Institut für Informatik
Professur Maschinelles Lernen

Prediction with Mixture Models

Kumulative Dissertation

zur Erlangung des akademischen Grades
"doctor rerum naturalium"
(Dr. rer. nat.)
in der Wissenschaftsdisziplin Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Universität Potsdam

von
Dipl.-Inf. Peter Haider

Potsdam, den 7. August 2013

Published online at the
Institutional Repository of the University of Potsdam:
URL http://opus.kobv.de/ubp/volltexte/2014/6961/
URN urn:nbn:de:kobv:517-opus-69617
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus-69617

Abstract

Learning a model for the relationship between the attributes and the annotated labels of data examples serves two purposes. Firstly, it enables the prediction of the label for examples without annotation. Secondly, the parameters of the model can provide useful insights into the structure of the data. If the data has an inherent partitioned structure, it is natural to mirror this structure in the model. Such mixture models predict by combining the individual predictions generated by the mixture components which correspond to the partitions in the data. Often the partitioned structure is latent, and has to be inferred when learning the mixture model. Directly evaluating the accuracy of the inferred partition structure is, in many cases, impossible because the ground truth cannot be obtained for comparison. However it can be assessed indirectly by measuring the prediction accuracy of the mixture model that arises from it. This thesis addresses the interplay between the improvement of predictive accuracy by uncovering latent cluster structure in data, and further addresses the validation of the estimated structure by measuring the accuracy of the resulting predictive model.

In the application of filtering unsolicited emails, the emails in the training set are latently clustered into advertisement campaigns. Uncovering this latent structure allows filtering of future emails with very low false positive rates. In order to model the cluster structure, a Bayesian clustering model for dependent binary features is developed in this thesis.

Knowing the clustering of emails into campaigns can also aid in uncovering which emails have been sent on behalf of the same network of captured hosts, so-called botnets. This association of emails to networks is another layer of latent clustering. Uncovering this latent structure allows service providers to further increase the accuracy of email filtering and to effectively defend against distributed denial-of-service attacks. To this end, a discriminative clustering model is derived in this thesis that is based on the graph of observed emails. The partitionings inferred using this model are evaluated through their capacity to predict the campaigns of new emails.

Furthermore, when classifying the content of emails, statistical information about the sending server can be valuable. Learning a model that is able to make use of it requires training data that includes server statistics. In order to also use training data where the server statistics are missing, a model that is a mixture over potentially all substitutions thereof is developed.

Another application is to predict the navigation behavior of the users of a website. Here, there is no a priori partitioning of the users into clusters, but to understand different usage scenarios and design different layouts for them, imposing a partitioning is necessary. The presented approach simultaneously optimizes the discriminative as well as the predictive power of the clusters.

Each model is evaluated on real-world data and compared to baseline methods. The results show that explicitly modeling the assumptions about the latent cluster structure leads to improved predictions compared to the baselines. It is beneficial to incorporate a small number of hyperparameters that can be tuned to yield the best predictions in cases where the prediction accuracy can not be optimized directly.

Zusammenfassung

Das Lernen eines Modells für den Zusammenhang zwischen den Eingabeattributen und annotierten Zielattributen von Dateninstanzen dient zwei Zwecken. Einerseits ermöglicht es die Vorhersage des Zielattributs für Instanzen ohne Annotation. Andererseits können die Parameter des Modells nützliche Einsichten in die Struktur der Daten liefern. Wenn die Daten eine inhärente Partitionsstruktur besitzen, ist es natürlich, diese Struktur im Modell widerzuspiegeln. Solche Mischmodelle generieren Vorhersagen, indem sie die individuellen Vorhersagen der Mischkomponenten, welche mit den Partitionen der Daten korrespondieren, kombinieren. Oft ist die Partitionsstruktur latent und muss beim Lernen des Mischmodells mitinferiert werden. Eine direkte Evaluierung der Genauigkeit der inferierten Partitionsstruktur ist in vielen Fällen unmöglich, weil keine wahren Referenzdaten zum Vergleich herangezogen werden können. Jedoch kann man sie indirekt einschätzen, indem man die Vorhersagegenauigkeit des darauf basierenden Mischmodells misst. Diese Arbeit beschäftigt sich mit dem Zusammenspiel zwischen der Verbesserung der Vorhersagegenauigkeit durch das Aufdecken latenter Partitionierungen in Daten, und der Bewertung der geschätzen Struktur durch das Messen der Genauigkeit des resultierenden Vorhersagemodells.

Bei der Anwendung des Filterns unerwünschter E-Mails sind die E-Mails in der Trainingsmende latent in Werbekampagnen partitioniert. Das Aufdecken dieser latenten Struktur erlaubt das Filtern zukünftiger E-Mails mit sehr niedrigen Falsch-Positiv-Raten. In dieser Arbeit wird ein Bayes'sches Partitionierunsmodell entwickelt, um diese Partitionierungsstruktur zu modellieren.

Das Wissen über die Partitionierung von E-Mails in Kampagnen hilft auch dabei herauszufinden, welche E-Mails auf Veranlassen des selben Netzes von infiltrierten Rechnern, sogenannten Botnetzen, verschickt wurden. Dies ist eine weitere Schicht latenter Partitionierung. Diese latente Struktur aufzudecken erlaubt es, die Genauigkeit von E-Mail-Filtern zu erhöhen und sich effektiv gegen verteilte Denial-of-Service-Angriffe zu verteidigen. Zu diesem Zweck wird in dieser Arbeit ein diskriminatives Partitionierungsmodell hergeleitet, welches auf dem Graphen der beobachteten E-Mails basiert. Die mit diesem Modell inferierten Partitionierungen werden via ihrer Leistungsfähigkeit bei der Vorhersage der Kampagnen neuer E-Mails evaluiert.

Weiterhin kann bei der Klassifikation des Inhalts einer E-Mail statistische Information über den sendenden Server wertvoll sein. Ein Modell zu lernen das diese Informationen nutzen kann erfordert Trainingsdaten, die Serverstatistiken enthalten. Um zusätzlich Trainingsdaten benutzen zu können, bei denen die Serverstatistiken fehlen, wird ein Modell entwickelt, das eine Mischung über potentiell alle Einsetzungen davon ist.

Eine weitere Anwendung ist die Vorhersage des Navigationsverhaltens von Benutzern einer Webseite. Hier gibt es nicht a priori eine Partitionierung der Benutzer. Jedoch ist es notwendig, eine Partitionierung zu erzeugen, um verschiedene Nutzungsszenarien zu verstehen und verschiedene Layouts dafür zu entwerfen. Der vorgestellte Ansatz optimiert gleichzeitig die Fähigkeiten des Modells, sowohl die beste Partition zu bestimmen als auch mittels dieser Partition Vorhersagen über das Verhalten zu generieren.

Jedes Modell wird auf realen Daten evaluiert und mit Referenzmethoden verglichen. Die Ergebnisse zeigen, dass das explizite Modellieren der Annahmen über die latente Partitionierungsstruktur zu verbesserten Vorhersagen führt. In den Fällen bei denen die Vorhersagegenauigkeit nicht direkt optimiert werden kann, erweist sich die Hinzunahme einer kleinen Anzahl von übergeordneten, direkt einstellbaren Parametern als nützlich.

Contents

Chapter 1

Introduction

In many applications, one wants to annotate instances of data with labels that indicate how to further process them. Examples include email messages, where the service provider wants to know whether to move them to a special folder because they are unsolicited (spam), or transmit them normally. In cases where obtaining the labels for all instances by other means is too costly or impractical, one can learn a model for the relationship between the examples' attributes and their labels. For that, the learner usually needs a much smaller set of examples where both the attributes and labels are known. In the following, let x denote the attributes of an example, y a label, θ the parameters of a model, and c a cluster of instances. In general, the learner is given a training set $X = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where the x_i specify the attributes of the i-th example, and the y_i the respective labels. The goal is to infer a prediction model $P(y|x;\theta)$, parameterized by some set of parameters θ. The latter are usually optimized according to the accuracy of the prediction model on the training data. After that, the model can be used to predict the labels y of new examples. Sometimes the predictions themselves are not relevant, but only the model parameters that serve as a way to interprete or explain the training data. But even in these cases, the predictions on new data are useful because their accuracy provides an indicator for the appropriateness of the model parameters.

1.1 Mixture Models

Mixture models are a class of prediction models that rely on an underlying partition structure in the relationship between attributes and label. They have an advantage of exhibiting greater complexity than their underlying single-component models, while in many cases, retaining their same efficient inference methods. For example, learning a linear classifier using a convex loss function can be done very efficiently, but the discriminative power of linear models is too low in some applications. A common approach to increasing the discriminative power is to map the examples into a higher-dimensional feature space and learn a decision funtion in the feature space. If the objective function is quadratically regularized, the solution can be expressed solely in terms of inner products of examples in the feature space, and thus the mapping does not have to be done explicitly, but rather the inner product may be defined directly as a kernel function. The resulting decision function can then be non-linear in the input examples. However kernel methods are usually much slower than linear methods, because evaluating the decision function requires iterating over potentially

all training examples. But there already exists a partitioning of the training set into components, learning a linear classifier for each component is easy, and the resulting mixture model is then effectively non-linear. Especially if the data actually exhibit an albeit unknown cluster structure, taking this structure into account when learning a predictive model can be advantageous.

