \mathbf{t}^{-i}))\right)\right] \quad (20)$$

The state variable s has to encode all information a policy needs to decide. Since the loss crucially depends on outbound rate $r^\pi(t'|\mathbf{x}, \mathbf{t})$, any throttling model must have access to the current outbound rate. The transition between a current and a subsequent rate depends on the time at which the next event occurs, but also on the entire backlog of events, because past events may drop out of the interval τ at any time. In analogy to the information that is available to the Erlang learning model, it is natural to encode states s_i as a vector of features $\phi(\mathbf{x}_i, \mathbf{t}_i)$ (see Section 5 for details) together with the current outbound rate $r^\pi(t_i|\mathbf{x}, \mathbf{t})$. Given a representation of the state and a state-action loss function, different approaches for defining the policy π and optimizing its parameters have been investigated. For our baselines, we use the following two methods.

Policy gradient. Policy gradient methods model a stochastic policy directly as a parameterized decision function. They perform a gradient descent that always converges to a local optimum [8]. The gradient of the expected loss with respect to the parameters is estimated in each iteration k for the distribution over episodes, states, and losses that the *current* policy π_k induces. However, in order to achieve fast convergence to the optimal polity, one would need to determine the gradient for the distribution over episodes, states, and losses induced by the *optimal* policy. We implement two policy gradient algorithms for experimentation which only differ in using Q_{it} and Q_{ub}, respectively. They are denoted PG$_{it}$ and PG$_{ub}$ in the experiments. Both use a logistic regression function as decision function, the two-class equivalent of the Gibbs distribution which is used in the literature.

Iterative Classifier. The second approach is to represent policies as classifiers and to employ methods for supervised classification learning. A variety of papers addresses this approach [6, 3, 7]. We use an algorithm that is inspired by [1, 2] and is adapted to the problem setting at hand. Blatt and Hero [2] investigate an algorithm that finds non-stationary policies for two-action T-step MDPs by solving a sequence of one-step decisions via a binary classifier. Classifiers π_t for time step t are learned iteratively on the distribution of states generated by the policy $(\pi_0, \ldots, \pi_{t-1})$. Our derived algorithm iteratively learns weighted support vector machine (SVM) classifier π_{k+1} in iteration $k{+}1$ on the set of instances and losses $Q^{\pi_k}(s, a)$ that were observed after classifier π_k was used as policy on the training sample. The weight vector of π_k is denoted θ_k. The weight of misclassification of s is given by $Q^{\pi_k}(s, -y)$. The SVM weight vector is altered in each iteration as $\theta_{k+1} = (1-\alpha_k)\theta_k + \alpha_k\hat{\theta}$, where $\hat{\theta}$ is the weight vector of the new classifier that was learned on the observed losses. In the experiments, two iterative SVM learner were implemented, denoted It-SVM$_{it}$ and It-SVM$_{ub}$, corresponding to the used state-action losses Q_{it} and Q_{ub}, respectively. Note that for the special case $\lambda = 0$ the iterative SVM algorithm reduces to a standard SVM algorithm.

All four procedures iteratively estimate the loss of a policy decision on the data via a state-action loss function and learn a new policy π based on this estimated cost of the decisions. Convergence guarantees typically require the Markov assumption; that is, the process is required to possess a stationary transition distribution $P(s_{i+1}|s_i, a_i)$. Since the transition distribution in fact depends on the entire backlog of time stamps and the duration over which state s_i has been maintained, the Markov assumption is violated to some extent in practice. In addition to that, λ-based loss estimates are sampled from a Poisson process. In each iteration π is learned to minimize sampled and inherently random losses of decisions. Thus, convergence to a robust solution becomes unlikely. In contrast, the Erlang learning model directly minimizes the λ-loss by assigning a rate limit. The rate limit implies an expectation of decisions. In other words, the λ-based loss is minimized without expliciteiy estimating the loss of any decisions that are implied by the rate limit. The convexity of the risk functional in Optimization Problem 1 guarantees convergence to the global optimum.

5 Application

The goal of our experiments is to study the relative benefits of the Erlang learning model and the four reference methods over a number of loss functions. The subject of our experimentation is the problem of suppressing spam and phishing messages sent from abusive accounts registered at a large email service provider. We sample approximately 1,000,000 emails sent from approximately

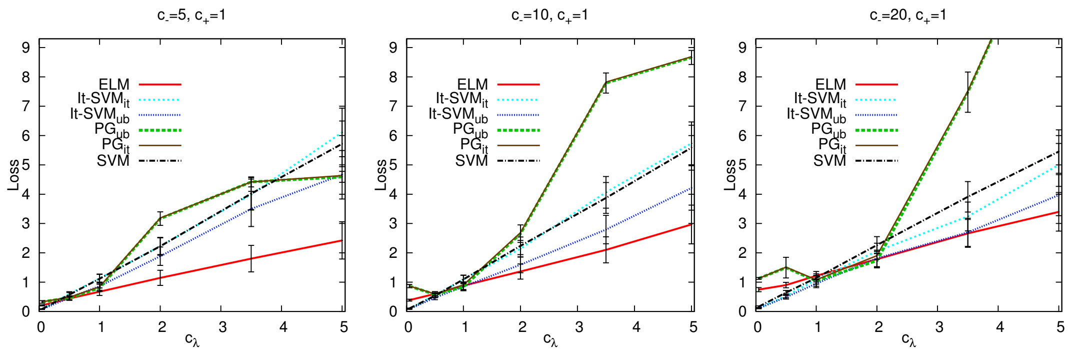


Figure 1: Average loss on test data depending on the influence of the rate loss c_λ for different immediate loss constants c_- and c_+.

10,000 randomly selected accounts over two days and label them automatically based on information passed by other email service providers via feedback loops (in most cases triggered by "report spam" buttons). Because of this automatic labeling process, the labels contain a certain smount of noise.

Feature mapping ϕ determines a vector of moving average and moving variance estimates of several attributes of the email stream. These attributes measure the frequency of subject changes and sender address changes, and the number of recipients. Other attributes indicate whether the subject line or the sender address have been observed before within a window of time. Additionally, a moving average estimate of the rate ρ is used as feature. Finally, other attributes quantify the size of the message and the score returned by a content-based spam filter employed by the email service.

We implemented the baseline methods that were descibed in Section 4, namely the iterative SVM methods It-SVM$_{ub}$ and It-SVM$_{it}$ and the policy gradient methods PG$_{ub}$ and PG$_{it}$. Additionally, we used a standard support vector machine classifier SVM with weights of misclassification corresponding to the costs defined in Equation 1. The Erlang learning model is denoted ELM in the plots. Linear decision functions were used for all baselines.

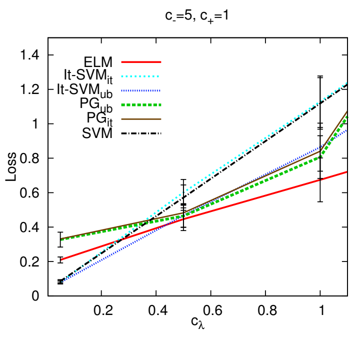
In our experiments, we assume a cost that is quadratic in the outbound rate. That is, $\lambda(1, r^\pi(t'|\mathbf{x}, \mathbf{t}))) = c_\lambda \cdot r^\pi(t'|\mathbf{x}, \mathbf{t})^2$ with $c_\lambda > 0$ determining the influence of the rate loss to the overall loss. The time interval τ was chosen to be 100 seconds. Regularizer $\Omega(\theta)$ as in Optimization problem 1 is the commonly used squared l_2-norm $\Omega(\theta) = \|\theta\|_2^2$.

We evaluated our method for different costs of incorrectly classified non-spam emails (c_-), incorrectly classified spam emails (c_+) (see the definition of ℓ in Equation 1), and rate of outbound spam messages (c_λ). For each setting, we repeated 100 runs; each run used about 50%, chosen at random, as training data and the remaining part as test data. Splits where chosen such that there were equally many spam episodes in training and test set. We tuned the regularization parameter η for the Erlang learning model as well as the corresponding regularization parameters of the iterative SVM methods and the standard SVM on a separate tuning set that was split randomly from the training data.

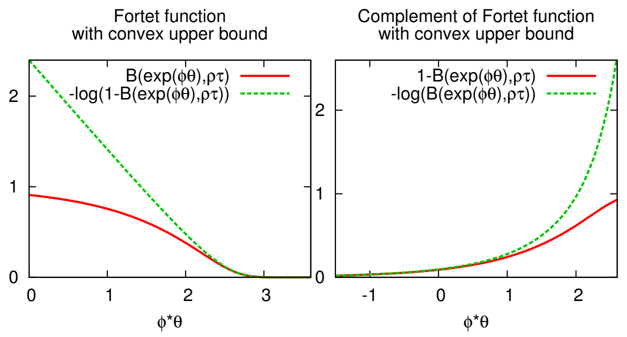
5.1 Results

Figure 1 shows the resulting average loss of the Erlang learning model and reference methods. Each of the three plots shows loss versus parameter c_λ which determines the influence of the rate loss on the overall loss. The left plot shows the loss for $c_- = 5$ and $c_+ = 1$, the center plot for $(c_- = 10, c_+ = 1)$, and the right plot for $(c_- = 20, c_+ = 1)$.

We can see in Figure 1 that the Erlang learning model outperforms all baseline methods for larger values of c_λ—more influence of the rate dependent loss on the overall loss—in two of the three settings. For $c_- = 20$ and $c_+ = 1$ (right panel), the performance is comparable to the best baseline method It-SVM$_{ub}$; only for the largest shown $c_\lambda = 5$ does the ELM outperform this baseline. The iterative classifier It-SVM$_{ub}$ that uses the approximated state-action loss Q_{ub} performs uniformly better than It-SVM$_{it}$, the iterative SVM method that uses the sampled loss from the previous iteration. It-SVM$_{it}$ itself surprisingly shows very similar performance to that of the standard SVM method; only for the setting $c_- = 20$ and $c_+ = 1$ in the right panel does this iterative SVM method show superior performance. Both policy gradient methods perform comparable to the Erlang learning model for smaller values of c_λ but deteriorate for larger values.



(a) Average loss and standard error for small values of c_λ.



(b) Left: Fortet's formula $B(e^{\phi\theta}, \rho\tau)$ (Equation 17) and its upper bound $-\log(1 - B(e^{\phi\theta}, \rho))$ for $\rho\tau = 10$. Right: $1 - B(e^{\phi\theta}, \rho)$ and respective upper bound $-\log(B(e^{\phi\theta}, \rho))$.

As expected, the iterative SVM and the standard SVM algorithms perform better than the Erlang learning model and policy gradient models if the influence of the rate pedendent loss is very small. This can best be seen in Figure 2(a). It shows a detail of the results for the setting $c_- = 5$ and $c_+ = 1$, for c_λ ranging only from 0 to 1. This is the expected outcome following the considerations in Section 4. If c_λ is close to 0, the problem approximately reduces to a standard binary classification problem, thus favoring the very good classification performance of support vector machines. However, for larger c_λ the influence of the rate dependent loss rises and more and more dominates the immediate classification loss ℓ. Consequently, for those cases — which are the important ones in this real world application — the better rate loss estimation of the Erlang learning model compared to the baselines leads to better performance.

The average training times for the Erlang learning model and the reference methods are in the same order of magnitude. The SVM algorithm took 14 minutes in average to converge to a solution. The Erlang learning model converged after 44 minutes and the policy gradient methods took approximately 45 minutes. The training times of the iterative classifier methods were about 60 minutes.

6 Conclusion

We devised a model for sequential decision-making problems in which events are generated by a Poisson process and the loss may depend on the rate of decision outcomes. Using a throttling policy that enforces a data-dependent rate-limit, we were able to factor the loss over single events. Applying a result from queuing theory led us to a closed-form approximation of the immediate event-specific loss under a rate limit set by a policy. Both parts led to a closed-form convex optimization problem. Our experiments explored the learning model for the problem of suppressing abuse of an email service. We observed significant improvements over iterative reinforcement learning baselines. The model is being employed to this end in the email service provided by web hosting firm STRATO. It has replaced a procedure of manual deactivation of accounts after inspection triggered by spam reports.

Acknowledgments

We gratefully acknowledge support from STRATO Rechenzentrum AG and the German Science Foundation DFG.

References

[1] J.A. Bagnell, S. Kakade, A. Ng, and J. Schneider. Policy search by dynamic programming. *Advances in Neural Information Processing Systems*, 16, 2004.

[2] D. Blatt and A.O. Hero. From weighted classification to policy search. *Advances in Neural Information Processing Systems*, 18, 2006.

[3] C. Dimitrakakis and M.G. Lagoudakis. Rollout sampling approximate policy iteration. *Machine Learning*, 72(3):157–171, 2008.

[4] M. Ghavamzadeh and Y. Engel. Bayesian policy gradient algorithms. *Advances in Neural Information Processing Systems*, 19, 2007.

[5] D.L. Jagerman, B. Melamed, and W. Willinger. Stochastic modeling of traffic processes. *Frontiers in queueing: models, methods and problems*, pages 271–370, 1996.

[6] M.G. Lagoudakis and R. Parr. Reinforcement learning as classification: Leveraging modern classifiers. In *Proceedings of the 20th International Conference on Machine Learning*, 2003.

[7] J. Langford and B. Zadrozny. Relating reinforcement learning performance to classification performance. In *Proceedings of the 22nd International Conference on Machine learning*, 2005.

[8] R.S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 2000.

Learning to Control a Structured-Prediction Decoder for Detection of HTTP-Layer DDoS Attackers

Uwe Dick · Tobias Scheffer

the date of receipt and acceptance should be inserted later

Abstract We focus on the problem of detecting clients that attempt to exhaust server resources by flooding a service with protocol-compliant HTTP requests. Attacks are usually coordinated by an entity that controls many clients. Modeling the application as a structured-prediction problem allows the prediction model to jointly classify a multitude of clients based on their cohesion of otherwise inconspicuous features. Since the resulting output space is too vast to search exhaustively, we employ greedy search and techniques in which a parametric controller guides the search. We apply a known method that sequentially learns the controller and the structured-prediction model. We then derive an online policy-gradient method that finds the parameters of the controller and of the structured-prediction model in a joint optimization problem; we obtain a convergence guarantee for the latter method. We evaluate and compare the various methods based on a large collection of traffic data of a web-hosting service.

1 Introduction

Distributed denial-of-service (DDoS) flooding attacks [37] intend to prevent legitimate users from using a web-based service by exhausting server or network resources. DDoS attacks can target the network level or the application level. One way for attackers to target the network level is to continuously request TCP connections and leave the connection in an incomplete state, which eventually exhausts the number of connections which the server can handle; this is called SYN flooding. Adaptive SYN-received timeouts, packet-filtering policies, and an increasing network capacity are making it more difficult to mount successful network-level attacks [26,37]. By comparison, server resources such as CPU, I/O bandwidth, database and disk throughput are becoming easier targets [4,28]. Attackers turn towards HTTP-layer flooding attacks in which they flood services with protocol-compliant

U. Dick
University of Potsdam, Department of Computer Science, Potsdam, Germany
E-mail: uwedick@cs.uni-potsdam.de

T. Scheffer
University of Potsdam, Department of Computer Science, Potsdam, Germany
E-mail: tobias.scheffer@uni-potsdam.de

requests that require the execution of scripts, expensive database operations, or the transmission of large files.

HTTP-layer attacks are more difficult to detect, because the detection mechanism ultimately has to decide whether all connecting clients have a legitimate reason for requesting a service in a particular way. In protocol-compliant application-level attacks, attackers have to sign their TCP/IP packets with their real IP address, because they have to complete the TCP handshake. One can therefore defend against flooding attacks by blacklisting offending IP addresses at the network router, provided that attacking clients can be singled out.

In order to detect attacking clients, one can engineer features of individual clients, train a classifier on labeled traffic data to detect attacking clients, and blacklist detected attackers. We follow this approach and evaluate it empirically, but the following considerations already indicate that it might work less than perfectly in practice. An individual protocol-compliant request is rarely conspicuous by itself; after all, the service is there to be requested. Most individual clients only post a small number of requests to a domain after which their IP address is not seen again. This implies that classification of individual clients will be difficult, and that aggregating information over requests into longitudinal client features [28,36,22] will only provide limited additional information.

However, DDoS attacks are usually coordinated by an entity that controls the attacking clients. Their joint programming is likely to induce some behavioral coherence of all attacking clients. Features of individual clients cannot reflect this cohesion. But a joint feature function that is parametrized with all clients x_i that interact with a domain and conjectured class labels y_i for all clients can measure the behavioral variance of all clients that are labeled as attackers. Structured-prediction methods [20,32] match this situation because they are based on joint feature functions of multiple dependent inputs x_i and their output values y_i. At application time, structured-prediction models have to solve the *decoding* problem of maximizing the decision function over all combinations of class labels. If the dependencies in the feature function are sequential or tree-structured, this maximization can be carried out efficiently using, for instance, the Viterbi algorithm for sequential data. In general as well as in this particular case, however, exhaustive search of the output space is intractable. Moreover, in our application environment, the search has to terminate after a fixed but *a priori* unknown number of computational steps due to a real-time constraint.