Examples of successful uses of mixture models include speech recognition [Povey 10], clustering of gene expression microarray data [McNicholas 10], modeling the returns of financial assets [Geweke 11], and image segmentation [Zhang 11].

In general, the partitioning of the training set is not given, hence the task has two parts. Firstly, the learner has to find a partioning C consisting of clusters $c_1, \ldots, c_m$, such that $\bigcup c_i = X$ and $\forall i \neq j : c_i \cap c_j = \emptyset$. Secondly, for each cluster, one has to estimate its predictive model parameters θ. They can be either estimated in a maximum-a-posteriori fashion,

$$\theta^* = \arg\max_\theta P(\theta)P(y|x;\theta),$$
$$P(y|x) = P(y|x;\theta^*),$$

or they can be integrated out in order to perform full Bayesian inference:

$$P(y|x) = \int P(\theta)P(y|x;\theta)d\theta.$$

While the latter is more exact, the former is used in cases when full Bayesian inference is computationally intractable.

Within a component c, the mixture model defines a distribution $P(y|x;c)$ because each component is associated with a set of parameters θ. The component-wise distributions can be combined by summing over the components, weighted by their probabilities given input x,

$$P(y|x) = \sum_c P(c|x)P(y|x;c). \tag{1.1}$$

Assessing the predictive accuracy of the resulting distribution $P(y|x)$ indirectly allows one to evaluate the performance of the partitioning C. Although there is no deterministic relationship between the two, in general a higher accuracy of $P(y|x)$ requires a higher accuracy of C. Of course, in simple settings this relationship breaks down; for example, if the output can be predicted from the input completely independent from the cluster structure. But we focus on applications where there is in fact a latent cluster structure in the data, and where knowing to which cluster an example belongs facilitates the prediction of its output.

This thesis is concerned with the interplay between two mutually dependent processes: the one hand the improvement of predictive accuracy by uncovering the latent mixture structure in the data, and the validation of the estimated mixture structure by measuring the accuracy of the resulting predictive model. It is not only of interest to evaluate a particular estimated clustering, but also to evaluate different methods of estimating clusterings. In particular, several new methods for estimating mixture models from training data are presented. The aim of the methods is to forgo unjustified assumptions about the data and to enable direct optimization of predictive accuracy when possible. This results in a clear relationship between the resulting predictive accuracy and the validity of the produced clusterings, setting the methods apart from previously published clustering approaches where there is, in general, no possibility to assess the partitionings inferred from real-world data.

1.2 Classifying Emails with Mixtures of Campaigns

In the application of spam filtering, the inputs x are the contents of emails and sometimes additional meta-data. The label that is to be predicted is binary, $y \in \{-1, 1\}$, representing non-spam or spam. The most common way to represent the content of an email is to use a bag-of-words representation of the text and use a binary indicator vector that has a component for each word in the vocabulary. Thus the vector x consists of ones that correspond to words that are included in the email and zeroes elsewhere. Due to the high dimensionality of the input vectors and the large number of training examples, usually generalized linear models are used to model $P(y|x)$. Let w be the vector of same dimensionality as x parameterizing the distribution, and g an arbitrary link function such that $g^{-1} : \mathbb{R} \to [-1; 1]$. Then the predictive distribution is defined as

$$P(y|x; w) = g^{-1}(w^\top x).$$

To guarantee very low false positive rates, an email provider wants to train a conservative spam filter, which does not try to capture the essence of "spamness", but instead filters only emails that belong to a known spam campaign. This is not possible using a single generalized linear model since those result in a planar decision boundary, and thus all emails that lie between the known spam campaigns in input space are filtered as well. A remedy is to combine several linear classifiers, one for each campaign of spam emails. Since the probability that a message is spam given that it belongs to a spam capaign is one, Equation 1.1 can be written as

$$P(y = 1|x) = \frac{\sum_{c \in C} P(x|c)P(c)}{P(x)}. \tag{1.2}$$

The main challenge is to find a clustering C of a training set X into campaigns. We do this by constructing a Bayesian mixture model over emails in a transformed feature space in which the dimensions are treated as independent:

$$P(X|C) = \prod_{c \in C} \int_\theta P(\theta) \prod_{x \in c} P(\phi(x)|\theta) d\theta. \tag{1.3}$$

Here, θ is the parameter vector that governs the distribution over transformed feature vectors within a cluster, and ϕ is the transformation mapping. It is an injective Boolean function, constructed with the goal to minimize the negative influence of the independence assumption for the transformed feature dimensions. Previous approaches for modeling the generative process of text data fall into two categories. The first category of approaches is to include an independence assumption in the input space which is clearly unjustified and harms performance. The second possibility is to use an n-gram model which inflates the dimensionality of the data and therefore suffers from the problem that the overwhelming majority of n-grams have never been seen before in the training set.

Using a multivariate Beta distribution for the prior $P(\theta)$ yields a closed-form solution for the integral in Equation 1.3, and we can find a good approximative solution to the clustering problem $\arg\max_C P(X|C)$ efficiently using a greedy sequential algorithm.

It is problematic to estimate $P(c)$ in Equation 1.2 from training data, because the activities of spam campaigns can change rapidly within short timeframes. Thus we assume a uniform distribution over campaigns. This could lead to the overestimation of $P(y = 1|x)$ for messages from campaigns that are split over multiple clusters by the clustering process. To correct for this, we replace the sum over campaigns by the maximum over campaigns. Finally, the constant $1/|C|$ from the uniform distribution over clusters can be omitted, since we are only concerned with the relative probabilities of messages being spam compared to each other. This leads us to the final classification score as

$$\text{score}(x) = \frac{\max_{c \in C} P(x|c)}{P(x)}.$$

In this application, not only the predictions of the final model are of interest, but also the clustering itself has intrinsic value. It provides insight into the behavior of spammers: it can tell us about how often the campaign templates change, how many different campaigns are active at any given time, and about the employed strategies for dissemination.

This approach is detailed and evaluated by Haider and Scheffer [Haider 09]. The paper makes three major contributions. The most import contribution is the generative model for a clustering of binary feature vectors, based on a transformation of the inputs into a space in which an independence assumption induces minimum approximation error. Furthermore, the paper presents an optimization problem for learning the transformation together with an appropriate algorithm. And finally it details a case study on real-world data that evaluates the clustering solution on email campaign detection.

For the paper I, derived the generative model and the feature transformation, proved the theorem, developed and implemented the algorithms and conducted the experiments.

1.3 Predicting Email Campaigns Using a Botnet Model

In this application, the prediction of email campaigns is mainly a method to evaluate the accuracy of the clustering. We are given a set of emails, consisting of their sender's IP address x and their spam campaign y. The goal is to find a clustering C of the pairs (x, y), such that the clusters correspond to the botnets on whose behalf the emails were sent. Knowing which IP address belongs to which botnet at any time facilitiates the defense against distributed denial-of-service (DDOS) attacks. However, there is no ground truth available with which to assess an inferred clustering. But by evaluating the prediction of campaigns given IP addresses and a current clustering, we can indirectly compare different methods of clustering emails into botnets. The more accurate the clustering C, the more accurate the predictive distribution that is based on the clustering tends to be:

$$P(y|x; C) = \sum_{c \in C} P(y|x; c)P(c|x).$$

We make the assumption that the campaign of an email is conditionally independent of its IP address given its cluster; i.e., $P(y|x;c) = P(y|c)$. Both the distribution over campaigns within a botnet $P(y|c)$ and the distribution over botnets given an IP address can be straightforwardly estimated using a maximum-likelihood approach. The main challenge is again to find the clustering C of a given set of training pairs. We construct a discriminative model $P(C|(x_1, y_1), \ldots, (x_n, y_n))$ that represents its input as a graph of emails, where there is an edge between two emails if and only if they are from the same campaign or IP address. The crucial observation is that edges present in the graph provide weak evidence for two emails belonging to the same botnet, whereas the absence of an edge only provides very weak evidence against it. Our model thus is defined purely in terms of the cliques of the graph. Within each clique, we define the posterior distribution of the partitioning of the emails of the clique to follow the Chinese Restarant Process (CRP). The individual clique distributions are then multiplied and normalized. Since there is no closed-form solution for the full Bayesian approach of averaging over all clusterings for predicting campaigns, we devise a blockwise Gibbs sampling scheme that, in the limit of an infinite number of iterations, generates samples $C_1, \ldots, C_T$ from the true posterior. Predictions are then averaged over the generated samples, such that the final predictive distribution is

$$P(y|x; C_1, \ldots, C_T) = \sum_t \sum_{c \in C_t} P(c|x, C_t) P(y|c).$$

This approach is detailed and evaluated by Haider and Scheffer [Haider 12b]. The paper contributes a discriminative model for clustering in a graph without making distributional assumptions about the generative process of the observables. Additionally, it presents a Gibbs sampling algorithm with which to efficiently traverse the state graph to generate unbiased samples from the posterior in the limit. Finally, it reports on a case study with data from an email service provider.

For the paper, I formulated the problem setting, derived the probabilistic model, proved the theorems, developed and implemented the algorithm, and conducted the experiments.

1.4 Classifying Emails with Missing Values

In spam filtering, one can use additional meta-information z about an email in addition to its content x, such as statistics about the sending server. Yet because we want to utilize training data from external sources that do not include these statistics, we are faced with the problem of learning a classifier from examples with missing values. At test time, the server statistics are always available.

The way to do this without introducing simplifications is to find a set of imputations[1] $\Omega = \{\omega_1, \ldots, \omega_D\}$ for the missing values. Each ω_d is a complete instantiation of the meta-information of all examples, and restricted to z in places where the true meta-information is known. We prove that using at most n different sets of imputations, where n is the size of the training set, is equivalent to using a mixture of

[1] In the area of learning from data with missing values, the assumed values are usually called imputations.

infinitely many imputation sets. We assume a known prior distribution $P(\omega)$ over imputations. It can, for example, represent the belief that imputed values are normally distributed about the mean of the actual values of other examples in the same feature dimension. Unlinke previous work, our approach finds a complete distribution over all possible imputations, instead of finding only a single imputation for each example.

The ω_d correspond to the components of a mixture model. Their respective predictive distributions $P(y|x, z; \omega)$ can be modeled and learned as a conventional binary classification problem. We choose a decision function kernelized with a Mercer kernel k,

$$f(x, z) = \sum_i \alpha_i y_i k(x, z, x_i, \omega),$$

where all mixture components share the same dual multipliers α_i.

Thus the task is to simultaneously estimate the multipliers α_i and the weights $\beta_1, \ldots, \beta_D$ of the individual mixture components. The most concise way to do this is to construct a joint optimization problem, where both are free parameters and the objective function is defined in terms of the classification error on the training examples, the prior knowledge $P(\omega)$ about the imputations, and a regularizer on the norm of the weight vector in feature space. This amounts to optimizing

$$\begin{aligned}\arg\min_{\alpha,\beta,\omega} \sum_i l^h(y_i, \sum_j \alpha_j y_j \sum_d \beta_d k(x_i, \omega_{d,i}, x_j, \omega_{d,j}) \\ + \lambda \sum_{i,j,d} \alpha_i \alpha_j \beta_d k(x_i, \omega_{d,i}, x_j, \omega_{d,j}) \\ + \lambda' \sum_d \beta_d \log P(\omega_d),\end{aligned}$$

where $l^h(y, f) = \max(1 - yf, 0)$ is the hinge loss, under the constraints that $\forall d : \beta_d \geq 0$ and $\sum_d \beta_d = 1$. It can be solved in its dual form using a min-max-procedure that iteratively selects the next best set of imputations ω_d and its weight β_d and then re-adjusts the multipliers α_i.

This approach is detailed and evaluated by Dick et al. [Dick 08]. The novel contributions of this paper are as follows. Firstly, it presents an optimization problem for learning a predictor from incomplete data that makes only minimal assumptions about the distribution of the missing attribute values, and allows potentially infintely many imputations. Secondly, it contains a proof that the optimal solution can be expressed in terms of a finite number of imputations. On multiple datasets it shows that the presented method consistently outperforms single imputations.

For this paper I developed the theoretical framework and the optimization problems, and proved the theorem. Uwe Dick implemented the algorithm and conducted the experiments.

1.5 Predicting User Behavior in Different Market Segments

In the final application, the predictions of the learned mixture model are only of secondary importance to the prediction model itself. Training examples consist of

attributes x and behavior y. The goal is to simultaneously find a set of k market segments with associated parameters $\theta_1, \ldots, \theta_k$, and a classifier $h : \mathcal{X} \to \{1, \ldots, k\}$. The classifier sorts each example into one of the segments, given its attributes. The segment parameters are supposed to predict the behavior of the examples in the segment, and thus serve as a description of the segment. For example, in the domain of user behavior on a website, each example corresponds to one browsing session. Its attributes are the timestamp, the referrer domain, and the category of the first pageview. Its behavior is represented by the categories of the subsequent pageviews and the layout elements in which the clicked links reside. Given a set of segments of the training examples, one can craft specific website layouts for each segment, which cater best to the included browsing sessions' preferences. New sessions can then be classified using h into one of the segments and presented with the corresponding layout.

The operator of the website wants the classifier to accurately discriminate examples given only their attributes, and also wants the segment parameters to predict the behavior of all examples that are classified into it with high accuracy. Therefore, first finding a clustering in the space of attributes and then selecting parameters for each cluster is not advisable because it can result in poor prediction of the behavior. Analogously, first computing a clustering in the space of behaviors leads to good predictions of the behavior within each cluster, but can gravely harm the capability of any classifier to separate examples into the respective clusters where their behavior is best predicted. The appropriate optimization problem is thus

$$\arg\max_{h,\theta} \sum_i \log P(y_i|\theta_{h(x_i)}). \tag{1.4}$$

The premise here is that the predictive distribution family $P(y|\theta)$ is chosen so that the parameters θ can be interpreted in a meaningful way to guide the design of segmented marketing strategies. Additionally, estimating the parameters within a segment c given the contained examples with indices $i \in c$ has to be efficient. For example, if the behavior is an exchangeable sequence of mutually independent multinomial variables the optimization problem

$$\arg\max_{\theta} \sum_{i \in c} \log P(y_i|\theta)$$

has a closed-form solution, and the parameters θ of the segment can be interpreted as the preferences of the members of the segment. Our approach stands in contrast to previously published market segmentation methods. Instead of defining a priori criteria for clustering the examples, we simultaneously optimize the clustering according to the goals of separability and homogeneity.

Solving the optimization problem of Equation 1.4 in the straightforward manner of alternating between optimizing h and θ does not work in practice because it gets stuck in poor local optima where, in most cases, the majority of segments are empty. As a remedy, we develop an algorithm that relies on the expression of the parameters of h as a function of the parameters θ. The resulting objective function can be maximized approximately using a variant of the EM-algorithm.

This approach is detailed and evaluated by Haider et al. [Haider 12a]. The paper presents the first concise optimization problem for market segmentation, which

consists of learning a classifier and component-wise predictors. An algorithm for approximately solving the problem is detailed, and evaluated on a large-scale dataset of user navigation behavior of a news website.

For the paper I formulated the problem setting and the optimization problem, derived the approximate solution, developed and implemented the algorithms, and conducted the main experiments. Luca Chiarandini preprocessed the data and implemented and conducted the baseline experiments.

Bayesian Clustering for Email Campaign Detection

Peter Haider HAIDER@CS.UNI-POTSDAM.DE
Tobias Scheffer SCHEFFER@CS.UNI-POTSDAM.DE
University of Potsdam, Department of Computer Science, August-Bebel-Strasse 89, 14482 Potsdam, Germany

Abstract

We discuss the problem of clustering elements according to the sources that have generated them. For elements that are characterized by independent binary attributes, a closed-form Bayesian solution exists. We derive a solution for the case of dependent attributes that is based on a transformation of the instances into a space of independent feature functions. We derive an optimization problem that produces a mapping into a space of independent binary feature vectors; the features can reflect arbitrary dependencies in the input space. This problem setting is motivated by the application of spam filtering for email service providers. Spam traps deliver a real-time stream of messages known to be spam. If elements of the same campaign can be recognized reliably, entire spam and phishing campaigns can be contained. We present a case study that evaluates Bayesian clustering for this application.

1. Introduction

In model-based clustering, elements $X = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$ have been created by an unknown number of sources; each of them creates elements according to its specific distribution. We study the problem of finding the most likely clustering of elements according to their source

$$C^* = \arg\max_C P(X|C), \tag{1}$$

where C consistently partitions the elements of X into clusters $C = \{c_1, \ldots, c_m\}$ that are mutually exclusive and cover each element.

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

Computing the likelihood of a set X under a clustering hypothesis C requires the computation of the joint likelihood of mutually dependent elements X_c within a partition c. This is usually done by assuming latent mixture parameters θ that generate the elements of each cluster and imposing a prior over the mixture parameters. The joint likelihood is then an integral over the parameter space, where individual likelihoods are independent given the parameters:

$$P(X_c) = \int \prod_{\mathbf{x} \in X_c} P(\mathbf{x}|\theta) P(\theta) d\theta. \tag{2}$$

For suitable choices of $P(\mathbf{x}|\theta)$ and $P(\theta)$, this integral has an analytic solution. In many cases, however, the space $\mathcal{X}$ of elements is very high-dimensional, and the choice of likelihood and prior involves a trade-off between expressiveness of the generative model and tractability regarding the number of parameters. If the elements are binary vectors, $\mathcal{X} = \{0,1\}^D$, one extreme would be to model $P(\mathbf{x}|\theta)$ as a full multinomial distribution over $\mathcal{X}$, involving 2^D parameters. The other extreme is to make an independence assumption on the dimensions of $\mathcal{X}$, reducing the model parameters to a vector of D Bernoulli probabilities. No remedy for this dichotomy is known that preserves the existence of an analytic solution to the integral in Equation 2.