Collective classification algorithms [23] conduct a greedy search for the highest-scoring joint labeling of the nodes of a graph. They do so by iteratively relabeling individual nodes given the conjectured labels of all neighboring nodes. We will apply this principle, and explore the resulting algorithm empirically. More generally, when exhaustive search for a structured-prediction problem is infeasible, an *undergenerating decoder* can still search a constrained part of the output space [13]. Explicit constraints that make the remaining output space exhaustively searchable may also exclude good solutions. One may instead resort to learning a search heuristic. HC search [10,11] first learns a heuristic that guides the search to the correct output for training instances, and then uses this heuristic to control the decoder during training and application of the structured-prediction model. We will apply this principle to our application, and study the resulting algorithm.

The search heuristic of the HC-search framework is optimized to guide the decoder from an initial labeling to the correct output for all training instances. It is subsequently applied to guiding the decoder to the output that maximizes the decision function of the structured-prediction model, while this model is being learned. One may argue that a heuristic that does well at guiding the search to the correct output (that is known for the training instances) may do poorly at guiding it to the output that maximizes some decision function. We will therefore derive a policy-gradient model in which the controller and the structured-prediction model that uses the controller are learned in a joint optimization problem; we will analyze convergence properties of this model.

Defense mechanisms against DDoS attacks have so far been evaluated using artificial or semi-artificial traffic data that have been generated under plausible model assumptions of benign and offending traffic [28,36,22,8]. By contrast, we will compare all models under investigation on a large data set of network traffic that we collect in a large shared web hosting environment and classified manually. It includes unusual high-volume network traffic for more than 1,546 domains over 22,645 time intervals of 10 seconds in which we observe several million connections of more than 450,000 unique clients.

The rest of the paper is structured as follows. Section 2 derives the problem setting from our motivating application. We model the application as an anomaly-detection problem in Section 3, as the problem of independently classifying clients in Section 4, as a collective classification problem in Section 5, and as a structured-prediction problem with a parametric decoder in Section 6. Section 7 discusses how all methods can be instantiated for the attacker-identification application. We present an empirical study in Section 8; Section 9 discusses our results against the background of work. Section 10 concludes.

2 Problem Setting, Motivating Application

This section first lays out the relevant details of the application and establishes a high-level problem setting that will be cast into various learning paradigms in the following sections.

We focus on *HTTP-layer denial-of-service flooding attacks* [37], which we define to be any malicious attempts at denying the service to its legitimate users by posting protocol-compliant HTTP requests so as to exhaust any computational resource, such as CPU, bandwidth, or database throughput. Our application environment is a shared web hosting service in which a large number of domains are hosted in a large computing center. The effects of an attack can be mitigated when the IP addresses of the attacking clients can be identified: IP addresses of known attackers can be temporarily blacklisted at the router. Anomalous traffic events can extend for as little as a few minutes; attacks can run for several hours. We decompose the problem into time intervals of 10 seconds. For each domain and each time interval t, attacker detection models have to label the entirety of clients that interact with that domain as legitimate or attacker.

In our application environment, a large number of domains continuously receives requests from an even larger number of legitimate or attacking clients. Any solution has to be designed such that it can be scaled by distributing it over multiple nodes. Since attackers usually target a specific domain, we split the overall problem into an independent sub-problem for each domain. A domain is constituted by the top-level and second-level domain in the HOST field of the HTTP header ("example.com"). This allows us to distribute the solution over multiple computing nodes, each of which handles a subset of the domains.

Hence, for each domain, we arrive at an independent learning problem that is characterized by an unknown distribution $p(\mathbf{x}, \mathbf{y})$ over sets $\mathbf{x} \in \mathcal{X}$ of clients x_j that interact with the domain within a 10-seconds interval and output variables $\mathbf{y} \in \mathcal{Y}(\mathbf{x}) = \{-1, 1\}^m$ which label clients as legitimate ($y_j = -1$) or attacker ($y_j = +1$). The classification problem for each 10-seconds interval has to be solved within ten seconds—otherwise, a backlog of decisions could build up. The number of CPU cycles that are available within these 10 seconds is not known *a priori* because it depends on the overall server load. For the structured-prediction models, we encode this *anytime constraint* by limiting the number of search steps to a random number T that is governed by some distribution. We can disregard this anytime constraint for the model that treats clients as independent (Section 4), because the resulting classifier is sufficiently fast at calculating the predictions.

Misclassified legitimate requests can potentially result in lost business while misclassified abusive requests consume computational resources; when CPU capacity, bandwidth, or database throughput capacities are exhausted, the service becomes unavailable. The resulting costs will be reflected in the optimization criteria by cost terms of false-negative and false-positive decisions. When the true labels of the clients $\mathbf{x}$ are $\mathbf{y}$, a prediction of $\hat{\mathbf{y}}$ incurs costs $c(\mathbf{x}, \mathbf{y}, \hat{\mathbf{y}}) \geq 0$. We will detail the exact cost function in Section 7.

3 Anomaly Detection

In our application, an abundance of the network traffic can be observed. However, manually labeling clients as legitimate and attackers is an arduous effort. Therefore, our first take is to model attacker detection as an anomaly detection problem.

3.1 Problem Setting for Anomaly Detection

In this formulation of the problem settings, the set of clients $\mathbf{x} = \{x_1, \ldots, x_m\}$ that are observed in each 10-seconds interval is decomposed into individual clients x_j. At application time, clients are labeled independently based on the value of a parametric decision function $f_\phi(\mathbf{\Phi}_\mathbf{x}(x_j))$ which is a function of feature vector $\mathbf{\Phi}_\mathbf{x}(x_j)$. We will define feature vector $\mathbf{\Phi}_\mathbf{x}(x_j)$ in Section 7.4.2; for instance, it includes the number of different resource paths that client x_j has accessed, the number of HTTP requests that have resulted in error codes, both in terms of absolute counts and in proportion to all clients that connect to the domain.

At learning time, an unlabeled sample $\mathbf{x}_1, \ldots, \mathbf{x}_n$ of sets of clients is available. Most of the clients in the training data are legitimate, but some fraction consists of attacking clients. The unlabeled training instances are pooled into a set of feature vectors

$$L^{AD} = \bigcup_{i=1}^{n} \{\mathbf{\Phi}_{\mathbf{x}_i}(x_{i,1}), \ldots, \mathbf{\Phi}_{\mathbf{x}_i}(x_{i,m_i})\}; \tag{1}$$

training results in model parameters ϕ.

3.2 Support Vector Data Description

Support-vector data description (SVDD) is an anomaly-detection method that uses *unlabeled* data to find a model for *unusual* instances. The decision function of *SVDD* is

$$f_\phi^{SVDD}(\mathbf{\Phi}_\mathbf{x}(x_j)) = ||\mathbf{\Phi}_\mathbf{x}(x_j) - \phi||^2; \tag{2}$$

that is, *SVDD* classifies a client as an attacker if the distance between feature vector $\mathbf{\Phi}_\mathbf{x}(x_j)$ and the parameter vector ϕ that describes *normal traffic* exceeds a threshold r.

$$\hat{y}_j = \begin{cases} -1 & \text{if } f_\phi^{SVDD}(\mathbf{\Phi}_\mathbf{x}(x_j)) \leq r \\ +1 & \text{else} \end{cases} \tag{3}$$

4 Independent Classification

This section models the application as a standard classification problem.

4.1 Problem Setting for Independent Classification

Clients $\mathbf{x} = \{x_1, \ldots, x_m\}$ of each 10-seconds interval are treated as independent observations, described by feature vectors $\mathbf{\Phi_x}(x_j)$. As in Section 3.1, these vector representations are classified independently, based on the value of a parametric decision function $f_{\boldsymbol{\phi}}(\mathbf{\Phi_x}(x_j))$. However, features may be engineered to depend on properties of all clients that interact with the domain in the time interval.

In the independent classification model, misclassification costs have to decompose into a sum over individual clients: $c(\mathbf{x}, \mathbf{y}, \hat{\mathbf{y}}) = \sum_{j=1}^{m} c(x_j, y_j, \hat{y}_j)$. At learning time, a labeled sample $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$ is available. Each pair $(\mathbf{x}_i, \mathbf{y}_i)$ contains instances $x_{i,1}, \ldots, x_{i,m_i}$ and corresponding labels $y_{i,1}, \ldots, y_{i,m_i}$. The training data are pooled into independent pairs of feature vectors and corresponding class labels

$$L^{IC} = \bigcup_{i=1}^{n} \{(\mathbf{\Phi}_{\mathbf{x}_i}(x_{i,1}), y_{i,1}), \ldots, (\mathbf{\Phi}_{\mathbf{x}_i}(x_{i,m_i}), y_{i,m_i})\}, \tag{4}$$

and training results in model parameters $\boldsymbol{\phi}$.

4.2 Logistic Regression

Logistic regression (LR) is a linear classification model that we use to classify clients independently. The decision function $f_{\boldsymbol{\phi}}^{LR}(\mathbf{\Phi_x}(x_j))$ of logistic regression squashes the output of a linear model into a normalized probability by using a logistic function:

$$f_{\boldsymbol{\phi}}^{LR}(\mathbf{\Phi_x}(x_j)) = \frac{1}{1 + e^{-\boldsymbol{\phi}^{\top}\mathbf{\Phi_x}(x_j)}}. \tag{5}$$

Labels are assigned according to

$$\hat{y}_j = \begin{cases} -1 & \text{if } f_{\boldsymbol{\phi}}^{LR}(\mathbf{\Phi_x}(x_j)) \leq \frac{1}{2} \\ +1 & \text{else} \end{cases} \tag{6}$$

Logistic regression models are trained by maximizing the conditional log-likelihood of the training class labels over the parameters $\boldsymbol{\phi}$. Costs are incorporated by weighting the log-loss of each observation with the cost of misclassifying it.

5 Structured Prediction with Approximate Inference

In Section 4, the decision function has been evaluated independently for each client. This prevented the model from taking joint features of particular groups of clients based on its predicted labels into account.

5.1 Problem Setting for Structured Prediction with Approximate Inference

In the structured-prediction paradigm, a classification model infers a collective assignment $\mathbf{y}$ of labels to the entirety of clients $\mathbf{x}$ that are observed in a time interval. In our application, all clients that interact with the domain in the time interval are dependent. The model therefore has to label the nodes of a fully connected graph. This problem setting is also referred to as *collective classification* [23].

Predictions $\hat{\mathbf{y}}$ of all clients are determined as the argument $\mathbf{y}$ that maximizes a decision function $f_\phi(\mathbf{x}, \mathbf{y})$ which may depend on a joint feature vector $\mathbf{\Phi}(\mathbf{x}, \mathbf{y})$ of inputs and outputs. The feature vector may reflect arbitrary dependencies between all clients $\mathbf{x}$ and all labels $\mathbf{y}$. At application time, the decoding problem

$$\hat{\mathbf{y}} \approx \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} f_\phi(\mathbf{x}, \mathbf{y}) \tag{7}$$

has to be solved approximately within an interval of 10 seconds. The number of processing cycles that are available for each decision depends on the overall server load. We model this by constraining the number of steps which can be spent on approximating the highest-scoring output to T plus a constant number, where $T \sim p(T|\tau)$ is governed by some distribution and its value is not known in advance. At training time, a labeled sample $L = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$ is available.

5.2 Iterative Classification Algorithm

The *iterative classification algorithm (ICA)* [24] is a standard collective-classification method. We use *ICA* as a method of approximate inference for structured prediction. *ICA* uses a feature vector $\mathbf{\Phi}_{\mathbf{x},\mathbf{y}}(x_j)$ for individual nodes and internalizes labels of neighboring nodes into this feature vector. For this definition of features, decision function $f_\phi(\mathbf{x}, \mathbf{y})$ is a sum over all nodes. For a binary classification problem, we can use logistic regression and the decision function simplifies to

$$f_\phi(\mathbf{x}, \mathbf{y}) = \sum_j \begin{cases} f_{\phi'}^{LR}(\mathbf{\Phi}_{\mathbf{x},\mathbf{y}}(x_j)) & \text{if } y_j = +1 \\ 1 - f_{\phi'}^{LR}(\mathbf{\Phi}_{\mathbf{x},\mathbf{y}}(x_j)) & \text{if } y_j = -1. \end{cases} \tag{8}$$

ICA only approximately maximizes this sum by starting an initial assignment $\hat{\mathbf{y}}$ which, in our, case, is determined by *logistic regression*. It then iteratively changes labels $\hat{y}_j$ such that the summand for j is maximized, until a fixed point is reached or the maximization is terminated after T steps. When a fixed point $\hat{\mathbf{y}}$ is reached, then $\hat{\mathbf{y}}$ satisfies

$$\forall j : \hat{y}_j = \begin{cases} -1 & \text{if } f_\phi^{LR}(\mathbf{\Phi}_{\mathbf{x},\hat{\mathbf{y}}}(x_j)) \leq \frac{1}{2} \\ +1 & \text{else.} \end{cases} \tag{9}$$

6 Structured Prediction with a Parametric Decoder

In this section, we allow for a guided search of the label space. Since the space is vastly large, we allow the search do be guided by a parametric model that itself is optimized on the training data.

6.1 Problem Setting for Structured Prediction with Parametric Decoder

At application time, prediction $\hat{\mathbf{y}}$ is determined by solving the decoding problem of Equation 7; decision function $f_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{y})$ depends on a feature vector $\boldsymbol{\Phi}(\mathbf{x}, \mathbf{y})$. The decoder is allowed T (plus a constant number of) evaluations of the decision function, where $T \sim p(T|\tau)$ is governed by some distribution and its value is not known in advance. The decoder has parameters $\boldsymbol{\psi}$ that control this choice of labelings.

In the available T time steps, the decoder has to create a set of candidate labelings $Y_T(\mathbf{x})$ for which the decision function is evaluated. The decoding process starts in a state $Y_0(\mathbf{x})$ that contains a constant number of labelings. In each time step $t+1$, the decoder can choose an *action* a_{t+1} from the *action space* A_{Y_t}; this space should be designed to be much smaller than the label space $\mathcal{Y}(\mathbf{x})$. Action a_{t+1} creates another labeling $\mathbf{y}_{t+1}$; this additional labeling creates successor state $Y_{t+1}(\mathbf{x}) = a_{t+1}(Y_t(\mathbf{x})) = Y_t(\mathbf{x}) \cup \{\mathbf{y}_{t+1}\}$.

In a basic definition, A_{Y_t} could consist of actions $\alpha_{\mathbf{y}j}$ (for all $\mathbf{y} \in Y_t$ and $1 \leq j \leq n_{\mathbf{x}}$, where $n_{\mathbf{x}}$ is the number of clients in $\mathbf{x}$) that take output $\mathbf{y} \in Y_t(\mathbf{x})$ and generate labeling $\bar{\mathbf{y}}$ by flipping the labeling of the j-th client; output $Y_{t+1}(\mathbf{x}) = Y_t(\mathbf{x}) \cup \{\bar{\mathbf{y}}\}$ is $Y_t(\mathbf{x})$ plus this modified output. This definition would allow the entire space $\mathcal{Y}(\mathbf{x})$ to be reached from any starting point. In our experiments, we will construct an action space that contains application-specific state transactions such as *flip the labels of the k addresses that have the most open connections*—see Section 7.3.

The choice of action a_{t+1} is based on parameters $\boldsymbol{\psi}$ of the decoder, and on a feature vector $\boldsymbol{\Psi}(\mathbf{x}, Y_t(\mathbf{x}), a_{t+1})$; for instance, actions may be chosen by following a stochastic policy $a_{t+1} \sim \pi_{\boldsymbol{\psi}}(\mathbf{x}, Y_t(\mathbf{x}))$. We will define feature vector $\boldsymbol{\Psi}(\mathbf{x}, Y_t(\mathbf{x}), a_{t+1})$ in Section 7.4.4; for instance, it contains the difference between the geographical distribution of clients whose label action a_{t+1} changes and the geographical distribution of all clients with that same label. Choosing an action a_{t+1} requires an evaluation of $\boldsymbol{\Psi}(\mathbf{x}, Y_t(\mathbf{x}), a_{t+1})$ for each possible action in $A_{Y_t(\mathbf{x})}$. Our problem setting is most useful for applications in which evaluation of $\boldsymbol{\Psi}(\mathbf{x}, Y_t(\mathbf{x}), a_{t+1})$ takes less time than evaluation of $\boldsymbol{\Phi}(\mathbf{x}, \mathbf{y}_{t+1})$—otherwise, it might be better to evaluate the decision function for a larger set of randomly drawn outputs than to spend time on selecting outputs for which the decision function should be evaluated. Feature vector $\boldsymbol{\Psi}(\mathbf{x}, Y_t(\mathbf{x}), a_{t+1})$ may contain a computationally inexpensive subset of $\boldsymbol{\Phi}(\mathbf{x}, \mathbf{y}_{t+1})$.