Our problem setting is motivated by the application of clustering messages according to campaigns; this will remain our application focus throughout the paper. Filtering spam and phishing messages reliably remains a hard problem. Email service providers operate *Mail Transfer Agents* which observe a stream of incoming messages, most of which have been created in bulk by a generator. A generator can be an application that dispatches legitimate, possibly customized newsletters, or a script that creates spam or phishing messages and disseminates them from the nodes of a botnet. Mail Transfer Agents typically blacklist known spam and phishing messages. Messages known to be spam can be collected by tapping into botnets, and by harvesting emails in spam traps. Spam traps are email addresses published invisibly on the web that

have no legitimate owner and can therefore not receive legitimate mail. In order to avoid blacklisting, spam dissemination tools produce emails according to probabilistic templates. This motivates our problem setting: If all elements that are generated in a joint campaign can be identified reliably, then all instances of that campaign can be blacklisted as soon as one element reaches a spam trap, or is delivered from a known node of a botnet. Likewise, all instances of a newsletter can be whitelisted as soon as one instance is confirmed to be legitimate.

While text classification methods are frequently reported to achieve extremely high accuracy for spam filtering under laboratory conditions, their practical contribution to the infrastructure of email services is smaller: they are often applied to decide whether accepted emails are to be delivered to the inbox or the spam folder. The vast majority of all spam delivery attempts, however, is turned down by the provider based on known message and IP blacklists. Text classifiers are challenged with continuously shifting distributions of spam and legitimate messages; their risk of false positives does not approach zero sufficiently closely to constitute a satisfactory solution to the spam problem.

This paper makes three major contributions. Firstly, we develop a generative model for a clustering of binary feature vectors, based on a transformation of the input vectors into a space in which an independence assumption incurs minimal approximation error. The transformations can capture arbitrary dependencies in the input space while the number of parameters stays reasonable and full Bayesian inference remains tractable. Secondly, we derive the optimization problem and algorithm that generates the feature transformation. Finally, we present a large-scale case study that explores properties of the Bayesian clustering solution for email campaign detection.

The paper is structured as follows. We present the Bayesian clustering model in Section 2, and an optimization problem and algorithm for transforming dependent features into independent features in Section 3. Section 4 discusses the estimation of prior parameters, Section 5 develops a sequential clustering algorithm based on Bayesian decisions. Section 6 reports on empirical results in our motivating application. We review related work in Section 7. Section 8 concludes.

2. Bayesian Clustering for Binary Features

In general, a clustering hypothesis C entails that the likelihood of the dataset factorizes into the likelihoods of the subsets X_c of elements in the clusters $c \in C$. Elements within a cluster are dependent, so the likelihood of each element depends on the preceding elements in its cluster, as in Equation 3.

$$\begin{aligned} P(X|C) &= \prod_{c \in C} P(X_c) \\ &= \prod_{c \in C} \prod_{i:\mathbf{x}^{(i)} \in c} P(\mathbf{x}^{(i)} | \{\mathbf{x}^{(j)} \in c : j < i\}) \end{aligned} \quad (3)$$

The crucial part is modeling the probability $P(\mathbf{x}|X')$ of a binary feature vector $\mathbf{x}$ given a set of elements X'. A natural way is to introduce latent model parameters θ and integrate over them as in Equation 4.

$$P(\mathbf{x}|X') = \int_\theta P(\mathbf{x}|\theta) P(\theta|X') d\theta \quad (4)$$

Modeling θ as a full joint distribution over all 2^D possible feature vectors, with D being the dimensionality of the input space $\mathcal{X}$, is intractable.

Let ϕ_e be independent binary features and let vector $\phi(\mathbf{x})$ be a representation of $\mathbf{x}$ in the space of independent features ϕ. In order to streamline the presentation of the clustering model, we postpone the rationale and construction of the feature transformation to Section 3. Under the assumption that attributes in the space ϕ are independent, the model parameters can be represented as a vector of Bernoulli probabilities, $\theta \in (0,1)^E$, and we can compute $P(\mathbf{x}|\theta)$ as $\prod_{e=1}^{E} P(\phi_e(\mathbf{x})|\theta_e)$. Furthermore, we impose a Beta prior on every component θ_e with parameters α_e and β_e. Since the Beta distribution is conjugate to the Bernoulli distribution, we can now compute the posterior over the model parameters analytically as in Equation 5, where $\#_e = |\{\mathbf{x}' \in X' : \phi_e(\mathbf{x}') = 1\}|$.

$$\begin{aligned} P(\theta|X') &= P(X'|\theta) \frac{P(\theta)}{P(X')} \\ &= \frac{\prod_{e=1}^{E} \prod_{\mathbf{x} \in X'} P(\phi_e(\mathbf{x})|\theta_e) P_{Beta}(\theta_e|\alpha_e, \beta_e)}{\int P(X'|\theta') P(\theta') d\theta'} \\ &= \prod_{e=1}^{E} P_{Beta}(\theta_e|\alpha_e + \#_e, \beta_e + |X| - \#_e) \end{aligned} \quad (5)$$

The integral in Equation 4 then has the analytic solution of Equation 6:

$$\begin{aligned} P(\mathbf{x}|X') = &\prod_{e:\phi_e(\mathbf{x})=1} \frac{\alpha_e + \#_e}{\alpha_e + \beta_e + |X'|} \\ &\prod_{e:\phi_e(\mathbf{x})=0} \frac{\beta_e + |X'| - \#_e}{\alpha_e + \beta_e + |X'|}. \end{aligned} \quad (6)$$

For a single element, independent of all others, the probability term simplifies to

$$P(\mathbf{x}) = \prod_{e:\phi_e(\mathbf{x})=1} \frac{\alpha_e}{\alpha_e+\beta_e} \prod_{e:\phi_e(\mathbf{x})=0} \frac{\beta_e}{\alpha_e+\beta_e}. \quad (7)$$

Furthermore, the joint probability of an interdependent set X' in one cluster can be computed as

$$\begin{aligned} P(X') &= \prod_{i:\mathbf{x}^{(i)}\in X'} P(\mathbf{x}^{(i)}|\{\mathbf{x}^{(j)} \in X' : j<i\}) \\ &= \prod_{e=1}^{E} \frac{\prod_{k=1}^{\#_e}(\alpha_e+k-1)\prod_{k=1}^{|X'|-\#_e}(\beta_e+k-1)}{\prod_{k=1}^{|X'|}(\alpha_e+\beta_e+k-1)} \\ &= \prod_{e=1}^{E} \frac{B(\alpha_e+\#_e, \beta_e+|X'|-\#_e)}{B(\alpha_e,\beta_e)}, \end{aligned}$$

where B denotes the Beta function.

3. Feature Transformation

In this section we will present a method of approximating a distribution over high-dimensional binary vectors that allows analytical integration over the induced model parameters. The idea is to find a mapping into another space of binary vectors where the dimensions are treated independently of each other, such that the divergence between the original distribution and the approximate distribution defined in terms of the mapped vectors is minimal.

A straightforward approach would be to employ a model that captures dependencies between small sets of attributes only and assumes independence otherwise. Instead of working with independence assumptions, we construct a search space of transformations. This space is complete in the sense that any possible interaction of attributes can be reflected in the newly constructed attributes. These attributes are constructed such that their product approximates the true distribution as closely as possible.

The Bayesian clustering model introduced in Section 2 requires us to infer the probability of a feature vector $\mathbf{x}$ given a set of feature vectors X' it depends on. We therefore want to approximate $P(\mathbf{x}|X')$ by a quantity $Q_\phi(\mathbf{x}|X')$, where ϕ is a mapping from the original vector space $\mathcal{X}=\{0,1\}^D$ to the image space $\mathcal{Z}=\{0,1\}^E$. We define Q_ϕ as a product over independent probabilities for each output dimension, as in Equation 8.

$$Q_\phi(\mathbf{x}|X') = \prod_{e=1}^{E} P(\phi_e(\mathbf{x})|\phi_e(X')) \quad (8)$$

By its definition as a product over probabilities, quantity $Q_\phi(\mathbf{x}|X)$ is always non-negative; however, it does not necessarily sum to one over the event space of possible inputs $\mathbf{x}$. Since Q_ϕ serves as an approximation of a probability in the Bayesian inference, it is desirable that $\sum_{\mathbf{x}} Q_\phi(\mathbf{x}|X) \leq 1$ for all X. Moreover, the natural measure of approximation quality – the Kullback-Leibler divergence – is only motivated for measures that add up to at most one and may be maximized by trivial solutions otherwise. Note that the sum does not have to be exactly 1, since an extra element $\bar{\mathbf{x}}$ with $P(\bar{\mathbf{x}}) = 0$ can be added to $\mathcal{X}$ that absorbs the remaining probability mass, $Q_\phi(\bar{\mathbf{x}}|X') \stackrel{\text{def}}{=} 1 - \sum_{\mathbf{x}\in\mathcal{X}} Q_\phi(\bar{\mathbf{x}}|X')$.

Normalization of $Q_\phi(\mathbf{x}|X)$ is intractable, since it would require explicit summation of Equation 8 over all 2^D possible input elements. We therefore have to define the space of possible transformations such that after any transformation Q_ϕ is guaranteed to sum to at most one. By Theorem 1 (see Appendix), this holds for all injective transformations.

Every mapping from $\mathcal{X}=\{0,1\}^D$ to the $\mathcal{Z}=\{0,1\}^E$ can be represented as a set of E Boolean functions, and every Boolean function can be constructed as a combination of elementary operations. Therefore we can define the search space as the set of all concatenations of elementary Boolean transformations ψ that preserve injectiveness. The choice of which elementary transformations to use is driven by the practical goal that ϕ also preserves sparseness. The following two elementary transformations are injective, sufficient to generate any Boolean function, and preserve sparsity:

$$\begin{aligned} &\psi^x_{ij}((\dots,x_i,\dots,x_j,\dots)^\top) = (\dots,x_i,\dots,x_i\neq x_j,\dots)^\top \\ &\psi^a_{ij}((\dots,x_i,\dots,x_j,\dots)^\top) \\ &\qquad = (\dots,x_i\wedge x_j,\dots,x_i\wedge\neg x_j,\neg x_i\wedge x_j,\dots)^\top. \end{aligned}$$

Every ψ replaces two features by Boolean combinations thereof, leaving every other feature untouched.

For any set of elements X, the quantity $Q_\phi(\mathbf{x}|X)$ should minimize the Kullback-Leibler divergence from the true distribution $P(\mathbf{x}|X)$. Hence, the optimization criterion (Equation 9) is the expected KL divergence between $Q_\phi(\mathbf{x}|X)$ and $P(\mathbf{x}|X)$ over all X.

$$\begin{aligned} \phi^* &= \arg\min_\phi \mathop{E}_{\mathrm{X}\sim\mathrm{P}(\mathbf{x})} [KL(P(\cdot|X)||Q_\phi(\cdot|X))] && (9) \\ &= \arg\min_\phi \mathop{E}_{\mathrm{X}\sim\mathrm{P}(\mathbf{x})} \left[\sum_{\mathbf{x}\in\mathcal{X}} P(\mathbf{x}|X)\log\frac{P(\mathbf{x}|X)}{Q_\phi(\mathbf{x}|X)}\right] && (10) \\ &= \arg\max_\phi \mathop{E}_{\mathrm{X}\sim\mathrm{P}(\mathbf{x})} \left[\sum_{\mathbf{x}\in\mathcal{X}} P(\mathbf{x}|X)\log Q_\phi(\mathbf{x}|X)\right] && (11) \\ &= \arg\max_\phi \mathop{E}_{\mathrm{X}\sim\mathrm{P}(\mathbf{x}),\mathbf{x}\sim\mathrm{P}(\mathbf{x}|\mathrm{X})} [\log Q_\phi(\mathbf{x}|X)]. && (12) \end{aligned}$$

Equation 10 expands the definition of the KL divergence; Equation 11 replaces minimization by maximization and drops the term $P(\mathbf{x}|X)\log P(\mathbf{x}|X)$ which is constant in ϕ. We approximate the expectation in Equation 12 by the sum over an empirical sample S, and obtain

Optimization Problem 1. *Over the set of concatenations of elementary transformations, $\phi \in \{\psi^x_{ij}, \psi^a_{ij}\}^*$, maximize*

$$\sum_{\mathbf{x}\in S} \log Q_\phi(\mathbf{x}|S \setminus \{\mathbf{x}\}).$$

The sum of log-probabilities can be calculated as

$$\begin{aligned}&\sum_{\mathbf{x}\in S} \log Q_\phi(\mathbf{x}|S \setminus \{\mathbf{x}\})\\ =&\sum_{e=1}^{E} \#_e^S \log(\alpha_0 + \#_e^S - 1) - |S|\log(\alpha_0 + \beta_0 + |S| - 1)\\ &+ (|S| - \#_e^S)\log(\beta_0 + |S| - \#_e^S - 1),\end{aligned}$$

where $\#_e^S = |\{\mathbf{x}' \in S : \phi_e(\mathbf{x}') = 1\}|$, and α_0, β_0 are the parameters of the Beta prior.

Optimization Problem 1 is non-convex, so we apply a greedy procedure that iteratively adds the next-best transformation starting with the identity transformation, as detailed in Algorithm 1.

Algorithm 1 Greedy Transformation Composition

```
φ^0 ← id
for t = 1 ... do
  ψ̂ ← arg max_ψ Σ_{x∈S} Q_{φ^{t-1}∘ψ}(x|S \ {x})
  if Σ_{x∈S} Q_{φ^{t-1}∘ψ̂}(x|S \ {x}) < Σ_{x∈S} Q_{φ^{t-1}}(x|S \ {x})
  then
    return φ^{t-1}
  else
    φ^t ← φ^{t-1} ∘ ψ̂
  end if
end for
```

4. Parameter Estimation

In the following section, we will derive a closed-form estimator for parts of the parameters of the prior $P(\theta)$. The decisions whether to merge an element $\mathbf{x}$ with a set X' depend strongly on the prior parameters via $P(\mathbf{x})$ in Equation 7 and $P(\mathbf{x}|X')$ in Equation 6.

Heller and Ghahramani (2005) derive an EM-like algorithm that maximizes the data likelihood by iteratively finding the best clustering and then performing gradient descent on the prior parameters. This approach is computationally very expensive, since the likelihood function is not convex and the entire dataset needs to be re-clustered in each iteration.

We overcome these problems by using an alternative parametrization of the Beta distribution. This allows us to estimate half of the parameters from an unclustered set of training examples S; the other half of the parameters is pooled into a single value and adjusted by a grid search on tuning data.

We re-parametrize the Beta priors as $\alpha_e = \mu_e\sigma$ and $\beta_e = (1-\mu_e)\sigma$, where the μ_e are the prior means and σ is the common precision[1] parameter. The probability of an element *not given* any other elements of the same cluster does not depend on the prior precisions, only on the means. Hence, the means have a stronger impact on the resulting partitioning.

Imposing a Beta-distributed hyperprior on μ_e with parameters $\alpha_0 > 1$ and $\beta_0 > 1$ we can compute the Maximum-A-Posteriori estimate of the means as

$$\begin{aligned}\mu_e &= \arg\max_\mu P(\mu|S) = \arg\max_\mu \prod_{\mathbf{x}\in S} P(\phi_e(\mathbf{x})|\mu)P(\mu)\\ &= \arg\max_\mu P_{Beta}\,(\mu|\alpha_0 + |\{\mathbf{x}\in S : \phi_e(\mathbf{x}) = 1\}|,\\ &\qquad\qquad \beta_0 + |\{\mathbf{x}\in S : \phi_e(\mathbf{x}) = 0\}|)\\ &= \frac{\alpha_0 + |\{\mathbf{x}\in S : \phi_e(\mathbf{x}) = 1\}| - 1}{\alpha_0 + \beta_0 + |S| - 2}.\end{aligned}$$

5. Sequential Bayesian Clustering

In this section we discuss the task of inferring the most likely partitioning of a set of emails and present our model-based sequential clustering algorithm.

Brute-force search over the entire space of possible partitionings in Equation 1 requires the evaluation of exponentially many clusterings and is therefore intractable for reasonable numbers of emails. A more efficient approach would be to perform Markov chain Monte Carlo sampling methods like in (Williams, 2000), which yields not only the Maximum-A-Posteriori partitioning, but samples from the posterior distribution.

Approximate agglomerative clustering algorithms (Heller & Ghahramani, 2005) are more efficient. Since in practice emails have to be processed sequentially, and decisions whether an email belongs to a spam campaign cannot be revised after delivering it, we adopt the sequential clustering algorithm of Haider et al.

[1] The precision parameter of a Beta distribution is inversely related to its variance.

Algorithm 2 Model-based Sequential Clustering

$C \leftarrow \{\}$
for $t = 1 \dots n$ **do**
  $c_j \leftarrow \arg\max_{c \in C} P(\mathbf{x}^{(t)}|X_c)$
  if $P(\mathbf{x}^{(t)}|X_c) < P(\mathbf{x}^{(t)})$ **then**
    $C \leftarrow C \cup \{\{\mathbf{x}^{(t)}\}\}$
  else
    $C \leftarrow C \setminus \{c_j\} \cup \{c_j \cup \{\mathbf{x}^{(t)}\}\}$
  end if
end for
return C

(2007). This greedy incremental algorithm has the advantage of approximately finding the best campaign association of a new email in $O(n)$, where n is the number of previously seen emails, instead of taking $O(n^3)$ operations for performing a full agglomerative re-clustering.

Instead of a weighted similarity measure as in (Haider et al., 2007), our clustering model is based on a generative model. We replace the weighted sum over pairwise features by an integral over the model parameters of a cluster. This gives us the model-based sequential clustering in Algorithm 2.