After T steps, the decoding process is terminated. At this point, the decision-function values $f_{\boldsymbol{\phi}}$ of a set of candidate outputs $Y_T(\mathbf{x})$ have been evaluated. Prediction $\hat{\mathbf{y}}$ is the argmax of the decision function over this set:

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y} \in Y_T(\mathbf{x})} f_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{y}). \tag{10}$$

At training time, a labeled sample $L = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$ is available.

6.2 HC Search

HC search [9] is an approach to structured prediction that learns parameters $\boldsymbol{\psi}$ of a search heuristic, and then uses a decoder with this search heuristic to learn parameters $\boldsymbol{\phi}$ of a structured-prediction model (the decision function $f_{\boldsymbol{\phi}}$ is called the cost-function in HC-search terminology). We apply this principle to our problem setting.

At application time, the decoder produces labeling $\hat{\mathbf{y}}$ that approximately maximizes $f_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{y})$ as follows. The starting point $Y_0(\mathbf{x})$ of each decoding problem contains the labeling produced by the *logistic regression* classifier (see Section 4.2). Action $a_{t+1} \in A_{Y_t}$ is chosen deterministically as the maximum of the search heuristic $f_{\boldsymbol{\psi}}(\boldsymbol{\Psi}(\mathbf{x}, Y_t(\mathbf{x}), a_{t+1})) = \boldsymbol{\psi}^\top \boldsymbol{\Psi}(\mathbf{x}, Y_t(\mathbf{x}), a_{t+1})$. After T steps, the argmax $\hat{\mathbf{y}}$ of $f_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{y})$ over all outputs in $Y_T(\mathbf{x}) = a_T(\ldots a_1(Y_0(\mathbf{x}))\ldots)$ (Equation 7) is returned as prediction.

At training time, *HC search* first learns a search heuristic with parameters $\boldsymbol{\psi}$ as follows. Let $L_{\boldsymbol{\psi}}$ be an initially empty set of training constraints for the heuristic. For each training instance $(\mathbf{x}_i, \mathbf{y}_i)$, starting state $Y_0(\mathbf{x}_i)$ contains the labeling produced by the *logistic regression* classifier (see Section 4.2). Time t is then iterated from 1 to an upper bound $\bar{T}$ on the number of time steps that will be available for decoding at application time. Then, iteratively, all elements a_{t+1} of the finite action space $A_{Y_t}(\mathbf{x}_i)$ and their corresponding outputs $\mathbf{y}'_{t+1}$ are enumerated and the action a^*_{t+1} that leads to the lowest-cost output $\mathbf{y}'_{t+1}$ is determined. Since the training data are labeled, the actual costs of labeling $\mathbf{x}_i$ as $\mathbf{y}'_{t+1}$ when the correct labeling would be $\mathbf{y}_i$ can be determined by evaluating the cost function. Search heuristic f_ψ has to assign a higher value to a^*_{t+1} than to any other a_{t+1}, and the costs $c(\mathbf{x}_i, \mathbf{y}_i, \mathbf{y}'_{t+1})$ of choosing a poor action should be included in the optimization problem. Hence, for each action $a_{t+1} \in A_{Y_t}(\mathbf{x}_i)$, constraint

$$\begin{aligned} f_\psi(\boldsymbol{\Psi}(\mathbf{x}_i, Y_t(\mathbf{x}_i), a^*_{t+1})) - f_\psi(\boldsymbol{\Psi}(\mathbf{x}_i, Y_t(\mathbf{x}_i), a_{t+1})) \\ > \sqrt{c(\mathbf{x}_i, \mathbf{y}_i, \mathbf{y}'_{t+1}) - c(\mathbf{x}_i, \mathbf{y}_i, \mathbf{y}^*_{t+1})} \end{aligned} \tag{11}$$

is added to $L_{\boldsymbol{\psi}}$. Model $\boldsymbol{\psi}$ should satisfy the constraints in $L_{\boldsymbol{\psi}}$. We use a soft-margin version of the constraints in $L_{\boldsymbol{\psi}}$ and squared slack-terms which results in a cost-sensitive multi-class SVM (actions a are the classes) with margin scaling [32].

After parameters $\boldsymbol{\psi}$ have been fixed, parameters $\boldsymbol{\phi}$ of structured-prediction model $f_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{y}) = \boldsymbol{\phi}^\top \boldsymbol{\Phi}(\mathbf{x}, \mathbf{y})$ are trained on the training data set of input-output pairs $(\mathbf{x}_i, \mathbf{y}_i)$ using SVM-struct with margin rescaling and using the search heuristic with parameters $\boldsymbol{\psi}$ as decoder. Negative pseudo-labels are generated as follows. For each $(\mathbf{x}_i, \mathbf{y}_i) \in L$, heuristic $\boldsymbol{\psi}$ is applied $\bar{T}$ times to produce a sequence of output sets $Y_0(\mathbf{x}_i), \ldots, Y_{\bar{T}}(\mathbf{x}_i)$. When $\bar{\mathbf{y}} = \text{argmax}_{\mathbf{y} \in Y_{\bar{T}}(\mathbf{x}_i)} \boldsymbol{\phi}^\top \boldsymbol{\Phi}(\mathbf{x}, \mathbf{y}) \neq \mathbf{y}_i$ violates the cost-rescaled margin, then a new training constraint is added, and parameters $\boldsymbol{\phi}$ are optimized to satisfy these constraints.

6.3 Online Policy-Gradient Decoder

The decoder of *HC search* has been trained to locate the labeling $\hat{\mathbf{y}}$ that minimizes the costs $c(\mathbf{x}_i, \mathbf{y}_i, \hat{\mathbf{y}})$ for given true labels. However, it is then applied to finding candidate labelings for which $f_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{y})$ is evaluated with the goal of maximizing $f_{\boldsymbol{\phi}}$. However, since the decision function $f_{\boldsymbol{\phi}}$ may be an imperfect approximation of the input-output relationship that is reflected in the training data, labelings that minimize the costs $c(\mathbf{x}_i, \mathbf{y}_i, \hat{\mathbf{y}})$ might be different from outputs that maximize the decision function. We will now derive a closed optimization problem in which decoder and structured-prediction model are jointly optimized. We will study its convergence properties theoretically.

We now demand that during the decoding process, the decoder chooses action $a_{t+1} \in A_{Y_t}$ which generates successor state $Y_{t+1}(\mathbf{x}) = a_{t+1}(Y_t(\mathbf{x}))$ according to a *stochastic* policy, $a_{t+1} \sim \pi_{\boldsymbol{\psi}}(\mathbf{x}, Y_t(\mathbf{x}))$, with parameter $\boldsymbol{\psi} \in \mathbb{R}^{m_2}$ (where m_2 is the dimensionality of the decoder feature space) and features $\boldsymbol{\Psi}(\mathbf{x}, Y_t(\mathbf{x}), a_{t+1})$. At time T, the prediction is the highest-scoring output from $Y_T(\mathbf{x})$ according to Equation 7.

The learning problem is to find parameters ϕ and ψ that minimize the expected costs over all inputs, outputs, and numbers of available decoding steps:

$$\underset{\phi,\psi}{\operatorname{argmin}}\, \mathbb{E}_{(\mathbf{x},\mathbf{y}),T,Y_T(\mathbf{x})}\left[c(\mathbf{x},\mathbf{y},\underset{\hat{\mathbf{y}}\in Y_T(\mathbf{x})}{\operatorname{argmax}} f_\phi(\mathbf{x},\hat{\mathbf{y}})\right] \tag{12}$$

$$\text{with } (\mathbf{x},\mathbf{y}) \sim p(\mathbf{x},\mathbf{y}), \quad T \sim p(T|\tau) \tag{13}$$

$$Y_T(\mathbf{x}) \sim p(Y_T(\mathbf{x})|\pi_\psi,\mathbf{x},T) \tag{14}$$

The costs $c(\mathbf{x},\mathbf{y},\hat{\mathbf{y}})$ of the highest-scoring element $\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y}'\in Y_T(\mathbf{x})} f_\phi(\mathbf{x},\mathbf{y}')$ may not be differentiable in ϕ. Let therefore loss $\ell(\mathbf{x},\mathbf{y},Y_T(\mathbf{x});\phi)$ be a differentiable approximation of the cost that ϕ induces on the set $Y_T(\mathbf{x})$. Section 7.2 instantiates the loss for the motivating problem. Distribution $p(\mathbf{x},\mathbf{y})$ is unknown. Given training data $S = \{(\mathbf{x}_1,\mathbf{y}_1),\ldots,(\mathbf{x}_m,\mathbf{y}_m)\}$, we approximate the expected costs (Equation 12) by the regularized expected empirical loss with convex regularizers Ω_ϕ and Ω_ψ:

$$\phi^*,\psi^* = \underset{\phi,\psi}{\operatorname{argmin}} \sum_{(\mathbf{x},\mathbf{y})\in S} V_{\phi,\psi,\tau}(\mathbf{x},\mathbf{y}) + \Omega_\phi + \Omega_\psi \tag{15}$$

$$\text{with } V_{\phi,\psi,\tau}(\mathbf{x},\mathbf{y}) = \sum_{T=1}^{\infty}\Big(p(T|\tau)\sum_{Y_T(\mathbf{x})} p(Y_T(\mathbf{x})|\pi_\psi,\mathbf{x},T)\ell(\mathbf{x},\mathbf{y},Y_T(\mathbf{x});\phi)\Big). \tag{16}$$

Equation 15 still cannot be solved immediately because it contains a sum over all values of T and all sets $Y_T(\mathbf{x})$. To solve Equation 15, we will liberally borrow ideas from the field of reinforcement learning. First, we will derive a formulation of the gradient $\nabla_{\psi,\phi}V_{\psi,\phi,\tau}(\mathbf{x},\mathbf{y})$. The gradient still involves an intractable sum over all sequences of actions, but its formulation suggests that it can be approximated by sampling action sequences according to the stochastic policy. By using a baseline function—which are a common tool in reinforcement learning [16]—we can reduce the variance of this sampling process.

Let $a_{1\ldots T} = a_1,\ldots,a_T$ with $a_{t+1} \in A_{Y_t}$ be a sequence of actions that executes a transition from $Y_0(\mathbf{x})$ to $Y_T(\mathbf{x}) = a_T(\ldots(a_1(Y_0(\mathbf{x})))\ldots)$. The available computation time is finite and hence $p(T|\tau) = 0$ for all $T > \bar{T}$ for some $\bar{T}$. We can rewrite Equation 16:

$$V_{\phi,\psi,\tau}(\mathbf{x},\mathbf{y}) = \sum_{a_{1\ldots\bar{T}}}\Big(p(a_{1\ldots\bar{T}}|\psi,Y_0(\mathbf{x}))\sum_{T=1}^{\bar{T}} p(T|\tau)\ell(\mathbf{x},\mathbf{y},a_T(\ldots(a_1(Y_0(\mathbf{x})))\ldots);\phi)\Big),$$

$$\text{with } \quad p(a_{1\ldots\bar{T}}|\psi,Y_0(\mathbf{x})) = \prod_{t=1}^{\bar{T}} \pi_\psi(a_t|\mathbf{x},a_{t-1}(\ldots(Y_0(\mathbf{x})\ldots)). \tag{17}$$

Equation 18 defines $D_{\ell,\tau}$ as the partial gradient ∇_ϕ of the expected empirical loss for an action sequence $a_1,\ldots,a_{\bar{T}}$ that has been sampled according to $p(a_{1,\ldots,\bar{T}}|\psi,Y_0(\mathbf{x}))$.

$$D_{\ell,\tau}(a_{1\ldots\bar{T}},Y_0(\mathbf{x});\phi) = \sum\nolimits_{T=1}^{\bar{T}} p(T|\tau)\nabla_\phi\ell(\mathbf{x},\mathbf{y},a_T(\ldots(a_1(Y_0(\mathbf{x})))\ldots);\phi) \tag{18}$$

The policy gradient ∇_ψ of a summand of Equation 17 is

$$\nabla_\psi p(a_{1\ldots\bar{T}}|\psi,Y_0(\mathbf{x}))\sum_{T=1}^{\bar{T}} p(T|\tau)\ell(\mathbf{x},\mathbf{y},a_T(\ldots(a_1(Y_0(\mathbf{x})))\ldots);\phi)$$

$$= \left(p(a_{1\ldots\bar{T}}|\boldsymbol{\psi}, Y_0(\mathbf{x})) \sum_{T=1}^{\bar{T}} \nabla_{\boldsymbol{\psi}} \log \pi_{\boldsymbol{\psi}}(a_T|\mathbf{x}, a_{T-1}(\ldots(a_1(Y_0(\mathbf{x}))\ldots))\right) \tag{19}$$
$$\sum_{T=1}^{\bar{T}} p(T|\tau)\ell(\mathbf{x}, \mathbf{y}, a_T(\ldots(a_1(Y_0(\mathbf{x})))\ldots); \boldsymbol{\phi}).$$

Equation 19 uses the "log trick" $\nabla_{\psi} p = p \log \nabla_{\psi} p$; it sums the gradients of all actions and scales with the accumulated loss of all initial subsequences. Baseline functions [16] reflect the intuition that a_T is not responsible for losses incurred prior to T; also, relating the loss to the expected loss for all sequences that contain a_T reflects the merit of a_T better. Equation 20 defines the policy gradient for an action sequence sampled according to $p(a_{1\ldots\bar{T}}|\boldsymbol{\psi}, Y_0(\mathbf{x}))$, modified by baseline function B.

$$E_{\ell,B,\tau}(a_{1\ldots\bar{T}}, Y_0(\mathbf{x}); \boldsymbol{\psi}, \boldsymbol{\phi})$$
$$= \sum_{T=1}^{\bar{T}} \nabla_{\boldsymbol{\psi}} \log \pi_{\boldsymbol{\psi}}(a_T|\mathbf{x}, a_{T-1}(\ldots(a_1(Y_0(\mathbf{x})))\ldots)) \tag{20}$$
$$\left(\sum_{t=T}^{\bar{T}} p(t|\tau)\ell(\mathbf{x}, \mathbf{y}, a_t(\ldots(a_1(Y_0(\mathbf{x})))\ldots); \boldsymbol{\phi}) - B(a_{1\ldots T-1}, Y_0(\mathbf{x}); \boldsymbol{\psi}, \boldsymbol{\phi}, \mathbf{x})\right)$$

Lemma 1 (General gradient) *Let $V_{\boldsymbol{\phi},\boldsymbol{\psi},\tau}(\mathbf{x}, \mathbf{y})$ be defined as in Equation 17 for a differentiable loss function ℓ. Let $D_{\ell,\tau}$ and $E_{\ell,B,\tau}$ be defined in Equations 18 and 20 for any scalar baseline function $B(a_{1\ldots T}, Y_0(\mathbf{x}); \boldsymbol{\psi}, \boldsymbol{\phi}, \mathbf{x})$. Then the gradient of $V_{\boldsymbol{\phi},\boldsymbol{\psi},\tau}(\mathbf{x}, \mathbf{y})$ is*

$$\nabla_{\boldsymbol{\phi},\boldsymbol{\psi}} V_{\boldsymbol{\phi},\boldsymbol{\psi},\tau}(\mathbf{x}, \mathbf{y}) = \sum_{a_{1\ldots\bar{T}}} p(a_{1\ldots\bar{T}}|\boldsymbol{\psi}, Y_0(\mathbf{x}))$$
$$\left[E_{\ell,B,\tau}(a_{1\ldots\bar{T}}, Y_0(\mathbf{x}); \boldsymbol{\psi}, \boldsymbol{\phi})^\top, D_{\ell,\tau}(a_{1\ldots\bar{T}}, Y_0(\mathbf{x}); \boldsymbol{\phi})^\top\right]^\top \tag{21}$$

Proof The gradient stacks the partial gradients of $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$ above each other. The partial gradient $\nabla_{\boldsymbol{\phi}} V_{\boldsymbol{\phi},\boldsymbol{\psi},\tau}(\mathbf{x}, \mathbf{y}) = \sum_{a_{1\ldots\bar{T}}} p(a_{1\ldots\bar{T}}|\boldsymbol{\psi}, Y_0(\mathbf{x})) D_{\ell,\tau}(a_{1\ldots\bar{T}}, Y_0(\mathbf{x}); \boldsymbol{\phi})$ follows from Equation 18. The partial gradient $\nabla_{\boldsymbol{\psi}} V_{\boldsymbol{\phi},\boldsymbol{\psi},\tau}(\mathbf{x}, \mathbf{y}) = \sum_{a_{1\ldots\bar{T}}} p(a_{1\ldots\bar{T}}|\boldsymbol{\psi}, Y_0(\mathbf{x})) E_{\ell,B,\tau}(a_{1\ldots\bar{T}}, Y_0(\mathbf{x}); \boldsymbol{\psi}, \boldsymbol{\phi})$ is a direct application of the Policy Gradient Theorem [31,27] for episodic processes.