In every step, the algorithm compares the hypotheses that the new element belongs to one of the existing clusters with the hypothesis that it forms its own cluster. The likelihoods of these hypotheses are calculated according to Equations 6 and 7. This greedy algorithm can be straightforwardly extended to using a non-uniform prior over clustering hypotheses, by replacing $P(\mathbf{x})$ with $P(\mathbf{x})P(C \cup \{\{\mathbf{x}^{(t)}\}\})$ and $P(\mathbf{x}^{(t)}|X_c)$ with $P(\mathbf{x}^{(t)}|X_c)P(C \setminus \{c_j\} \cup \{c_j \cup \{\mathbf{x}^{(t)}\}\})$.

6. Email Campaign Detection

In this section, we explore the behavior of the feature transformation procedure, and conduct a case study of the Bayesian clustering method for spam filtering.

In unsupervised clustering, there is no ground truth available that the output of the clustering algorithm can be compared with. Fortunately, our motivating application scenario – email spam containment – has a natural evaluation criterion: the contribution of the produced partitionings to accurate filtering.

We design an experimental setting that evaluates the Bayesian clustering solution and the feature transformation technique for the problem of detecting spam campaigns at an email service provider. Benchmark data sets such as the SpamTREC corpus are not suitable for our evaluation. A fair evaluation relies on a stream that contains realistic proportions of messages of mailing campaigns, in the correct chronological order. Benchmark corpora contain messages that have been received by users, and have therefore passed unknown filtering rules employed by the receiving server. Furthermore, benchmark data sets do not contain reliable time stamps whereas the actual chronological order is crucial.

Our experimental setting relies on a stream of spam messages received by the mail transfer agent of an email service provider. Between July and November 2008, we recorded a small fraction of spam messages, a total of 139,250 spam messages in correct chronological order. The messages have been tagged as spam because the delivering agent was listed on the Spamhaus IP block list which is maintained manually. We simulate the practical setting where one has a stream of verified spams, and one stream of unknown emails, by taking every other email from the set as training example. The rest is split into test examples (90%) and tuning examples. In order to maintain the users' privacy, we blend the stream of spam messages with an additional stream of 41,016 non-spam messages from public sources. The non-spam portion contains newsletters and mailing lists in correct chronological order as well as Enron emails and personal mails from public corpora which are not necessarily in chronological order. Every email is represented by a binary vector of 1,911,517 attributes that indicate the presence or absence of a word. The feature transformation technique introduces an additional 101,147 attributes.

6.1. Feature Transformation

In order to assess the capability of our feature transformation technique for approximating a high dimensional probability distribution, we train the transformation on an additional set S_1 of 10,000 older emails including spam and ham (*i.e.,* non-spam) messages in equal parts, and test on another set S_2 of emails of the same size. Since we cannot measure the Kullback-Leibler divergence from the true distribution directly, we measure the quantity $\frac{-1}{|S_i|} \sum_{\mathbf{x} \in S_i} \log Q_\phi(\mathbf{x}|S_i \setminus \{\mathbf{x}\})$, which is the average entropy of an email, given all other emails of the set. We compare the entropies on the training and test sets for the transformation found by the Greedy Transformation Composition algorithm to the entropy of the identity transformation. The identity transformation corresponds to an assumption of independent attributes in the input space.

In addition to the overall optimal transformation, we compute the optimal transformations ϕ^a and ϕ^x composed of only elementary transformations of the forms

ψ^a_{ij} or ψ^x_{ij}, respectively. Preliminary experiments showed that the choice of prior parameters α_0 and β_0 has negligible influence within reasonable ranges, so we report the results for $\alpha_0 = 1.1$ and $\beta_0 = 100$ in Table 1. We can see that including elementary transformations of the form ψ^x_{ij} decreases training entropy, but increases test entropy. The best transformation reduces test entropy compared to the identity transformation by about 33%. This shows that factorizing the probability over the dimensions in the image space yields a much better approximation than factorizing over the dimensions in the original feature space.

Table 1. Comparison of training and test entropies using different feature transformations.

Transformation	id	ϕ^*	ϕ^a	ϕ^x
Training entropy	1013.7	**574.1**	585.3	632.4
Test entropy	1007.1	687.5	**672.0**	720.2

6.2. Bayesian Clustering for Spam Filtering

Our evaluation protocol is as follows. We use a training window of 5,000 known spam messages, corresponding to a history of approximately 11 days. The training messages are partitioned using Algorithm 2. In each step, the clustering algorithm adds the chronologically next 100 known spam emails to the partitioning and removes the 100 oldest. We then classify the 100 next test messages. We use the odds ratio

$$\frac{\max_{X_{spam}} P(\mathbf{x}|X_{spam})}{P(\mathbf{x})}$$

as classification score, the maximum is over all spam clusters in the training window. Test messages are not added to the window of partitioned training messages.

A main difficulty in spam filtering is that ham emails can be very diverse, and it is unrealistic that one has training examples available from every region of the true distribution. We conduct experiments in two different settings that assess performance when ham emails from the test distribution are available and the performance without access to ham emails, respectively. In setting A, we train the feature transformation and parameters μ_e with 10,000 ham emails from the test distribution, and in setting B, we train on 10,000 spam messages instead.

As baseline for setting A, we use a Support Vector Machine that is trained in every step on the history of the last 5,000 spam and on the same 10,000 ham emails as the clustering method. Hence, the SVM baseline receives the same training data. In setting B, we use a one-class SVM, trained on the history of 5,000 spam messages. Additionally, we evaluate the benefit of the feature transformation by comparing with a clustering algorithm that uses the identity transformation.

An EM clustering procedure that uses a point estimates for the model parameters serves as an additional reference. We use a MAP estimate based on the same model prior used for the Bayesian model. EM requires the number of clusters to be known. We use the number of clusters that the Bayesian model identifies as input to the EM clustering.

We use two evaluation measures. Firstly, we measure the area under the ROC curve (AUC). Secondly, we use an evaluation measure that reflects the characteristics of the application more closely. An MTA has to be extremely confident when deciding to refuse a message for delivery from a contacting agent. We therefore measure the rate of true positives (spam messages identified as such) at a false positive rate of zero. We adjust the hyperparameters σ for the clustering model and C or ν for the standard SVM and one-class SVM, respectively, on the tuning set. We tune the parameters separately for optimal AUC, and for an optimal rate of true positives at a false positive rate of zero. Figure 1 shows ROC curves.

We can see that in the setting with ham emails available for training, the SVM outperforms the clustering-based filter in terms of AUC. In terms of the true positive rate at a false positive rate of zero, the clustering method outperforms the SVM classifier, by achieving a true positive rate of $0.945 \pm 9.1 \times 10^{-4}$ compared to $0.938 \pm 9.6 \times 10^{-4}$ of the SVM. The cluster-based filter shows its full strength in setting B, in the absence of non-spam training messages from the test distribution. Here, it achieves an AUC value of $0.992 \pm 2.6 \times 10^{-4}$ and a true positive rate of $0.749 \pm 1.7 \times 10^{-3}$, whereas the one-class SVM attains an AUC of $0.770 \pm 1.4 \times 10^{-3}$ and a true positive rate of $0.102 \pm 1.2 \times 10^{-3}$. That is, the Bayesian clustering method increases the true positive rate at zero false positives almost sevenfold, in the setting where no training emails from the distribution of the test hams are available. Clustering with the identity transformation as well as clustering with the EM algorithm performs worse in all settings than Bayesian clustering with the feature transformation. In setting A (ham messages from the test distribution available) with the parameters tuned for a high true positive rate at a false positive rate of zero, the EM algorithm achieves a true positive rate of only 0.04.