The choice of a baseline function B influences the variance of the sampling process, but not the gradient; a lower variance means faster convergence. Let $E_{\ell,B,\tau,T}$ be a summand of Equation 20 with a value of T. Variance $\mathbb{E}[(E_{\ell,B,\tau,T}(a_{1\ldots\bar{T}}, Y_0(\mathbf{x}); \boldsymbol{\psi}, \boldsymbol{\phi}) - \mathbb{E}[E_{\ell,B,\tau,T}(a_{1\ldots\bar{T}}, Y_0(\mathbf{x}); \boldsymbol{\psi}, \boldsymbol{\phi})|a_{1..T}])^2|a_{1..T}]$ is minimized by the baseline that weights the loss of all sequences starting in $a_T(\ldots(a_1(Y_0(\mathbf{x})))\ldots)$ by the squared gradient [16]:

$$B_{\mathrm{G}}(a_{1\ldots T}, Y_0(\mathbf{x}); \boldsymbol{\psi}, \boldsymbol{\phi}, \mathbf{x}) = \frac{\sum_{a_{T+1}} G(a_{1..T+1}, Y_0)^2 Q(a_{1..T+1}, Y_0)}{\sum_{a_{T+1}} G(a_{1..T+1}, Y_0)^2} \tag{22}$$

$$\text{with } Q(a_{1..T+1}, Y_0) = \mathop{\mathbb{E}}_{a_{T+2\ldots\bar{T}}}\left[\sum_{t=T+1}^{\bar{T}} p(t|\tau)\ell(\mathbf{x}, \mathbf{y}, a_t(\ldots(Y_0(\mathbf{x}))\ldots); \boldsymbol{\phi}) \middle| a_{1..T+1}\right] \tag{23}$$

$$\text{and } G(a_{1..T+1}, Y_0) = \nabla \log \pi_{\psi}(a_{T+1}|\mathbf{x}, a_T(\ldots(a_1(Y_0(\mathbf{x})))\ldots)). \tag{24}$$

This baseline function is intractable because it (intractably) averages the loss of all action sequences that start in state $Y_T(\mathbf{x}) = a_T(\dots a_1(Y_0(\mathbf{x}))\dots)$ with the squared length of the gradient of their first action a_T. Instead, the assumption that the expected loss of all sequences starting at T is half the loss of state $Y_T(\mathbf{x})$ gives the approximation:

$$B_{\mathrm{HL}}(a_{1\dots T}, Y_0(\mathbf{x}); \boldsymbol{\psi}, \boldsymbol{\phi}, \mathbf{x}) = \frac{1}{2} \sum\nolimits_{t=T+1}^{\bar{T}} p(t)\ell(\mathbf{x}, \mathbf{y}, a_T(...a_1(Y_0(\mathbf{x}))...); \boldsymbol{\phi}). \quad (25)$$

We will refer to the policy-gradient method with baseline function B_{HL} as *online policy gradient with baseline*. Note that inserting baseline function

$$B_{\mathrm{R}}(a_{1\dots T}, Y_0(\mathbf{x}); \boldsymbol{\psi}, \boldsymbol{\phi}, \mathbf{x}) = -\sum_{t=1}^{T} p(t)\ell(\mathbf{x}, \mathbf{y}, a_t(\dots(a_1(Y_0(\mathbf{x})))\dots); \boldsymbol{\phi}) \quad (26)$$

into Equation 20 resolves each summand of Equation 21 to Equation 19, the unmodified policy gradient for $a_{1\dots,\bar{T}}$. We will refer to the online policy-gradient method with baseline function B_{R} as *online policy gradient without baseline*. Algorithm 1 shows the *online policy-gradient* learning algorithm. It optimizes parameters $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$ using a stochastic gradient by sampling action sequences from the intractable sum over all action sequences of Equation 21 Theorem 1 proves its convergence under a number of conditions. The step size parameters $\alpha(i)$ have to satisfy

$$\sum\nolimits_{i=0}^{\infty} \alpha(i) = \infty \ , \sum\nolimits_{i=0}^{\infty} \alpha(i)^2 < \infty. \quad (27)$$

Loss function ℓ is required to be bounded. This can be achieved by constructing the loss function such that for large values it smoothly approaches some arbitrarily high ceiling C. However, in our case study we could not observe cases in which the algorithm does not converge for unbounded loss functions. Baseline function B is required to be differentiable and bounded for the next theorem. However, no gradient has to be computed in the algorithm. All baseline functions that are considered in Section 7 meet this demand.

Algorithm 1 Online Stochastic Policy-Gradient Learning Algorithm

```
Input: Training data $S$, starting parameters $\boldsymbol{\psi}_0, \boldsymbol{\phi}_0$.
1: let $i = 0$.
2: **repeat**
3:    Draw $(\mathbf{x}, \mathbf{y})$ uniformly from $S$
4:    Sample action sequence $a_{1\dots\bar{T}}$ with each $a_t \sim \pi_{\psi_i}(\mathbf{x}, a_{t-1}(\dots a_1(Y_0(\mathbf{x}))\dots))$.
5:    $\boldsymbol{\psi}_{i+1} = \boldsymbol{\psi}_i - \alpha(i)(E_{\ell,B}(a_{1\dots\bar{T}}, Y_0(\mathbf{x}); \boldsymbol{\psi}_i, \boldsymbol{\phi}_i) + \nabla_{\boldsymbol{\psi}}\Omega_{\boldsymbol{\psi}_i})$
6:    $\boldsymbol{\phi}_{i+1} = \boldsymbol{\phi}_i - \alpha(i)(D_{\ell,\tau}(a_{1\dots\bar{T}}, Y_0(\mathbf{x}); \boldsymbol{\psi}_i) + \nabla_{\boldsymbol{\phi}}\Omega_{\boldsymbol{\phi}_i})$
7:    increment $i$.
8: **until** convergence
Return $\boldsymbol{\psi}_i, \boldsymbol{\phi}_i$
```

Theorem 1 (Convergence of Algorithm 1) *Let the stochastic policy $\pi_{\boldsymbol{\psi}}$ be twice differentiable, let both $\pi_{\boldsymbol{\psi}}$ and $\nabla_{\boldsymbol{\psi}}\pi_{\boldsymbol{\psi}}$ be Lipschitz continuous, and let $\nabla_{\boldsymbol{\psi}} \log \pi_{\boldsymbol{\psi}}$ be bounded. Let step size parameters $\alpha(i)$ satisfy Equation 27. Let loss function ℓ be differentiable in $\boldsymbol{\phi}$ and both ℓ and $\nabla_{\boldsymbol{\phi}}\ell$ be Lipschitz continuous. Let ℓ be bounded. Let B be differentiable and both B and $\nabla_{\boldsymbol{\phi}\boldsymbol{\psi}}B$ be bounded. Let $\Omega_{\boldsymbol{\phi}} = \gamma_1\|\boldsymbol{\phi}\|^2$,$\Omega_{\boldsymbol{\psi}} = \gamma_2\|\boldsymbol{\psi}\|^2$. Then, Algorithm 1 converges with probability 1.*

Proof For space limitations and in order to improve readability, throughout the proof we omit dependencies on $\mathbf{x}$ and $Y_0(\mathbf{x})$ in the notations when dependence is clear from the context. For example, we use $p(a_{1..\bar{T}}|\boldsymbol{\psi})$ instead of $p(a_{1..\bar{T}}|\boldsymbol{\psi}, Y_0(\mathbf{x}))$. We use Theorem 2 from Chapter 2 and Theorem 7 from Chapter 3 of [6] to prove convergence. We first show that the full negative gradient $-\sum_{(\mathbf{x},\mathbf{y})} \nabla_{\boldsymbol{\psi},\boldsymbol{\phi}} V_{\boldsymbol{\psi}_i,\boldsymbol{\phi}_i,\tau}(\mathbf{x},\mathbf{y}) - [\gamma_2 \boldsymbol{\psi}_i^\top, \gamma_1 \boldsymbol{\phi}_i^\top]^\top$ is Lipschitz continuous.

Let $L(a_{T..\bar{T}}, \boldsymbol{\phi}) = \sum_{t=T}^{\bar{T}} p(t)\ell(\mathbf{x},\mathbf{y},a_t(..Y_0(\mathbf{x})..);\boldsymbol{\phi})$. We proceed by showing that $p(a_{1..\bar{T}}|\boldsymbol{\psi})E_{\ell,B,\tau}(a_{1..\bar{T}};\boldsymbol{\psi},\boldsymbol{\phi}) = \sum_{T=1}^{\bar{T}}(p(a_{1..\bar{T}}|\boldsymbol{\psi})\ \nabla_{\boldsymbol{\psi}} \log \pi_{\boldsymbol{\psi}}(a_T|\boldsymbol{\psi})(L(a_{T..\bar{T}},\boldsymbol{\phi}) - B(a_{1..T-1},\boldsymbol{\psi},\boldsymbol{\phi})))$ is Lipschitz in $[\boldsymbol{\psi}^\top, \boldsymbol{\phi}^\top]^\top$. It is differentiable in $[\boldsymbol{\psi}^\top, \boldsymbol{\phi}^\top]^\top$ per definition and it suffices to show that the derivative is bounded. By the product rule,

$$
\begin{aligned}
&\nabla_{\boldsymbol{\psi},\boldsymbol{\phi}}(p(a_{1..\bar{T}}|\boldsymbol{\psi})\nabla_{\boldsymbol{\psi}} \log \pi_{\boldsymbol{\psi}}(a_T|\boldsymbol{\psi})(L(a_{T..\bar{T}},\boldsymbol{\phi}) - B(a_{1..T-1},\boldsymbol{\psi},\boldsymbol{\phi})))\\
=&\nabla_{\boldsymbol{\psi},\boldsymbol{\phi}}(p(a_{1..\bar{T}}|\boldsymbol{\psi})\nabla_{\boldsymbol{\psi}} \log \pi_{\boldsymbol{\psi}}(a_T|\boldsymbol{\psi}))(L(a_{T..\bar{T}},\boldsymbol{\phi}) - B(a_{1..T-1},\boldsymbol{\psi},\boldsymbol{\phi})) \qquad (28)\\
&+ p(a_{1..\bar{T}}|\boldsymbol{\psi})\nabla_{\boldsymbol{\psi}} \log \pi_{\boldsymbol{\psi}}(a_T|\boldsymbol{\psi})\nabla_{\boldsymbol{\psi},\boldsymbol{\phi}}(L(a_{T..\bar{T}},\boldsymbol{\phi}) - B(a_{1..T-1},\boldsymbol{\psi},\boldsymbol{\phi})). \qquad (29)
\end{aligned}
$$

We can see that line 29 is bounded because p, $\nabla_{\boldsymbol{\psi}} \log \pi_{\psi}$, $\nabla_{\boldsymbol{\phi}} L$ and $\nabla_{\boldsymbol{\phi},\boldsymbol{\psi}} B$ are bounded by definition and products of bounded functions are bounded. Regarding line 28, we state that L and B are bounded by definition. Without loss of generality, let $T = 1$.

$$
\begin{aligned}
&\nabla_{\boldsymbol{\psi}}(p(a_{1..\bar{T}}|\boldsymbol{\psi})\nabla_{\boldsymbol{\psi}} \log \pi_{\boldsymbol{\psi}}(a_1|\boldsymbol{\psi}))\\
&= \nabla_{\boldsymbol{\psi}}(\nabla_{\boldsymbol{\psi}}\pi_{\boldsymbol{\psi}}(a_1|\boldsymbol{\psi})p(a_{2..\bar{T}}|a_1\boldsymbol{\psi})) \qquad (30)\\
&= p(a_{2..\bar{T}}|a_1\boldsymbol{\psi}))\nabla_{\boldsymbol{\psi}}\nabla_{\boldsymbol{\psi}}\pi_{\boldsymbol{\psi}}(a_1|\boldsymbol{\psi}) + \nabla_{\boldsymbol{\psi}}\pi_{\boldsymbol{\psi}}(a_1|\boldsymbol{\psi})\nabla_{\boldsymbol{\psi}}p(a_{2..\bar{T}}|a_1,\boldsymbol{\psi})) \qquad (31)
\end{aligned}
$$

Equation 30 follows from $p\nabla_{\psi} \log p = \nabla \boldsymbol{\psi} p$. The left summand of Equation 31 is bounded because both p and $\nabla_{\boldsymbol{\psi}}\nabla_{\boldsymbol{\psi}}\pi_{\boldsymbol{\psi}}$ are bounded by definition. Furthermore, $\nabla_{\boldsymbol{\psi}} p(a_{2..\bar{T}}|\boldsymbol{\psi}) = \nabla_{\boldsymbol{\psi}}\pi_{\boldsymbol{\psi}}(a_2)p(a_{3..\bar{T}}|\boldsymbol{\psi}) + \pi_{\boldsymbol{\psi}}(a_2)\nabla_{\boldsymbol{\psi}}p(a_{3..\bar{T}}|\boldsymbol{\psi})$ is bounded because $\nabla_{\boldsymbol{\psi}}\pi_{\boldsymbol{\psi}}(a_t)$ and $p(a_{t..\bar{T}}|\boldsymbol{\psi})$ are bounded for all t and we can expand $\nabla_{\boldsymbol{\psi}}p(a_{3..\bar{T}}|\boldsymbol{\psi})$ recursively. From this it follows that the right summand of Equatio 31 is bounded as well. Thus we have shown the above claim.

$p(a_{1..\bar{T}}|\boldsymbol{\psi})D_{\ell,\tau}(a_{1..\bar{T}};\boldsymbol{\phi})$ is Lipschitz because $p(a_{1..\bar{T}}|\boldsymbol{\psi})$ is Lipschitz and bounded and $D_{\ell,\tau}$ is a sum of bounded Lipschitz functions. The product of two bounded Lipschitz functions is bounded. $[\gamma_1\boldsymbol{\psi}^\top, \gamma_2\boldsymbol{\phi}^\top]^\top$ is obviously Lipschitz as well, which concludes the considerations regarding the full negative gradient.

Let $M_{i+1} = [E_{\ell,B,\tau}(a_{1..\bar{T}};\boldsymbol{\psi}_i,\boldsymbol{\phi}_i)^\top, D_{\ell,\tau}(a_{1..\bar{T}};\boldsymbol{\psi}_i)^\top]^\top - \sum_{(\mathbf{x},\mathbf{y})} \nabla_{\boldsymbol{\psi},\boldsymbol{\phi}} V_{\boldsymbol{\psi}_I,\boldsymbol{\phi}_i,\tau}(\mathbf{x},\mathbf{y}))$, where $E_{\ell,B,\tau}(a_{1..\bar{T}};\boldsymbol{\psi}_i,\boldsymbol{\phi}_i)$ and $D_{\ell,\tau}(a_{1..\bar{T}};\boldsymbol{\psi}_i)$ are samples as computed by Algorithm 1. We show that $\{M_i\}$ is a Martingale difference sequence with respect to the increasing family of σ-fields $\mathcal{F}_i = \sigma([\boldsymbol{\phi}_0^\top, \boldsymbol{\psi}_0^\top]^\top, M_1, ..., M_i), i \geq 0$. That is, $\forall i \in \mathbb{N}$, $\mathbb{E}[M_{i+1}|\mathcal{F}_i] = 0$ *almost surely*, and $\{M_i\}$ are square-integrable with $\mathbb{E}[||M_{i+1}||^2|\mathcal{F}_i] \leq K(1 + ||[\boldsymbol{\phi}_i^\top, \boldsymbol{\psi}_i^\top]^\top||^2)$ *almost surely*, for some $K > 0$.

$\mathbb{E}[M_{i+1}|\mathcal{F}_n] = 0$ is given by the definition of M_{i+1} above. We have to show $\mathbb{E}[||M_{i+1}||^2|\mathcal{F}_i] \leq K(1 + ||[\boldsymbol{\phi}_i^\top, \boldsymbol{\psi}_i^\top]^\top||^2)$ for some K. We proceed by showing that for each $(\mathbf{x},\mathbf{y},a_{1..\bar{T}})$ it holds that $||[E_{\ell,B,\tau}(a_{1..\bar{T}};\boldsymbol{\psi}_i,\boldsymbol{\phi}_i)^\top, D_{\ell,\tau}(a_{1..\bar{T}};\boldsymbol{\psi}_i)^\top]^\top||^2 \leq K(1 + ||[\boldsymbol{\phi}_i^\top, \boldsymbol{\psi}_i^\top]^\top||^2)$. From that it follows that

$$
\begin{aligned}
||\sum_{\mathbf{x},\mathbf{y}} \sum_{a_{1..\bar{T}}} p(a_{1..\bar{T}}|\boldsymbol{\psi})[E_{\ell,B,\tau}(a_{1..\bar{T}};\boldsymbol{\psi},\boldsymbol{\phi})^\top, D_{\ell,\tau}(a_{1..\bar{T}};\boldsymbol{\phi})^\top]^\top||^2&\\
\leq K(1 + ||[\boldsymbol{\phi}_i^\top, \boldsymbol{\psi}_i^\top]^\top||^2)&
\end{aligned}
$$

and $||M_{i+1}||^2 \leq 4K(1 + ||[\boldsymbol{\phi}_i^\top, \boldsymbol{\psi}_i^\top]^\top||^2)$ which proves the claim.