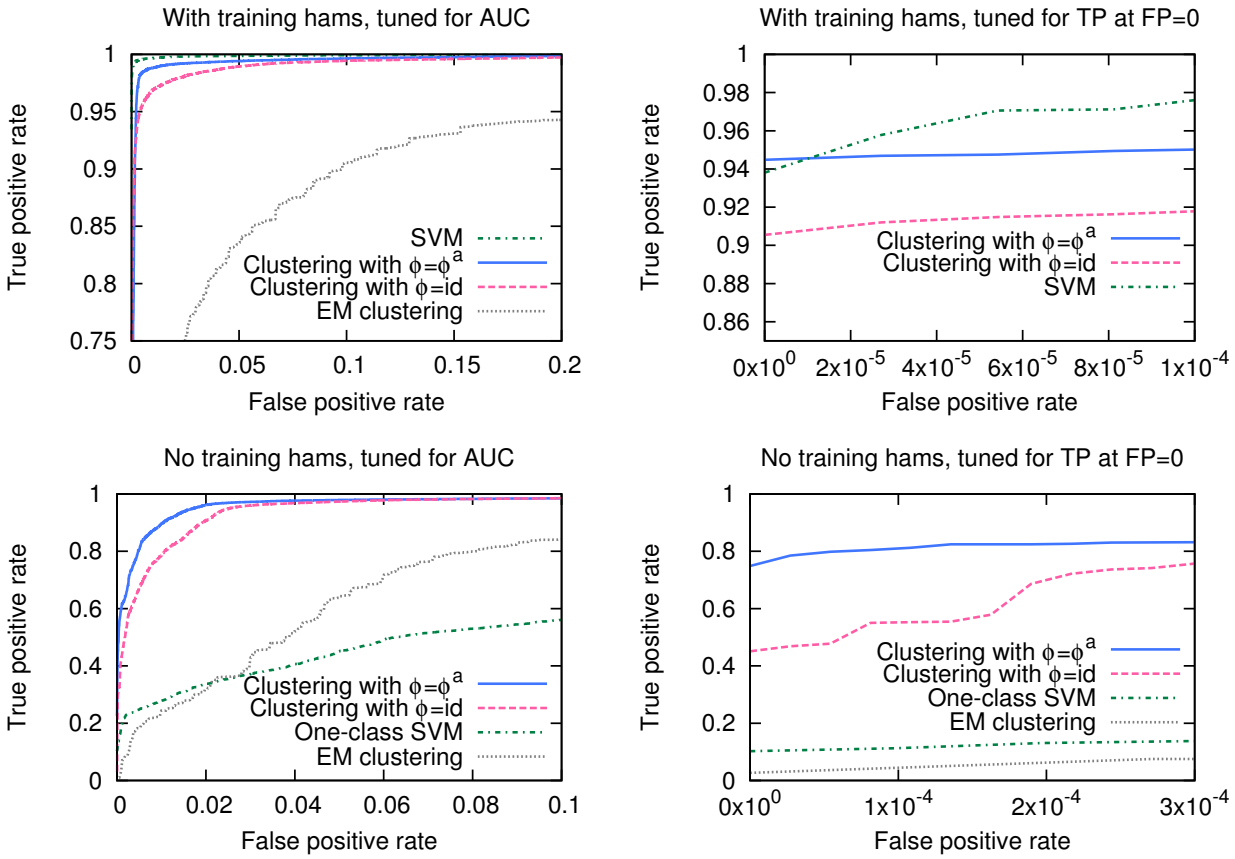

Figure 1. Evaluation of spam filtering performance.

7. Related Work

Previous work on Bayesian clustering explored in great detail the use of hierarchical priors for the cluster structure and algorithms for inference under such priors, for example (Williams, 2000), (Heller & Ghahramani, 2005), and (Lau & Green, 2007). These approaches focus on modeling hierarchical dependencies between elements, while modeling only low-level dependencies between the attributes within elements, such as Gaussian covariances. By contrast, we assume a uniform prior over the cluster structure, and instead focus on modeling arbitrary dependencies between binary attributes. We find that a non-uniform prior over partitionings is in fact not necessary, because properly taking the prior over mixture parameters $P(\theta)$ into account also prevents the trivial solution of assigning every element to its own cluster from being optimal.

Haider et al. (2007) devise a technique for mailing campaign detection that relies on training data that are manually clustered by campaigns. We find that the effort of manually partitioning training data into clusters is prohibitive in practice. Note that the effort of partitioning data is much higher than the effort of labeling data for classification because pairs of examples have to be considered.

Multi-way dependencies between attributes have been considered for instance by Zheng and Webb (2000) and Webb et al. (2005). They model the probability of an attribute vector as a product of conditional probabilities, such that each attribute can depend on multiple other attributes. If these approaches were to be used for Bayesian clustering, the number of mixture parameters would grow exponentially in the degree of dependencies. For our application, the high number of attributes renders these approaches infeasible.

Remedies for the problem of constantly changing distributions in the Spam filtering domain have been proposed in the area of adverserial learning. Teo et al. (2008) developed a formulation that allows to model test emails as modified versions of the training emails and optimize the classifier against the worst-case scenario of modifications. This approach leads to classi-

fiers that are more robust against changes of the distribution of spam emails, but still require the availability of recent spam and ham training data.

8. Conclusion

We devised a model for Bayesian clustering of binary feature vectors. The model is based on a closed-form Bayesian solution of the data likelihood in which the model parameters are integrated out. It allows for arbitrary dependencies between the input features, by transforming them into a space in which treating them as independent incurs minimal approximation error. We derived an optimization problem for learning such a transformation as a concatenation of elementary Boolean operations. In order to estimate the parameters of the prior from unlabeled data, we rewrite the parameters of the beta distribution in terms of mean values and a common variance. The mean values can be inferred in closed form from unlabeled data efficiently, the common variance constitutes a parameter that is adjusted on tuning data. We adapted a sequential clustering algorithm to use it with Bayesian clustering decisions. In a case study, we observed that the Bayesian clustering solution achieves higher true positive rates at a false positive rate of zero than an SVM. The benefit of the clustering solution is particularly visible when no non-spam training messages from the test distribution are available.

Acknowledgments

We gratefully acknowledge support from STRATO Rechenzentrum AG.

References

Haider, P., Brefeld, U., & Scheffer, T. (2007). Supervised Clustering of Streaming Data for Email Batch Detection. *Proceedings of the 24th International Conference on Machine Learning* (pp. 345–352).

Heller, K. A., & Ghahramani, Z. (2005). Bayesian hierarchical clustering. *Proceedings of the 22nd International Conference on Machine Learning* (pp. 297–304).

Lau, J., & Green, P. (2007). Bayesian Model-Based Clustering Procedures. *Journal of Computational and Graphical Statistics*, *16*, 526–558.

Teo, C., Globerson, A., Roweis, S., & Smola, A. (2008). Convex Learning with Invariances. *Advances in Neural Information Processing Systems*, *20*, 1489–1496.

Webb, G., Boughton, J., & Wang, Z. (2005). Not So Naive Bayes: Aggregating One-Dependence Estimators. *Machine Learning*, *58*, 5–24.

Williams, C. (2000). A MCMC approach to hierarchical mixture modelling. *Advances in Neural Information Processing Systems*, *12*, 680–686.

Zheng, Z., & Webb, G. (2000). Lazy Learning of Bayesian Rules. *Machine Learning*, *41*, 53–84.

Appendix

Theorem 1. *Let the implicit model parameter θ_e of the distribution $P(z_e)$ be Beta-distributed with parameters α_e and β_e for each $e \in \{1,\ldots,E\}$. Then the quantity $Q_\phi(\mathbf{x}|X')$ as defined in Equation 8 sums to at most 1 for all X' iff ϕ is injective.*

Proof. First we show indirectly that from $\forall X' : \sum_{\mathbf{z}\in\mathcal{Z}} |\{\mathbf{x} : \phi(\mathbf{x}) = \mathbf{z}\}| \prod_{e=1}^{E} P(z_e|\phi_e(X')) \leq 1$ follows $\forall \mathbf{z} : |\{\mathbf{x} : \phi(\mathbf{x}) = \mathbf{z}\}| \leq 1$. Assume that there exists a $\mathbf{z}^* \in \mathcal{Z}$ with $|\{\mathbf{x} : \phi(\mathbf{x}) = \mathbf{z}^*\}| \geq 2$. Then choose an $\mathbf{x}^*$ with $\phi(\mathbf{x}^*) = \mathbf{z}^*$. Without loss of generality, let $\forall e : \mathbf{z}^*_e = 1$. Then set $n = \max_e \left(\sqrt[E]{0.5}(\alpha_e + \beta_e) - \alpha_e \right) / \left(1 - \sqrt[E]{0.5}\right) + 1$ and $X^* = \{\mathbf{x}^*, \ldots, \mathbf{x}^*\}$ with $|X^*| = n$. It follows that

$$\prod_{e=1}^{E} P(z^*_e|\phi_e(X^*)) = \prod_{e=1}^{E} \int P(z^*_e|\theta_e) P(\theta_e|\phi_e(X^*)) d\theta$$

$$= \prod_{e=1}^{E} \frac{\alpha_e + n}{\alpha_e + \beta_e + n} > \prod_{e=1}^{E} \sqrt[E]{0.5} = 0.5,$$

and thus

$$\sum_{\mathbf{z}\in\mathcal{Z}} |\{\mathbf{x} : \phi(\mathbf{x}) = \mathbf{z}\}| \prod_{e=1}^{E} P(z_e|\phi_e(X^*))$$

$$\geq |\{\mathbf{x} : \phi(\mathbf{x}) = \mathbf{z}^*\}| \prod_{e=1}^{E} P(z^*_e|\phi_e(X^*)) > 1.$$

The opposite direction follows from the fact that $\forall X' : \sum_{\mathbf{z}\in\mathcal{Z}} \prod_{e=1}^{E} P(z_e|\phi_e(X')) \leq 1$, because $P(z_e|\phi_e(X'))$ is not an approximation, but a true Bayesian probability. Now we have

$$\forall X' : \sum_{\mathbf{x}\in\mathcal{X}} Q_\phi(\mathbf{x}|X') \leq 1$$

$$\Leftrightarrow \forall X' : \sum_{\mathbf{x}\in\mathcal{X}} \prod_{e=1}^{E} P(\phi_e(\mathbf{x})|\phi_e(X')) \leq 1$$

$$\Leftrightarrow \forall X' : \sum_{\mathbf{z}\in\mathcal{Z}} |\{\mathbf{x} : \phi(\mathbf{x}) = \mathbf{z}\}| \prod_{e=1}^{E} P(z_e|\phi_e(X')) \leq 1$$

$$\Leftrightarrow \forall \mathbf{z} : |\{\mathbf{x} : \phi(\mathbf{x}) = \mathbf{z}\}| \leq 1 \Leftrightarrow \phi \text{ is injective.}$$

□

Finding Botnets Using Minimal Graph Clusterings

Peter Haider HAIDER@CS.UNI-POTSDAM.DE
Tobias Scheffer SCHEFFER@CS.UNI-POTSDAM.DE
University of Potsdam, Department of Computer Science, August-Bebel-Str. 89, 14482 Potsdam, Germany

Abstract

We study the problem of identifying botnets and the IP addresses which they comprise, based on the observation of a fraction of the global email spam traffic. Observed mailing campaigns constitute evidence for joint botnet membership, they are represented by cliques in the graph of all messages. No evidence against an association of nodes is ever available. We reduce the problem of identifying botnets to a problem of finding a minimal clustering of the graph of messages. We directly model the distribution of clusterings given the input graph; this avoids potential errors caused by distributional assumptions of a generative model. We report on a case study in which we evaluate the model by its ability to predict the spam campaign that a given IP address is going to participate in.