Regarding $E_{\ell,B,\tau}$, we assume that $\|\nabla \log \pi_{\boldsymbol{\psi}}\|^2$ is bounded by some K'' and it follows that $\| \sum_{T=1}^{\bar{T}} \nabla_{\boldsymbol{\psi}} \log \pi_{\boldsymbol{\psi}}(a_T|\mathbf{x}, a_{T-1}(..Y_0(\mathbf{x})..)\|^2$ is also bounded by $\bar{T}^2 K''$. $\|\ell(\mathbf{x}, \mathbf{y}, a_T(..Y_0(\mathbf{x})..); \boldsymbol{\phi})\|^2 \leq K'(1 + \|\boldsymbol{\phi}\|^2)$ and B bounded per assumption and thus $\sum_{t=T}^{\bar{T}} p(t)\ell(\mathbf{x}, \mathbf{y}, a_t(..Y_0(\mathbf{x})..)(\mathbf{x}); \boldsymbol{\phi}) - B(a_{1..T-1}; \boldsymbol{\phi}, \boldsymbol{\psi}) \leq \bar{K}'(1 + \|\boldsymbol{\phi}\|^2)$ with some $\bar{K}'$. It follows that $\|E_{\ell,B,\tau}(a_{1..\bar{T}}; \boldsymbol{\psi}_i, \boldsymbol{\phi}_i)\|^2 \leq 2^{\bar{T}} K'' \bar{K}'(1 + \|\boldsymbol{\phi}\|^2)$. As $\nabla_{\boldsymbol{\phi}} \ell(\mathbf{x}, \mathbf{y}, a_T(..Y_0(\mathbf{x})..); \boldsymbol{\phi})$ is bounded per assumption, $\|D_{\ell,\tau}\|^2 \leq K'''$ for some $K''' > 0$. The claim follows: $\|[E_{\ell,B,\tau}(a_{1..\bar{T}}; \boldsymbol{\psi}_i, \boldsymbol{\phi}_i)^\top, D_{\ell,\tau}(a_{1..\bar{T}}; \boldsymbol{\psi}_i)^\top]^\top\|^2 = \|E_{\ell,B,\tau}(a_{1..\bar{T}}; \boldsymbol{\psi}_i, \boldsymbol{\phi}_i)\|^2 + \|D_{\ell,\tau}(a_{1..\bar{T}}; \boldsymbol{\psi}_i)\|^2 \leq K''' + 2^{\bar{T}} K'' \bar{K}'(1 + \|\boldsymbol{\phi}\|^2) \leq K''' + \bar{T}^2 K'' \bar{K}'(1 + \|[\boldsymbol{\psi}^\top, \boldsymbol{\phi}^\top]^\top\|^2)$.

We can now use Theorem 2 from Chapter 2 of [6] to prove convergence by identifying function $h([\boldsymbol{\phi}_i^\top, \boldsymbol{\psi}_i^\top]^\top)$ as assumed in the assumptions of that theorem with the full negative gradient $-\sum_{(\mathbf{x},\mathbf{y})} \nabla_{\boldsymbol{\psi},\boldsymbol{\phi}} V_{\boldsymbol{\psi}_i,\boldsymbol{\phi}_i,\tau}(\mathbf{x}, \mathbf{y}) - \left[\gamma_2 \boldsymbol{\psi}_i^\top, \gamma_1 \boldsymbol{\phi}_i^\top\right]^\top$. The theorem states that the algorithm converges with probability 1 if the iterates $[\boldsymbol{\phi}_{i+1}^\top, \boldsymbol{\psi}_{i+1}^\top]^\top$ stay bounded.

Now, let $h_r(\xi) = h(r\xi)/r$. Next, we show that $\lim_{r\to\infty} h_r(\xi) = h_\infty(\xi)$ exists and that the origin in $\mathbb{R}^{m_1+m_2}$ is an asymptotically stable equilibrium for the o.d.e. $\dot{\xi}(t) = h_\infty(\xi(t))$. With this, Theorem 7 from Chapter 3 of [6]—originally from [7]—states that the iterates stay bounded and Algorithm 1 converges. Next, we show that h meets (A4):

$$
\begin{aligned}
&h_r(\boldsymbol{\phi}, \boldsymbol{\psi}) \\
&\quad = \frac{1}{r} \sum_{\mathbf{x},\mathbf{y}} \sum_{a_{1..\bar{T}}} p(a_{1..\bar{T}}|r\boldsymbol{\psi}) \left[E_{\ell,B,\tau}(a_{1..\bar{T}}; r\boldsymbol{\psi}, r\boldsymbol{\phi})^\top, D_{\ell,\tau}(a_{1..\bar{T}}; r\boldsymbol{\phi})^\top\right]^\top \\
&\qquad + 1/r\left[\gamma_1 r\boldsymbol{\psi}_i^\top, \gamma_2 r\boldsymbol{\phi}_i^\top\right]^\top \\
&\quad = \sum_{\mathbf{x},\mathbf{y}} \sum_{a_{1..\bar{T}}} p(a_{1..\bar{T}}|r\boldsymbol{\psi}) \sum_{T=1}^{\bar{T}} \left[\nabla_{\boldsymbol{\psi}} \pi_{\boldsymbol{\psi}}(a_T|r\boldsymbol{\psi})^\top (L(a_{T..\bar{T}}, r\boldsymbol{\phi}) - B(a_{1..T-1}))/r,\right. \\
&p(a_{1..\bar{T}}|r\boldsymbol{\psi}) D_{\ell,\tau}(a_{1..\bar{T}}; r\boldsymbol{\phi})^\top/r\big]^\top + \left[\gamma_1 \boldsymbol{\psi}_i^\top, \gamma_2 \boldsymbol{\phi}_i^\top\right]^\top,
\end{aligned}
$$

$\nabla_{\boldsymbol{\psi}} \pi_{\boldsymbol{\psi}}, L$ and B are all bounded and it follows that $p(a_{1..\bar{T}}|r\boldsymbol{\psi}) \sum_{T=1}^{\bar{T}} \left[\nabla_{\boldsymbol{\psi}} \pi_{\boldsymbol{\psi}}(a_T|r\boldsymbol{\psi})^\top (L(a_{T..\bar{T}}, r\boldsymbol{\phi}) - B(a_{1..T-1}))/r \rightarrow 0\right.$. The same holds for the other part as $p(a_{1..\bar{T}}|r\boldsymbol{\psi})$ and $D_{\ell,\tau}(a_{1..\bar{T}}; r\boldsymbol{\phi})$ are bounded. It follows that $h_\infty([\boldsymbol{\psi}^\top, \boldsymbol{\phi}^\top]^\top) = \left[\gamma_1 \boldsymbol{\psi}_i^\top, \gamma_2 \boldsymbol{\phi}_i^\top\right]^\top$. Therefore, the ordinary differential equation $\dot{\xi}(t) = h_\infty(\xi(t))$ has an asymptotically stable equilibrium at the origin, which shows that (A4) is valid.

7 Identification of DDoS Attackers

We will now implement a DDoS-attacker detection mechanism using the techniques that we derived in the previous sections. We engineer a cost function, suitable feature representations $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$, policy π_ψ, and loss function ℓ that meet the demands of Theorem 1.

7.1 Cost Function

False-positive decisions (legitimate clients that are mistaken for attackers) lead to the temporary blacklisting of a legitimate user. This will result in unserved requests, and potentially lost business. False-negative decisions (attackers that are not recognized as such) will result in a wasteful allocation of server resources, and possibly in a successful DDoS attack that leaves the service unavailable for legitimate users. We decompose cost function $c(\mathbf{x}, \mathbf{y}, \hat{\mathbf{y}})$ for a set of clients $\mathbf{x}$ into the following parts.

We measure two cost-inducing parameters of false-negative decisions: the *number of connections* opened by attacking clients and the *CPU use* triggered by clients' requests. According to the experience of the data-providing web hosting service, the same damage is done by attacking clients that a) collectively initiate 200 connections per 10-seconds interval t and b) collectively initiate scripts that use 10 CPUs for 10 seconds. However, those costs are not linear in their respective attributes. Instead, only limited resources are available, such as a finite number of CPUs, and the rise in costs of two scripts that use 80% or 90%, resp., of all available CPUs is different from the rise in costs of two scripts that use 20% or 30% of CPUs. We define costs incurred by connections initiated by attackers to be quadratic in the number of connections. Similarly, costs for CPU usage are also quadratic.

The hosting service assesses that blocking a legitimate client incurs the same cost as opening 200 HTTP connections to attackers in an interval or wasting 100 CPU seconds. Also, by blocking 50 connections of legitimate client the same cost is added. Based on these requirements, we define costs

$$\begin{aligned} c(\mathbf{x}, \mathbf{y}, \hat{\mathbf{y}}) = &\sum_{x_i: y_i=-1, \hat{y}_i=+1} 1 + \frac{1}{50} \times \text{\#connections by } x_i + \left(\frac{1}{200} \sum_{x_i: y_i=+1, \hat{y}_i=-1} \text{\#connections by } x_i\right)^2 \\ &+ \left(\frac{1}{100} \sum_{x_i: y_i=+1, \hat{y}_i=-1} \text{CPU seconds initiated by } x_i\right)^2. \end{aligned}$$

7.2 Loss Function

In order for the *online policy-gradient* method to converge, Theorem 1 states that loss functions ℓ need to be differentiable and both ℓ and $\nabla\ell$ have to be Lipschitz continuous. We discuss loss functions in this section. As mentioned in Section 6.3, the boundedness assumption on loss functions can be enforced by smoothly transitioning the loss function to a function that approaches some arbitrarily high ceiling C. We first define the difference in costs of a prediction $\hat{\mathbf{y}}$ and an optimal label $\mathbf{y}^*$ as $\rho(\hat{\mathbf{y}}, \mathbf{y}^*) = c(\mathbf{x}, \mathbf{y}, \hat{\mathbf{y}}) - c(\mathbf{x}, \mathbf{y}, \mathbf{y}^*)$. We denote the margin as $g_{\mathbf{x},\mathbf{y}}(\mathbf{y}^*, \hat{\mathbf{y}}; \boldsymbol{\phi}) = \sqrt{\rho(\hat{\mathbf{y}}, \mathbf{y}^*)} - \boldsymbol{\phi}^\top(\boldsymbol{\Phi}(\mathbf{x}, \hat{\mathbf{y}}) - \boldsymbol{\Phi}(\mathbf{x}, \mathbf{y}^*))$. The clipped squared hinge loss is differentiable:

$$h_{\mathbf{x},\mathbf{y}}(\mathbf{y}^*, \hat{\mathbf{y}}; \boldsymbol{\phi}) = \begin{cases} 0 & \text{if } \sqrt{\rho(\hat{\mathbf{y}}, \mathbf{y}^*)} \leq \boldsymbol{\phi}^\top(\boldsymbol{\Phi}(\mathbf{x}, \hat{\mathbf{y}}) - \boldsymbol{\Phi}(\mathbf{x}, \mathbf{y}^*)) \\ g_{\mathbf{x},\mathbf{y}}(\mathbf{y}^*, \hat{\mathbf{y}}; \boldsymbol{\phi})^2 & \text{if } 0 < \boldsymbol{\phi}^\top(\boldsymbol{\Phi}(\mathbf{x}, \hat{\mathbf{y}}) - \boldsymbol{\Phi}(\mathbf{x}, \mathbf{y}^*)) < \sqrt{\rho(\hat{\mathbf{y}}, \mathbf{y}^*)} \\ 2\boldsymbol{\phi}^\top(\boldsymbol{\Phi}(\mathbf{x}, \hat{\mathbf{y}}) - \boldsymbol{\Phi}(\mathbf{x}, \mathbf{y}^*)) + 2 & \text{if } \boldsymbol{\phi}^\top(\boldsymbol{\Phi}(\mathbf{x}, \hat{\mathbf{y}}) - \boldsymbol{\Phi}(\mathbf{x}, \mathbf{y}^*)) \leq 0 \end{cases}$$

Equation 32 defines the loss that $\boldsymbol{\phi}$ induces on $Y_T(\mathbf{x})$ as the average squared hinge loss of all labels in $Y_T(\mathbf{x})$ except the one with minimal costs, offset by these minimal costs.

$$\ell_h(\mathbf{x}, \mathbf{y}, Y_T(\mathbf{x}); \boldsymbol{\phi}) = c(\mathbf{x}, \mathbf{y}, \mathbf{y}^*) + \frac{1}{|Y_T(\mathbf{x}) - 1|} \sum_{\hat{\mathbf{y}} \in Y_T(\mathbf{x}), \hat{\mathbf{y}} \neq \mathbf{y}^*} h_{\mathbf{x},\mathbf{y}}(\mathbf{y}^*, \hat{\mathbf{y}}; \boldsymbol{\phi}) \quad (32)$$
$$\text{with } \mathbf{y}^* = \operatorname*{argmin}_{\hat{\mathbf{y}} \in Y_T} c(\mathbf{x}, \mathbf{y}, \hat{\mathbf{y}})$$

In contrast to the standard squared margin-rescaling loss for structured prediction that uses the hinge loss of the output that maximally violates the margin, here we average the Huber loss over all labels in $Y_T(\mathbf{x})$; this definition of ℓ_h is differentiable and Lipschitz continuous, as required by Theorem 1. *Online policy gradient* employs loss function ℓ_h in our experimentations. We will refer to *HC search* with loss function ℓ_h as *HC search with average margin* and will also conduct experiments with *HC search with max margin* by using the standard squared margin-rescaled loss

$$\ell_m(\mathbf{x}, \mathbf{y}, Y_T(\mathbf{x}), \boldsymbol{\phi}) = \max_{\hat{\mathbf{y}} \in Y_T(\mathbf{x}), \hat{\mathbf{y}} \neq \mathbf{y}^*} \max\{g_{\mathbf{x},\mathbf{y}}(\mathbf{y}^*, \hat{\mathbf{y}}; \boldsymbol{\phi}), 0\}^2. \qquad (33)$$

7.3 Action Space and Stochastic Policy

This section defines the action space $A_{Y_i}(\mathbf{x}_i)$ of *HC search* and the *online policy-gradient* method as well as the stochastic policy $\pi_{\boldsymbol{\psi}}(\mathbf{x}, Y_t(\mathbf{x}))$ of *online policy-gradient*.

The action space is based on 21 rules $r \in R$ that can be instantiated for the elements $\mathbf{y} \in Y_t(\mathbf{x})$; the action space A_{Y_t} contains all instantiations $a_{t+1} = (r, \mathbf{y})$ that add a new labeling $r(\mathbf{y})$ to the successor state: $Y_{t+1}(\mathbf{x}) = Y_t(\mathbf{x}) \cup \{r(\mathbf{y})\}$. We define the initial set Y_0 to contain labels $\{-1\}^{n_\mathbf{x}}$ and $\{+1\}^{n_\mathbf{x}}$, where $n_\mathbf{x}$ is the number of clients in $\mathbf{x}$. Some of the following rules refer to the score of a binary classifier that classifies clients independently; we use the *logistic regression* regression classifier as described in Section 4.2 in our experiments.

- Switch the labels of the 1, 2, 5, or 10 clients from -1 to $+1$ that have the highest number of connections, the highest score of the baseline classifier, or CPU consumption. All combinations of these attributes yield 12 possible rules.
- Switch the labels of the client from -1 to 1 that has the second-highest number of connections, independent classifier score, or CPU consumption (3 rules).
- Switch the label of the client from 1 to -1 that has the lowest or second-lowest number of connections, baseline classifier score, or CPU consumption (6 rules).
- Switch all clients from -1 to $+1$ whose independent classifier score exceeds -1, -0.5, 0, 0.5, or 1 (5 rules).

Theorem 1 requires that the stochastic policy be twice differentiable in $\boldsymbol{\psi}$ and that both $\pi_{\boldsymbol{\psi}}$ and $\nabla_{\boldsymbol{\psi}}\pi_{\boldsymbol{\psi}}$be Lipschitz continuous. We define $\pi_{\boldsymbol{\psi}}$ as

$$\pi_{\boldsymbol{\psi}}(a_{t+1}|\mathbf{x}, Y_t(\mathbf{x})) = \frac{\exp(\boldsymbol{\psi}^\top \boldsymbol{\Psi}(\mathbf{x}, Y_t(\mathbf{x}), a_{t+1}))}{\sum_{a \in A_{Y_t}} \exp(\boldsymbol{\psi}^\top \boldsymbol{\Psi}(\mathbf{x}, Y_t(\mathbf{x}), a))}.$$

7.4 Feature Representations

We engineer features that refer to base traffic parameters that we explain in Section 7.4.1. From these base traffic parameters, we derive feature representations for all learning approaches that we study. Figure 1 gives an overview of all features.

7.4.1 Base Traffic Parameters

In each 10-seconds interval, we calculate base traffic parameters of each client that connects to the domain. For clients that connect to the domain over a longer duration, we calculate moving averages that are reset after two minutes of inactivity. On the TCP protocol level, we extract the absolute numbers of full connections, open connections, open and resent FIN

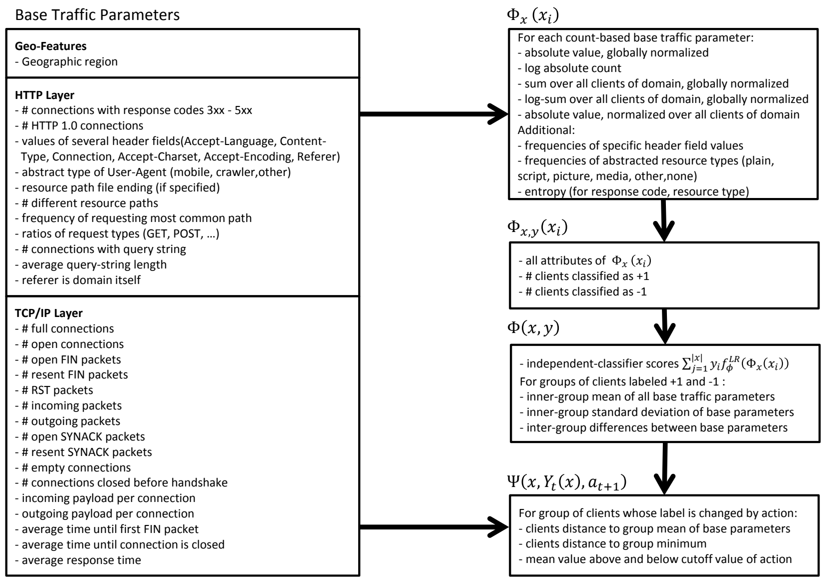

Base Traffic Parameters
Geo-Features
- Geographic region
HTTP Layer
- # connections with response codes 3xx - 5xx
- # HTTP 1.0 connections
- values of several header fields(Accept-Language, Content-Type, Connection, Accept-Charset, Accept-Encoding, Referer)
- abstract type of User-Agent (mobile, crawler,other)
- resource path file ending (if specified)
- # different resource paths
- frequency of requesting most common path
- ratios of request types (GET, POST, …)
- # connections with query string
- average query-string length
- referer is domain itself
TCP/IP Layer
- # full connections
- # open connections
- # open FIN packets
- # resent FIN packets
- # RST packets
- # incoming packets
- # outgoing packets
- # open SYNACK packets
- # resent SYNACK packets
- # empty connections
- # connections closed before handshake
- incoming payload per connection
- outgoing payload per connection
- average time until first FIN packet
- average time until connection is closed
- average response time
$\Phi_x(x_i)$
For each count-based base traffic parameter:
- absolute value, globally normalized
- log absolute count
- sum over all clients of domain, globally normalized
- log-sum over all clients of domain, globally normalized
- absolute value, normalized over all clients of domain
Additional:
- frequencies of specific header field values
- frequencies of abstracted resource types (plain, script, picture, media, other,none)
- entropy (for response code, resource type)
$\Phi_{x,y}(x_i)$
- all attributes of $\Phi_x(x_i)$
- # clients classified as +1
- # clients classified as -1
$\Phi(x,y)$
- independent-classifier scores $\sum_{j=1}^{|x|} y_i f_\phi^{LR}(\Phi_x(x_i))$
For groups of clients labeled +1 and -1 :
- inner-group mean of all base traffic parameters
- inner-group standard deviation of base parameters
- inter-group differences between base parameters
$\Psi(x, Y_t(x), a_{t+1})$
For group of clients whose label is changed by action:
- clients distance to group mean of base parameters
- clients distance to group minimum
- mean value above and below cutoff value of action

Fig. 1 Feature representations

packets, timeouts, RST packets, incoming and outgoing packets, open and resent SYNACK packets, empty connections, connections that are closed before the handshake is completed, incoming and outgoing payload per connection. We determine the average durations until the first FIN packet is received and until the connection is closed, as well as the response time.

From the HTTP protocol layer, we extract the number of connections with HTTP response status codes 3xx, 4xx, and 5xx, the absolute counts of HTTP 1.0 connections and of the values of several HTTP header fields (Accept-Language, Content-Type, Connection, Accept-Charset, Accept-Encoding, Referer). We also extract User-Agent and define *mobile* and *crawler* which count all occurrences of a predefined set of known mobile user agents (Android and others) and crawlers (GoogleBot and others), respectively.

We count the number of different resource paths that a client accesses and also count how often each client requests the currently most common path on the domain. If a specific resource is directly accessed we extract and categorize the file ending into *plain, script, picture, download, media, other, none*, which can give a hint on the type of the requested resource. We measure the fractions of request types per connection (*GET, POST*, or *OTHER*). We extract the number of connections with a query string and the average length of each query in terms of number of fields per client. We count the number of connection in which the referrer is the domain itself. Geographic locations are encoded in terms of 21 parameters that represent a geographic region.

7.4.2 Input Features for SVDD, Logistic Regression and ICA

Independent classification uses features $\boldsymbol{\Phi}_{\mathbf{x}}(x_j)$ that refer to a particular client x_j and to the entirety of all clients $\mathbf{x}$ that interact with the domain. For each of the count-style base traffic parameters, $\boldsymbol{\Phi}_{\mathbf{x}}(x_j)$ contains the absolute value, globally normalized over all clients of all domains, a logarithmic absolute count, the globally normalized sums and log-sums over all clients that interact with the domain, and the absolute values, normalized by the values of all clients that interact with the domain. For HTTP response code, resource type header fields, we also determine the entropy and frequencies per client on for all clients on the domain. See also Fig 1.

Feature vector $\mathbf{\Phi}_{\mathbf{x},\mathbf{y}}(x_j)$ for ICA contains all features from $\mathbf{\Phi}_{\mathbf{x}}(x_j)$ plus the numbers of clients that are assigned class $+1$ and -1, respectively, in $\mathbf{x}, \mathbf{y}$.

7.4.3 Features for Structured Prediction

Feature vector $\mathbf{\Phi}(\mathbf{x}, \mathbf{y})$ contains as one feature the sum $\sum_{j=1}^{|\mathbf{x}|} y_j f_{\boldsymbol{\phi}}^{LR}(\mathbf{\Phi}_{\mathbf{x}}(x_j))$ of scores of a previously trained *logistic regression* classifier over all clients $x_j \in \mathbf{x}$. In addition, we distinguish between the groups of clients that $\mathbf{y}$ labels as -1 and $+1$ and determine the inner-group means, inner-group standard deviations, inter-group differences of the base traffic parameters. This results in a total of 297 features.

7.4.4 Decoder Features

For *HC search* and *online policy gradient*, the parametric decoders depend on a joint feature representation $\mathbf{\Psi}(\mathbf{x}, Y_t(\mathbf{x}), a_{t+1})$ of input $\mathbf{x}$ and action $a_{t+1} = (r, \mathbf{y})$. It contains 92 joint features of the clients whose label a_{t+1} changes and the group (clients of positive or negative class) that a_{t+1} assigns the clients to. Features include the clients' distance to the group mean and the clients' distance to the group minimum for the base traffic parameters. For the fourth group of control actions, the feature representation includes the mean values of these same base attributes for all clients above and below the cutoff value. In order to save computation time, the mean and minimal group values before reassigning the clients are copied from $\mathbf{\Phi}(\mathbf{x}, \mathbf{y})$ which must have been calculated previously.

7.4.5 Execution-Time Constraint

We model distribution $p(T|\tau)$ that limits the number of time steps that are available for *HC search* and *online policy gradient* as a beta distribution with $\alpha = 5$ and $\beta = 3$ that is capped at a maximum value of $\bar{T} = 10$. We allow *ICA* to iterate over all instances for five times; the results do not improve after that. The execution time of *logistic regression* is negligible and therefore unconstrained.

8 Experimental Study

This section explores the practical benefit of all methods for attacker detection.

8.1 Data Collection

In order to both train and evaluate the attacker-detection models, we collect a data set of TCP/IP traffic from the application environment. We focus our data collection on high-traffic events in which a domain *might* be under attack. When the number of connections to a domain per unit of time, the number of clients that interact with the domain, and the CPU capacity used by a domain lie below safe lower bounds, we can rule out the possibility of a DDoS attack. Throughout an observation period of several days, we store all HTTP traffic to any domain for which a traffic threshold is exceeded starting 10 minutes before the threshold is exceeded and stopping 10 minutes after no threshold is exceeded any longer. We will refer to the entirety of traffic to a particular domain that occurs during one of these episodes as an *event*. Over our observation period, we collect 1,546 events. We record all traffic parameters described in Section 7.4. All data of one domain that are recorded within a time slot of 10 seconds are stored as a block. The same threshold-based pre-filtering is applied in the operational system, and therefore our data collection reflects the distribution which the attacker-detection system is exposed to in practice.

We then label all traffic events as *attacks* or *legitimate traffic* and all clients as *attackers* or *legitimate clients* in a largely manual process. In a joint effort with experienced administrators, we decide for each of the 1,546 unusual event whether it is in fact a flooding attack. For this, we employ several tools and information sources. We search for known vulnerabilities in the domain's scripts, analyze the domain's recent regular connection patterns, check for unusual geo-location patterns and analyze the query strings and HTTP header fields. This labeling task is inherently difficult. On one hand, repeated queries by several clients that lead to the execution of a CPU-heavy script with either identical or random parameters might very likely indicate an attack. On the other hand, when a resource is linked to by a high-traffic web site and that resource is delivered via a computationally expensive script, the resulting traffic may look very similar to traffic observed during an attack and one has to search for and check the referrer for plausibility to identify the traffic as legitimate.

After having labeled all events, we label individual clients that connect to a domain during an attack event. We use several heuristics to group clients with a nearly identical and potentially malicious behavior and label them jointly by hand. We subsequently label the remaining clients after individual inspection.

In total, 50 of the 1,546 events are actually attacks with 10,799 unique attackers. A total of 448,825 client IP addresses are labeled as legitimate. In order to reduce memory and storage usage we use a sample from all 10-second intervals that were labeled. We draw 25% of intervals per attack and 10% of intervals (but at least 5 if the event is long enough) per non-attack event. Our final data set consists of 1,096,196 labeled data points; each data point is a client that interacts with a domain within one of the 22,645 intervals of 10 seconds.

8.2 Experimental Setting

Our data includes 50 attack events; we therefore run 50-fold stratified cross validation with one attack event per fold. Since the attack durations vary, the number of test instances varies between folds. We determine the costs of all methods as the average costs over the 50 folds. In each fold, we reserve 20% of the training portion to tune the hyperparameters of all models by a grid search.

8.3 Reference Methods

All previous studies on detecting and mitigating application-layer DDoS flooding attacks are based on anomaly-detection methods [37,28,36,8,22]. A great variety of heuristic and principled approaches is used. In our study, we represent this family of approaches by *SVDD* which has been used successfully for several related computer-security problems [12,15]. Prior work generally uses smaller feature sets. Since we have not been able to improve our anomaly-detection or classification results by feature subset selection, we refrain from

Table 1 Costs, true-positive rates, and false-positive rates of all attacker-detection models. Costs marked with "$*$" are significantly lower than the costs of *logistic regression*

Classification Method	Mean costs per fold	TPR	FPR ($\times 10^{-4}$)
No filtering	3.363 ± 1.348	0	0
SVDD	2.826 ± 1.049	0.121 ± 0.036	149.8 ± 89.5
Log. Reg. w/o domain-dependent features	1.322 ± 0.948	0.394 ± 0.056	7.0 ± 2.1
Logistic Regression	1.045 ± 0.715	0.372 ± 0.056	2.1 ± 0.6
ICA	$0.946 \pm 0.662*$	0.369 ± 0.056	3.2 ± 1.0
HC search with average margin	1.042 ± 0.715	0.406 ± 0.056	9.1 ± 4.2
HC search with max-margin	$1.040 \pm 0.714*$	0.398 ± 0.056	7.0 ± 3.3
Policy gradient with baseline function	$0.945 \pm 0.664*$	0.394 ± 0.055	3.7 ± 1.2
Policy gradient without baseline function	$0.947 \pm 0.665*$	0.394 ± 0.055	3.7 ± 1.2

conducting experiments with the specific feature subsets that are used in published prior work.

Some prior work uses features or inference methods that cannot be applied in our application environment. DDosShield [28] calculates an *attack suspicion score* by measuring a client's deviation from inter-arrival times and session workload profiles of regular traffic. Monitoring workload profiles is not possible in our case because the attacker-detection system is running on a different machine; it cannot monitor the workload profiles of the large number of host computers whose traffic it monitors. DDosShield also uses a scheduler and prioritizes requests by suspicion score. This approach is also not feasible in our application environment because it still requires all incoming requests to be processed (possibly by returning an error code). Xie and Yu [36] also follow the anomaly-detection principle. They employ a hidden Markov model whose state space is the number of individual web pages. In our application environment, both the number of clients and of hosted individual pages are huge and prohibit state inference for of each individual client.

8.4 Results

Table 1 shows the costs, true-positive rates, and false-positive rates of all methods under investigation. All methods reduce the costs that are incurred by DDoS attacks substantially at low false-positive rates. *SVDD* reduces the costs of DDoS attacks compared to not employing any attacker-detection mechanism (*no filtering*) by about 16%. *Logistic regression* reduces the costs of DDoS attacks compared (*no filtering*) by about 69%; *online policy gradient* reduces the costs by 72%. Differences between *no filtering*, *SVDD*, and *logistic regression* are highly significant. Cost values marked with an asterisk star ("$*$") are significantly lower than logistic regression in a paired t-test at $p < 0.1$. While *HC search* is only marginally (insignificantly) better than *logistic regression*, all other structured-prediction models improve upon *logistic regression*. *Policy gradient with baseline function* incurs marginally lower costs than *policy gradient without baseline function* and *ICA*, but the differences are not significant.

Logistic regression w/o domain-dependent features does not get access to features that take into account all other clients of that domain and to the entropy features. This shows that engineering context features into the feature representation of independent classification already leads to much of the benefit of structured prediction. From a practical point of view, all classification methods are useful, reduce the costs associated with DDoS attacks by around 70% while misclassifying only an acceptable proportion (below 10^{-3}) of legitimate

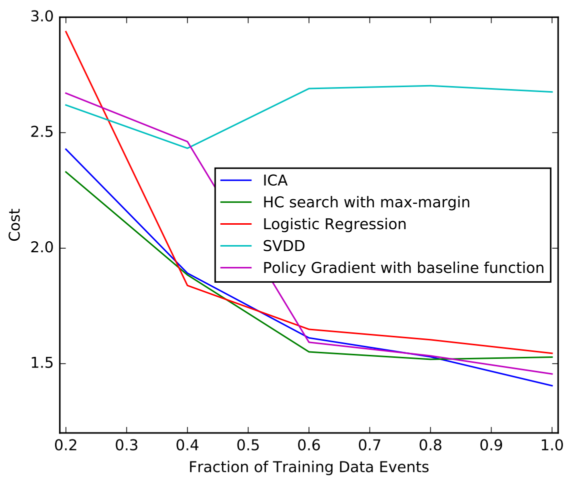


Fig. 2 Learning curves over varying fractions of training events.

clients. We conclude that *ICA* and *policy gradient* achieve a small additional cost reduction over independent classification of clients.

8.5 Analysis

In this section, we quantitatively explore which factors contribute to the residual costs of structured prediction models. The overall costs incurred by *policy gradient* decompose into costs that are incurred because f_ϕ fails to select the best labeling from the decoding set $Y_T(\mathbf{x})$, and costs that are incurred because decoder π_ψ approximates an exhaustive search by a very narrow and directed search that is biased by ψ.

We conduct an experiment in which decoder π_ψ is learned on training data, and a perfect decision function f^*_ϕ is passed down by way of divine inspiration. To this end, we learn π_ψ on training data, use it to construct decoding sets $Y_T(\mathbf{x}_i)$ for the test instances, and identify the elements $\hat{\mathbf{y}} = \text{argmin}_{\mathbf{y} \in Y_T(\mathbf{x}_i)} c(\mathbf{x}_i, \mathbf{y}_i, \mathbf{y})$ that have the smallest true costs; note that this is only possible because the true label $\mathbf{y}_i$ is known for the test instances. We observe costs of 0.012 ± 0.008 for the perfect decision function, compared to costs of 0.945 ± 0.664 when ϕ is learned on training data. The costs of a perfect decoder that exhaustively searches the space of all labelings, in combination with perfect decision function f^*_ϕ, would be zero. This implies that the decoder with learned parameters ψ performs almost as well as an (intractable) exhaustive search; it contributes only 1.3% of the total costs whereas 98.7% of the costs are due to the imperfection of f_ϕ. Increasing the decoding time T does not change these results.

This leaves parameter uncertainty of ϕ caused by limited labeled training data and the definition of the model space as possible sources the residual costs. We conduct a learning curve analysis to explore how decreasing parameter uncertainty decreases the costs. We determine costs for various fractions of training *events* using 10-fold cross validation in Figure 2. We use 10-fold cross validation in order to make sure that each test fold contains at least one attack event when reducing the number of events to 0.2. Since Table 1 uses 50-fold cross validation (which results in a higher number of training events), the end points of Figure 2 are not directly comparable to the values in Table 1. Figure 2 shows that the costs of

Table 2 Most relevant features of f_ϕ

Weight	Description
3.01	Average length of query strings of client
-2.38	Number of different resource paths of client
2.34	Sum of incoming payload of all clients of domain
2.27	Fraction of connections of client that request the most frequent resource path
2.25	Sum of response times of all clients of domain
2.05	Sum of response times of client
1.64	Fraction of connections for domain that accepts any version of English (e.g., en-us) in Accept-Language
-1.46	Entropy of request type (GET/POST/OTHER)
-1.32	Sum of outgoing payload of all clients
1.27	Sum of number of open FINs of all clients at end of 10-seconds interval
1.23	Average length of query string per connection
-1.21	Fraction of connections for domain that accepts any language other than EN, DE, ES, PT, CN, RU in Accept-Language
1.19	Fraction of all connections of all clients that query most frequent path
1.17	Sum of durations of all connections of all clients of domain
1.13	Fraction of connections of client that accepts any version of English (e.g., en-us) in Accept-Language
-1.13	Fraction of combined connections of all clients that directly request a picture type
-1.11	Fraction of connections of client that specified HTTP header field Content-Type as any text variant
-1.09	Fraction of connections of client that accepts any language other than EN, DE, ES, PT, CN, RU in Accept-Language
1.08	Log-normalized combined outgoing payload of client
-1.07	Fraction of all connections of all clients that specified HTTP header field Content-Type as any text variant

all classification methods continue to decrease with an increasing number of training events. A massively larger number of training events would be required to estimate the convergence point. We conclude that parameter uncertainty of ϕ is the dominating source of costs of all classification models. Anomaly-detection method *SVDD* only requires unlabeled data that can be recorded in abundance. Interestingly, *SVDD* does not appear to benefit from a larger sample. This matches our subjective perception of the data: HTTP traffic rarely follows a "natural" distribution; anomalies are ubiquitous, but most of the time they are not caused by attacks.

8.6 Feature Relevance

For the independent classification model, leaving out features that take into account all clients that connect to the domain deteriorates the performance (see Line 3 of Table 1. We have not been able to eliminate any particular group of features by feature subset selection without deteriorating the system performance. Table 2 shows the most relevant features; that is, the features that have the highest average weights (over 50-fold cross validation) in the *logistic regression* model.

8.7 Execution Time

In our implementation, the step of extracting features $\mathbf{\Phi}$ takes on average 1 ms per domain for *logistic regression* and *ICA*. The additional calculations take about 0.03 ms for *logistic* *regression* and 0.04 ms for *ICA* with five iterations over the nodes which results in nearly identical total execution times of 1.03 and 1.04 ms, respectively.

HC search and *online policy gradient* start with an execution of *logistic regression*. For $T = 10$ decoding steps, repeated calculations of $\mathbf{\Phi}(\mathbf{x}, \mathbf{y})$ and $\mathbf{\Psi}(\mathbf{x}, Y_t(\mathbf{x}), a)$ lead to a total execution time of 3.1 ms per domain in a high-traffic event.

9 Discussion and Related Work

Mechanisms that merely *detect DDoS attacks* still leave it to an operator to take action. Methods for *detecting malicious HTTP requests* can potentially prevent SQL-injection and cross-site scripting attacks, but their potential to mitigate DDoS flooding attacks is limited, because all incoming HTTP requests still have to be accepted and processed. Defending against network-level DDoS attacks [26,37] is a related problem; but since network-layer attacks are not protocol-compliant, better detection and mitigation mechanisms (*e.g.,* adaptive timeout thresholds, ingress/egress filtering) are available.

Since known detection mechanisms against network-level DDoS attacks are fairly effective in practice, our study focuses on application-level attacks—specifically, on HTTP-level flooding attacks. Prior work on *defending against application-level DDoS attacks* has focused on detecting anomalies in the behavior of clients over time [28,36,22,8]. Clients that deviate from a model of legitimate traffic are trusted less and less, and the rate at which their requests are processed is throttled. Trust-based and throttling approaches leave it necessary to accept incoming HTTP requests, maintain records of all connecting clients, and process the requests—possibly by returning an error code instead of the requested result. In our application environment, this would not sufficiently relieve the servers. Prior work on defending against application-level DDoS attacks have so far been evaluated using artificial or semi-artificial traffic data that have been generated under model assumptions of benign and offending traffic. This paper presents the first large-scale empirical study based on over 1,500 high-traffic events that we detected while monitoring several hundred thousand domains over several days.

Detection of *DDoS attacks* and *malicious HTTP requests* have been modeled as anomaly detection and classification problems. Anomaly detection mechanisms employ a model of legitimate network traffic [36]—and treat unlikely traffic patterns as attacks. For the detection of SQL-injection, cross-site-scripting (XSS), and PHP file-inclusion (L/RFI), traffic can be modeled based on HTTP header and query string information using HMMs [5], n-gram models [35], general kernels [12], or other models [29]. Anomaly-detection mechanisms were investigated, from centroid anomaly-detection models [18] to setting hard thresholds on the likelihood of new HTTP requests given the model, to unsupervised learning of support-vector data description (SVDD) models [12,15].

Classification-based models require traffic data to be labeled; this gives classification methods an information advantage over anomaly-detection models. In practice, network traffic rarely follows predictable patterns. Spikes in popularity, misconfigured scripts, and crawlers create traffic patterns that resemble those of attacks; this challenges anomaly-detection approaches. Also, in shared hosting environments domains appear and disappear on a regular basis, making the definition of normal traffic even more challenging. A binary SVM trained on labeled data has been observed to consistently outperform a one-class SVM using n-gram features [35]. Similarly, augmenting SVDDs with labeled data has been observed to greatly improve detection accuracy [15]. Other work has studied SVMs [17,21] and other classification methods [19,25,14].

Structured-prediction algorithms jointly predict the values of multiple dependent output variables—in this case, labels for all clients that interact with a domain—for a (structured) input [20, 32, 2]. At application time, structured-prediction models have to find the highest-scoring output during the decoding step. For sequential and tree-structured data, the highest-scoring output can be identified by dynamic programming. For fully connected graphs, exact inference of the highest-scoring output is generally intractable. Many approaches to approximate inference have been developed; for instance, for CRFs [1], structured SVMs [13], and general graphical models [3]. Several algorithmic schemes are based on iterating over the nodes and changing individual class labels locally. The iterative classification algorithm [24] for collective classification simplistically classifies individual nodes, given the conjectured labels of all neighboring nodes, and reiterates until this process reaches a fixed points.

Online policy-gradient is the first method that optimizes the parameters of the structured-prediction model and the decoder in a joint optimization problem. This allows us to prove its convergence for suitable loss functions. By contrast, *HC search* [9, 10] first learns a search heuristic that guides the search to the correct labeling for the training data, and subsequently learns the decision function of a structured-prediction model using this search heuristic as a decoder. Shi et al. [30] follow a complementary approach by first training a probabilistic structured model, and then using reinforcement learning to learn a decoder.

Wick et al. [34] sample structured outputs using a predefined, hand-crafted proposer function that samples outputs sequentially. In other work [33] a cascade of Markov models is learned that uses increasing higher-order features and prunes unlikely local outputs per cascade level. This work assumes a ordering of such cliques into levels, which is not applicable for fully connected graphs.

10 Conclusion

We have engineered mechanisms for detection of DDoS attackers based on anomaly detection, independent classification of clients, collective classification of clients, and structured-prediction with *HC search*. We have then developed the *online policy-gradient* method that learns a decision function and a stochastic policy which controls the decoding process in an integrated optimization problem. We have shown that this method is guaranteed to converge for appropriate loss functions. From our empirical study that is based on a large, manually-labeled collection of HTTP traffic with 1,546 high-traffic events we can draw three main conclusions. (a) All classification approaches outperform the anomaly-detection method *SVDD* substantially. (b) From a practical point of view, even the most basic *logistic regression* model is useful and reduces the costs by 69% at a false-positive rate of 2.1×10^{-4}. (c) *ICA* and *online policy gradient* reduce the costs just slightly further, to about 72%.

Acknowledgment

This work was supported by grant SCHE540/12-2 of the German Science Foundation DFG and by a grant from STRATO AG.

References

1. Discriminative probabilistic models for relational data. In *Eighteenth Conference on Uncertainty in Artificial Intelligence*, 2002.

2. Max-margin Markov networks. In *Advances in Neural Information Processing Systems*, volume 16, 2004.
3. Approximated structured prediction for learning large scale graphical models. Arxiv 1006.2899, 2010.
4. C. Amza, E. Cecchet, A. Chanda, A. Cox, S. Elnikety, R. Gil, J. Marguerite, K. Rajamani, and W. Zwaenepoel. Bottleneck characterization of dynamic web site benchmarks. Technical report TR-02-391, Rice University, 2002.
5. Davide Ariu, Roberto Tronci, and Giorgio Giacinto. HMMPayl: An intrusion detection system based on hidden Markov models. *Computers & Security*, 30(4):221–241, 2011.
6. Vivek S. Borkar. *Stochastic approximation: A Dynamical Systems Viewpoint*. Cambridge University Press, 2008.
7. Vivek S. Borkar and Sean P. Meyn. The ODE method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal on Control and Optimization*, 38(2):447–469, 2000.
8. S. Renuka Devi and P. Yogesh. Detection of application layer DDsS attacks using information theory based metrics. Department of Information Science and Technology, College of Engineering Guindy 10.5121/csit.2012.2223, 2012.
9. Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. HC-search: Learning heuristics and cost functions for structured prediction. In *AAAI*, volume 2, page 4, 2013.
10. Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. HC-search: A learning framework for search-based structured prediction. *Journal of Artificial Intelligence Research*, 50(1):369–407, 2014.
11. Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. Structured prediction via output space search. *The Journal of Machine Learning Research*, 15(1):1317–1350, 2014.
12. Patrick Düssel, Christian Gehl, Pavel Laskov, and Konrad Rieck. Incorporation of application layer protocol syntax into anomaly detection. In *Information Systems Security*, pages 188–202. Springer, 2008.
13. T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *Proceedings of the International Conference on Machine Learning*, 2008.
14. Farnaz Gharibian and Ali A Ghorbani. Comparative study of supervised machine learning techniques for intrusion detection. In *Annual Conference on Communication Networks and Services Research*, pages 350–358. IEEE, 2007.
15. Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. Toward supervised anomaly detection. *Journal of Artificial Intelligence Research*, 46:235–262, 2013.
16. Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *The Journal of Machine Learning Research*, 5:1471–1530, 2004.
17. Latifur Khan, Mamoun Awad, and Bhavani Thuraisingham. A new intrusion detection system using support vector machines and hierarchical clustering. *International Journal on Very Large Databases*, 16(4):507–521, 2007.
18. Marius Kloft and Pavel Laskov. Security analysis of online centroid anomaly detection. *Journal of Machine Learning Research*, 13(1):3681–3724, 2012.
19. Levent Koc, Thomas A Mazzuchi, and Shahram Sarkani. A network intrusion detection system based on a hidden naïve Bayes multiclass classifier. *Expert Systems with Applications*, 39(18):13492–13500, 2012.
20. John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, 2001.
21. Yinhui Li, Jingbo Xia, Silan Zhang, Jiakai Yan, Xiaochuan Ai, and Kuobin Dai. An efficient intrusion detection system based on support vector machines and gradually feature removal method. *Expert Systems with Applications*, 39(1):424–430, 2012.
22. H. Liu and K. Chang. Defending systems against tilt DDoS attacks. In *Proceedings of the International Conference on Telecommunication Systems, Services, and Applications*, 2011.
23. Luke K. McDowell, Kalyan Moy Gupta, and David W. Aha. Cautious collective classification. *The Journal of Machine Learning Research*, 10:2777–2836, 2009.
24. Jennifer Neville and David Jensen. Iterative classification in relational data. In *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, 2000.
25. Sandhya Peddabachigari, Ajith Abraham, Crina Grosan, and Johnson Thomas. Modeling intrusion detection system using hybrid intelligent systems. *Journal of Network and Computer Applications*, 30(1):114–132, 2007.
26. Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM Computing Surveys*, 39(1):3, 2007.
27. Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

28. S. Ranjan, R. Swaminathan, M. Uysal, and E. Knightley. DDoS-resilient scheduling to counter application layer attacks under imperfect detection. In *Proceedings of IEEE INFOCOM*, 2006.
29. William K. Robertson and Federico Maggi. Effective anomaly detection with scarce training data. In *Network and Distributed System Security Symposium*, 2010.
30. Tianlin Shi, Jacob Steinhardt, and Percy Liang. Learning where to sample in structured prediction. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 875–884, 2015.
31. Richard S. Sutton, David Mcallester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *In Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press, 2000.
32. I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.
33. David Weiss and Ben Taskar. Structured prediction cascades. In *International Conference on Artificial Intelligence and Statistics*, pages 916–923, 2010.
34. Michael Wick, Khashayar Rohanimanesh, Kedar Bellare, Aron Culotta, and Andrew McCallum. Samplerank: Training factor graphs with atomic gradients. In *Proceedings of the 28th International Conference on Machine Learning*, pages 777–784, 2011.
35. Christian Wressnegger, Guido Schwenk, Daniel Arp, and Konrad Rieck. A close look on n-grams in intrusion detection: Anomaly detection vs. classification. In *Proceedings of the ACM Workshop on Artificial Intelligence and Security*, pages 67–76, 2013.
36. Y. Xie and S. Z. Yu. A large-scale hidden semi-markov model for anomaly detection on user browsing behaviors. *IEEE/ACM Transactions on Networking*, 17(1):54–65, 2009.
37. Saman Taghavi Zargar, James Joshi, and David Tipper. A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. *IEEE Communications Surveys & Tutorials*, 15(4):2046–2069, 2013.

Chapter 5

Discussion

Each presented paper advances the state of the art in its specific domain of internet security applications as well as in its respective field of machine learning research. In the following, I will categorize contributions made by each paper into one of three categories. The next section embeds each paper into related work covering its respective field of internet security. By doing so, I highlight their contributions to tackling their respective internet security application. Section 5.2 shows advancements in the way the respective application can be modeled as a machine learning problem and in terms of derived optimization problems and theoretical guarantees of algorithms. The last section concludes by describing how each of the unique large scale real world data sets was assembled and how it was used to experimentally evaluate improvements of the developed methods over baseline methods.

5.1 Application-Specific Related Work

In this section I embed each of the three applications covered in this thesis into their respective literature on an application level. To this effect, the section discusses related work regarding the practical problems of inbound and outbound email spam filtering as well as DDoS attack detection and mitigation.

Research on defense mechanisms against incoming email spam has a long history in which the development and application of machine learning algorithms played a prominent part from the early beginnings. Generally, a wide range of methods should be combined to protect email infrastructure and customer inboxes against flooding of unsolicited bulk email. From rather simple but effective IP blacklists such as Spamhaus SBL[1], to spam campaign detection mechanisms using Bayesian clustering of the incoming email stream [Haider 09], and regular expression matching on spam campaigns [Prasse 15]. However, from the very beginnings of automated spam detection content-based binary classification algorithms were used to decide for spam. A wide range of algorithms and feature representations were investigated in the literature, see e.g. [Blanzieri 09] for an overview. In order to learn good spam filters, a variety of data sets of spam messages were published, such as those corpora used in the TREC spam track[2]. However, using only public corpora will not result in

[1]https://www.spamhaus.org/sbl/

[2]http://trec.nist.gov/data/spam.html

state-of-the-art performance of spam filters, both because spam senders have access to those data sets and because spam emails change heavily over time [Wang 13]. Best results may be achieved when combining both in-house data sets and external data sets, a scenario we investigated in chapter 2.

Although filtering outbound emails for spam has been identified as a priority by email service providers [Goodman 04], literature on this specific problem has not been as manifold as for filtering incoming emails. Standard content-based spam filters could be employed—e.g. if clients try to send emails via a HTTP webmail interface—and could be combined with some reverse Touring test mechanism such as CAPTCHAS. A different approach was taken by [Goodman 04], who employed content-agnostic rate limits for users. This mechanism did not prove to be very effective, though. [Zhong 05] presented an ad-hoc adaptive rate-limiting approach that proposed to limit the outgoing email rate based on an estimated spam score. While using a throttling mechanism might intuitively be appealing, only in [Dick 10] (Chapter 3) did we show that the approach of learning an adaptive rate-limit follows directly from the properly defined problem definition.

Work on the detection of distributed denial of service (DDoS) attacks is manifold and can to be partitioned into research that aims at detecting attacks versus papers that investigate detecting attackers. In chapter 4 we investigated approaches that were employed against HTTP-level attacks, in contrast to network level attacks where IPs could be spoofed in some cases [Peng 07, Zargar 13]. If attack detection mechanisms are employed in a server infrastructure, attack mitigation mechanisms still have to take appropriate countermeasures against attacking clients in order to mitigate the effect of an attack. A large selection of binary classification algorithms has been used to detect attacks on the HTTP level, using a diverse set of features [Ariu 11, Wressnegger 13, Düssel 08, Robertson 10]. As large data sets of attacks and regular traffic are often hard to acquire and label, and also because it is sometimes argued that potential attack vectors can not be modeled or recorded exhaustively, anomaly detection methods were also used to detect DDoS attacks [Düssel 08, Görnitz 13]. The same holds for prior work on *defending against* application-level DDoS attacks which often focused on detecting anomalies in the behavior of clients over time [Ranjan 06, Xie 09, Liu 11, Devi 12]. In [Dick 16], we focused on detecting attackers using discriminative models that employ labeled data from both attackers and regular clients for training classifiers. More importantly, we investigated the usefulness of structured prediction models for DDoS attacker identification that consider dependencies between labels of individual clients.

5.2 Application-Oriented Modeling and Algorithms

In this section I discuss how each of the papers presented in chapters 2, 3, and 4 relate to prior work in their respective field of machine learning research. I first describe how a learning problem is deducted from the demands of each application and then compare the derived solutions to related literature.

5.2.1 Inbound Spam Filtering with Incomplete Data

In [Dick 08] (Chapter 2), an email spam classifier was learned from data that was compiled from different data sources. All features for computing the decisions are present when the model is evaluated in the email provider's email infrastructure, but training instances coming from external data sources do not include some of the features such as those representing sender reputation. Learning from incomplete data has been studied extensively in the machine learning literature. One of the first approaches was to estimate one imputation per missing value, e.g. its mean over the observed instances or of the k-nearest neighbors, and to subsequently learn with the completed data using standard learning algorithms [Little 87]. Other approaches only consider the available features without modeling a distribution over imputations. [Chechik 08] uses this approach for applications where features are not *missing at random* but are structurally missing. They argue that those features are non-existent rather than missing. They maximize the margin of a prediction model based on visible features only. A third group estimates a distribution of imputations in a first step before learning the decision function based on the estimated distribution of imputations [Shivaswamy 06, Smola 05]. A final group estimates both a distribution of imputations and the decision function jointly [Liao 07, Wang 10]. In chapter 2 we followed the same approach of learning a distribution of imputations and decision function jointly but allowed for more flexibility of the solution—a nonparametric distribution of missing values was estimated that is only biased towards a distributional assumption by a regularizer. [Wang 10] also learn imputations and decision function simultaneously by using a Mixture of Experts model formulation with linear local experts. Features are assumed to be drawn according to a mixture of local Gaussian distributions. The number of experts is inferred using a nonparametric Dirichlet process prior, thereby extending the model of [Liao 07]. An important difference is that our work differs in the choice of classification method, where we allowed for a wider range of kernel methods for learning the decision function, rather than using a mixture of Gaussian experts. However, we also learned both the distribution of imputations and the decision function parameters simultaneously. We showed that depending on regularizer and loss function, the optimal probability distribution consists of $n+2$ imputations. An algorithm was derived that learns the decision function in an iterative manner by greedily adding new imputations in each iteration.

5.2.2 Outbound Spam Filtering with Non-Separable Loss

In [Dick 10] (Chapter 3), a filter was learned that detects unsolicited emails in the outbound email stream. Outbound spam filtering is related to inbound spam filtering, as both content and meta information is available at the time of a decision. Similar to the inbound case, meta information corresponds to knowledge about the sender. However, because customers that request delivery of their emails via the outgoing mail server have to login using their credentials, in the outbound case, sender information can be tied to a specific customer account. Also, rather than minimizing the global fractions of undetected spam and falsely detected ham emails in order to improve customer experience, email service providers face a slightly different learning problem in the case of outgoing spam emails. The loss incurred by undetected spams is a function of the rate of outgoing spam emails per time and the duration over which a high rate is maintained, which corresponds to the risk of being blacklisted. The loss of each decision depends on previous decisions, and the overall loss is therefore not separable over individual filter outputs. Minimizing the expected loss over sequences of decisions can generally be solved using Reinforcement Learning methods [Busoniu 10]. Reinforcement Learning methods learn a policy which is a mapping from states—here corresponding to some aggregated representation of sequences of emails and decisions—to distributions of actions—binary decisions of blocking or sending—with the goal of minimizing the expected loss over sequences of states and actions. We tested that approach using a Policy Gradient reinforcement learning algorithm in [Dick 10]. Another, more ad-hoc, approach is to learn a standard cost-sensitive binary classifier that, at each time point, decides if an email should be blocked or not. While such general purpose algorithms have shown good performance in other applications, we developed a filter that is specifically designed for the given application. In chapter 3 we showed that the loss can be minimized by learning a filter that throttles the rate of emails a user is allowed to send per time. This approach differs from the approach of using independent classifiers by explicitly taking into account the potential damage an illegitimate email sender can impose on the system due to its sending process. An email is blocked if sending it would exceed the rate limit. Exact delivery times are unknown in advance, so the filter has to integrate over all possible sequences of delivery times in order to minimize the expected risk of wrong decisions. Assuming a Poisson process, the misclassification probability given a rate limit and a Poisson sending rate can be modeled using Fortet's formula. We derived a convex optimization problem that can be solved using a wide range of optimizers.

5.2.3 DDoS Filtering with Structured Prediction Models

As mentioned in section 5.1, machine learning DDoS detection algorithms often aim at learning to detect attacks rather than attackers. In chapter 4, attacker detection was investigated using a novel approach that collectively classifies all client IPs that access a domain. Previous work only considered learning decision functions that decide for attackers without considering the predicted labels of other clients that access a web service [Liu 11, Devi 12]. The main reason for modeling the problem as a structured prediction problem is that the cost of not detecting attacking clients is non-linear in the number of undetected malicious clients. Such non-linear costs can be incorporated into structured prediction models but are generally hard to integrate into non-structured learning models because the loss is not separable over individual clients. This is the first time that non-linearity of the loss is taken into account when learning a DDoS filter. General structured prediction models [Lafferty 01, Tsochantaridis 05] learn a scoring function for each instance-label pair, and the decoder outputs the highest scoring label for a given input. Searching the space of possible outputs may be possible in certain cases such as sequence learning [Lafferty 01], but if the structure of the output space is less restricted, such as the exponential set of all subsets of clients, searching the complete space is impossible. To make things worse, the application in chapter 4 demands that decisions have to made in real time, thus setting even stronger restrictions on the searchable part of the space of label assignments. Learning to assign binary interdependent labels to sets of clients is a special case of *collective classification*, where relations are usually defined by graph structures. Here, the dependency graph is fully connected. Collective Classification methods aim at finding the best collective labeling by iteratively changing individual labels according to binary classification models or by randomly switching labels in order to maximize the likelihood of a Markov model [Neville 00]. In both cases, the decoder repeatedly iterates over all instances and switches individual labels according to some model that incorporates interactions between labels.

The approach we took in chapter 4 learns an informed search heuristic on the output space and is potentially applicable to any structured prediction task. Learning decoders that search the output space of structured prediction problems are also investigated in [Doppa 14a] and [Doppa 14b]. [Doppa 14b] investigate learning a decision function that serves the purpose of both deciding which label to choose when decoding time is up and deciding which outputs to investigate next. [Doppa 14a] follow the approach of separating search heuristic learning and decision function learning. In chapter 4 we also modeled search heuristic and decision function using two separate sets of parameters. We additionally estimated a distribution of available time steps and focused the learning of both search heuristic and decision function on more likely output steps. The decoder's search strategy is learned and evaluated using computationally cheap features of potentially useful outputs. The most important theoretical contribution is the derivation of an optimization problem that minimizes the expected loss over both parameter sets. The derived algorithm simultaneously optimizes over heuristic and decision function by using stochastic approximations of the real expected gradient. In [Doppa 14a], search strategy and decision function are learned in two separate steps. We proved that our combined algorithm always converges under reasonable assumptions.

5.3 Empirical Evaluation

All models and methods that were derived and investigated in this thesis were also experimentally tested on real world data sets that were collected and labeled at a large web hosting company.

In chapter 2, we evaluated the derived method and baseline methods on an email data set that was collected by the hosting company. It included both spam and ham emails, where half of the spam emails were drawn from a *hard-to-detect* set of emails that had been misclassified by other classifiers in the past. Experiments showed that the derived method can indeed lower the misclassification rate. Additional experiments on classification and regression tasks confirmed its good performance.

In chapter 3, we collected about 1,000,000 emails from approximately 10,000 accounts over the space of two days and labeled them automatically based on information passed by other email service providers via feedback loops (in most cases triggered by "report spam" buttons). This data set was used to experimentally compare the investigated throttling mechanism with baseline methods. Results showed that by using the developed method the loss can be reduced significantly.

In order to learn a DDoS attacker detection mechanism, we collected web traffic data coming to and from the hosting company's web servers. Data were collected on a per-domain basis by splitting traffic according to the HTTP header field "Host". All data corresponding to one particular domain in a time interval was considered an *event*. We defined unusual traffic patterns on a domain using a set of rules compiled by operators of the web hosting company. Rules were adjusted so that no potential attack should go unnoticed. We only stored data corresponding to 1546 unusual traffic events. Those traffic events were then labeled on a per-IP basis in a largely manual process. In a first step, complete events were checked for whether they contain an attack, and, if so, a set of heuristics was used to label chunks of IPs that access a domain as attackers if, for example, they exhibit very similar and potentially malicious behavior. All remaining IPs were labeled by manually checking their behavior. We used the assembled data set in chapter 4 to experimentally answer three questions. We evaluated if machine learning methods can in fact reduce the cost that DDoS attacks impose on a web hosting company and if classification methods perform better than anomaly detection methods. More importantly, we tested if the derived method for structured prediction can learn a good search strategy, and if structured prediction is feasible and helpful for HTTP-layer DDoS attacker detection. The results showed that it may indeed be helpful to employ attacker detection methods and that one should prefer structured classification algorithms over both independent classification methods and anomaly detection methods. The experiments also showed that the developed method works at least as good as baseline methods. To the best of our knowledge, this was the first time that the performance of machine learning methods for DDoS detection was tested on such a large real world data set.

Bibliography

[Ariu 11] Davide Ariu, Roberto Tronci & Giorgio Giacinto. *HMMPayl: An intrusion detection system based on Hidden Markov Models.* Computers & Security, vol. 30, no. 4, pages 221–241, 2011.

[Blanzieri 09] Enrico Blanzieri & Anton Bryl. *A survey of learning-based techniques of email spam filtering.* Artificial Intelligence Review, vol. 29, no. 1, pages 63–92, 2009.

[Busoniu 10] Lucian Busoniu, Robert Babuska, Bart De Schutter & Damien Ernst. Reinforcement learning and dynamic programming using function approximators. CRC press, 2010.

[Chechik 08] Gal Chechik, Geremy Heitz, Gal Elidan, Pieter Abbeel & Daphne Koller. *Max-margin classification of data with absent features.* The Journal of Machine Learning Research, vol. 9, pages 1–21, 2008.

[Devi 12] S. Renuka Devi & P. Yogesh. *Detection of Application Layer DDoS Attacks using Information Theory based Metrics.* Department of Information Science and Technology, College of Engineering Guindy 10.5121/csit.2012.2223, 2012.

[Dick 08] Uwe Dick, Peter Haider & Tobias Scheffer. *Learning from Incomplete Data with Infinite Imputations.* In Proceedings of the 25th international conference on Machine learning, pages 232–239. ACM, 2008.

[Dick 10] Uwe Dick, Peter Haider, Thomas Vanck, Michael Brückner & Tobias Scheffer. *Throttling Poisson Processes.* In Advances in Neural Information Processing Systems, pages 505–513, 2010.

[Dick 16] Uwe Dick & Tobias Scheffer. *Learning to Control a Structured-Prediction Decoder for Detection of DDoS Attackers.* Preprint of Article in Machine Learning (2016) 104: 385. doi:10.1007/s10994-016-5581-9, 2016.

[Doppa 14a] Janardhan Rao Doppa, Alan Fern & Prasad Tadepalli. *HC-Search: A learning framework for search-based structured prediction.* Journal of Artificial Intelligence Research, vol. 50, no. 1, pages 369–407, 2014.

[Doppa 14b] Janardhan Rao Doppa, Alan Fern & Prasad Tadepalli. *Structured prediction via output space search.* The Journal of Machine Learning Research, vol. 15, no. 1, pages 1317–1350, 2014.

[Düssel 08] Patrick Düssel, Christian Gehl, Pavel Laskov & Konrad Rieck. *Incorporation of application layer protocol syntax into anomaly detection.* In Information Systems Security, pages 188–202. Springer, 2008.

[Goodman 04] Joshua T Goodman & Robert Rounthwaite. *Stopping outgoing spam.* In Proceedings of the 5th ACM conference on Electronic commerce, pages 30–39. ACM, 2004.

[Görnitz 13] Nico Görnitz, Marius Kloft, Konrad Rieck & Ulf Brefeld. *Toward Supervised Anomaly Detection.* Journal of Artificial Intelligence Research, vol. 46, pages 235–262, 2013.

[Haider 09] Peter Haider & Tobias Scheffer. *Bayesian clustering for email campaign detection.* In Proceedings of the 26th Annual International Conference on Machine Learning, pages 385–392. ACM, 2009.

[Lafferty 01] John Lafferty, Andrew McCallum & Fernando CN Pereira. *Conditional random fields: Probabilistic models for segmenting and labeling sequence data.* In Proceedings of the International Conference on Machine Learning, volume 18, 2001.

[Liao 07] Xuejun Liao, Hui Li & Lawrence Carin. *Quadratically gated mixture of experts for incomplete data classification.* In Proceedings of the 24th International Conference on Machine learning, pages 553–560. ACM, 2007.

[Little 87] Roderick JA Little & Donald B Rubin. Statistical analysis with missing data. John Wiley & Sons, 1987.

[Liu 11] H. Liu & K. Chang. *Defending systems Against Tilt DDoS attacks.* In Proceedings of the International Conference on Telecommunication Systems, Services, and Applications, 2011.

[Neville 00] Jennifer Neville & David Jensen. *Iterative classification in relational data.* In Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data, 2000.

[Peng 07] Tao Peng, Christopher Leckie & Kotagiri Ramamohanarao. *Survey of network-based defense mechanisms countering the DoS and DDoS problems.* ACM Computing Surveys, vol. 39, no. 1, page 3, 2007.

[Prasse 15] Paul Prasse, Christoph Sawade, Niels Landwehr & Tobias Scheffer. *Learning to Identify Concise Regular Expressions that Describe Email Campaigns.* Journal of Machine Learning Research, vol. 16, pages 3687–3720, 2015.

[Ranjan 06] S. Ranjan, R. Swaminathan, M. Uysal & E. Knightley. *DDoS-Resilient Scheduling to Counter Application Layer Attacks under Imperfect Detection.* In Proceedings of IEEE INFOCOM, 2006.

[Robertson 10] William K. Robertson & Federico Maggi. *Effective Anomaly Detection with Scarce Training Data.* In Network and Distributed System Security Symposium, 2010.

[Shivaswamy 06] Pannagadatta K Shivaswamy, Chiranjib Bhattacharyya & Alexander J Smola. *Second order cone programming approaches for handling missing and uncertain data.* The Journal of Machine Learning Research, vol. 7, pages 1283–1314, 2006.

[Smola 05] Alexander J Smola, SVN Vishwanathan & Thomas Hofmann. *Kernel Methods for Missing Variables.* In The tenth International Workshop on Artificial Intelligence and Statistics (AISTATS), 2005.

[Tsochantaridis 05] I. Tsochantaridis, T. Joachims, T. Hofmann & Y. Altun. *Large margin methods for structured and interdependent output variables.* Journal of Machine Learning Research, vol. 6, pages 1453–1484, 2005.

[Wang 10] Chunping Wang, Xuejun Liao, Lawrence Carin & David B Dunson. *Classification with incomplete data using dirichlet process priors.* The Journal of Machine Learning Research, vol. 11, pages 3269–3311, 2010.

[Wang 13] De Wang, Danesh Irani & Calton Pu. *A study on evolution of email spam over fifteen years.* In 9th International Conference Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), pages 1–10. IEEE, 2013.

[Wressnegger 13] Christian Wressnegger, Guido Schwenk, Daniel Arp & Konrad Rieck. *A Close Look on N-grams in Intrusion Detection: Anomaly Detection vs. Classification.* In Proceedings of the ACM Workshop on Artificial Intelligence and Security, pages 67–76, 2013.

[Xie 09] Y. Xie & S. Z. Yu. *A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors.* IEEE/ACM Transactions on Networking, vol. 17, no. 1, pages 54–65, 2009.

[Zargar 13] Saman Taghavi Zargar, James Joshi & David Tipper. *A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks.* IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pages 2046–2069, 2013.

[Zhong 05] Zhenyu Zhong, Kun Huang & Kang Li. *Throttling Outgoing SPAM for Webmail Services.* In Conference on Email and Anti-Spam, 2005.