1. Introduction

We address the problem of identifying botnets that are capable of exploiting the internet in a coordinated, distributed, and harmful manner. Botnets consist of computers that have been infected with a software virus which allows them to be controlled remotely by a botnet operator. Botnets are used primarily to disseminate email spam, to stage distributed denial-of-service (DDoS) attacks, and to harvest personal information from the users of infected computers (Stern, 2008).

Providers of computing, storage, and communication services on the internet, law enforcement and prosecution are interested in identifying and tracking these threats. An accurate model of the set of IP addresses over which each existing botnet extends would make it possible to protect services against distributed denial-of-service attacks by selectively denying service requests from the nodes of the offending botnet.

Evaluating botnet models is difficult, because the ground truth about the sets of IP addresses that constitute each botnet at any given time is entirely unavailable (Dittrich & Dietrich, 2008). Many studies on botnet identification conclude with an enumeration of the conjectured number and size of botnets (Zhuang et al., 2008). Reliable estimates of the current size of one particular botnet require an in-depth analysis of the communication protocol used by the network. For instance, the size of the Storm botnet has been assessed by issuing commands that require all active nodes to respond (Holz et al., 2008). However, once the communication protocol of a botnet is understood, the botnet is usually taken down by law enforcement, and one is again ignorant of the remaining botnets.

We develop an evaluation protocol that is guided by the basic scientific principle that a model has to be able to predict future observable events. We focus on email spam campaigns which can easily be observed by monitoring the stream of messages that reach an email service provider. Our evaluation metric quantifies the model's ability to predict which email spam campaign a given IP address is going to participate in.

Previous studies have employed clustering heuristics to aggregate IP addresses that participated in joint campaigns into conjectured botnets (Xie et al., 2008; Zhuang et al., 2008). Because an IP address can be a part of multiple botnets during an observation interval, this approach is intrinsicaly inaccurate. The problem is furthermore complicated as it is possible that a botnet processes multiple campaigns simultaneously, and multiple botnets may be employed for large campaigns. We possess very little background knowledge about whether multiple networks, each of which has been observed to act in a coordinated way, really form one bigger, joint network. Also, distributional assumptions about the generation of the observable events are very hard to motivate. We address this lack of prior knowledge by directly modeling the conditional distri-

Appearing in *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012.

bution of clusterings given the observable data, and by searching for minimal clusterings that refrain from merging networks as long as empirical evidence does not render joint membership in a botnet likely.

Other studies have leveraged different types of data in order to identify botnets. For example Mori et al. (2010); DiBenedetto et al. (2010) record and cluster fingerprints of the spam-sending hosts' TCP behavior, exploiting that most bot types use their own protocol stacks with unique characteristics. Yu et al. (2010) identify bot-generated search traffic from query and click logs of a search engine by detecting shifts in the query and click distributions compared to a background model. Another angle to detect bots is to monitor traffic from a set of potentially infected hosts and find clusters in their outgoing and incoming packets (Gu et al., 2008; John et al., 2009); for example, DNS requests of bots used to connect to control servers (Choi et al., 2009) or IRC channel activity (Goebel & Holz, 2007). The major difference here is that access to *all* the traffic of the hosts is required, and thus these methods only work for finding infected hosts in a network under one's control.

The rest of this paper is organized as follows. In Section 2, we discuss our approach to evaluating botnet models by predicting participation in spamming campaigns. In Section 3, we establish the problem of minimal graph clustering, devise a probabilistic model of the conditional distribution of clusterings given the input graph, and derive a Gibbs sampler. Section 4 presents a case study that we carried out with an email service provider. Section 5 concludes.

2. Problem Setting and Evaluation

The ground truth about the sets of IP addresses that constitute each botnet is unavailable. Instead, we focus on the botnet model's ability to predict observable events. We consider email spam campaigns which are one of the main activities that botnets are designed for, and which we can easily observe by monitoring the stream of emails that reach an email service provider. Most spam emails are based on a campaign template which is instantiated at random by the nodes of a botnet. Clustering tools can identify sets of messages that are based on the same campaign template with a low rate of errors (Haider & Scheffer, 2009). A single campaign can be disseminated from the nodes of a single botnet, but it is also possible that a botnet processes multiple campaigns simultaneously, and multiple botnets may be employed for large campaigns.

We formalize this setting as follows. Over a fixed period of time, n messages are observed. An adjacency matrix X of the graph of messages reflects evidence that pairs of messages originate from the same botnet. An edge between nodes i and j—represented by an entry of $X_{ij} = 1$—is present if messages i and j have been sent from the same IP address within the fixed time slice, or if a campaign detection tool has assigned the messages to the same campaign cluster. Both types of evidence are uncertain, because IP addresses may have been reassigned within the time slice and the campaign detection tool may incur errors. The absence of an edge is only very weak and unreliable evidence against joint botnet membership, because the chance of not observing a link between nodes that are really part of the same botnet is strongly dependent on the observation process.

The main part of this paper will address the problem of inferring a reflexive, symmetric *edge selector matrix* Y in which entries of $Y_{ij} = 1$ indicate that the messages represented by nodes i and j originate from the same botnet. The transitive closure Y^+ of matrix Y defines a clustering C_Y of the nodes. The clustering places each set of nodes that are connected to one another by the transitive closure Y^+ in one cluster; the clustering is the union of clusters:

$$C_Y = \bigcup_{i=1}^{n} \{\{j : Y^+_{ij} = +1\}\}. \quad (1)$$

Because Y^+ is reflexive, symmetric and transitive, it partitions all nodes into disjoint clusters; that is, $c \cap c' = \emptyset$ for all $c, c' \in C_Y$, and $\bigcup_{c \in C_Y} c = \{1, \ldots, n\}$.

An unknown process generates future messages which are characterized by two observable and one latent variable. Let the multinomial random variables s indicate the campaign cluster of a newly received message, a indicate the IP address, and let latent variable c indicate the originating cluster, associated with a botnet. We quantify the ability of a model C_Y to predict the observable variable s of a message given a in terms of the likelihood

$$P(s|a, C_Y) = \sum\nolimits_c P(s|c, C_Y) P(c|a, C_Y). \quad (2)$$

Equation 2 assumes that the distribution over campaigns is conditionally independent of the IP address given the botnet; that is, botnet membership alone determines the distribution over campaigns.

Multinomial distribution $P(s|c, C_Y)$ quantifies the likelihood of campaign s within the botnet c. It can be estimated easily on training data because model C_Y fixes the botnet membership of each message. Multinomial distribution $P(c|a, C_Y)$ quantifies the probability that IP address a is part of botnet c given model

C_Y. Model C_Y assigns each node—that is, message—to a botnet. However, an address can be observed multiple times within the fixed time slice, and the botnet membership can change withing the time interval. Hence, a multinomial distribution $P(c|a, C_Y)$ has to be estimated for each address a on the training data, based on the model C_Y. Note that at application time, $P(c|a, C_Y)$ and hence the right hand side of Equation 2 can only be determined for addresses a that occur in the training data on which C_Y has been inferred.

3. Minimal Graph Clustering

Let X be the adjacency matrix of the *input graph* with n nodes. Entries of $X_{ij} = 1$ indicate an edge between nodes i and j which constitutes uncertain evidence for joint membership of these nodes in a botnet. The input matrix is assumed to be reflexive ($X_{ii} = 1$ for all i), and symmetric ($X_{ij} = X_{ji}$).

The *outcome* of the clustering process is represented by a reflexive, symmetric *edge selector matrix* Y in which entries of $Y_{ij} = 1$ indicate that nodes i and j are assigned to the same cluster, which indicates that the messages originate from the same botnet. The transitive closure Y^+ of matrix Y defines a clustering C_Y of the nodes according to Equation 1. Intuitively, the input matrix X can be thought of as data, whereas output matrix Y should be thought of as the model that encodes a clustering of the nodes. A trivial baseline would be to use X itself as edge selector matrix Y. In our application, this would typically lead to all messages being grouped in one single cluster.

No prior knowledge is available on associations between botnets in the absence of empirical evidence. If the adjacency matrix X does not contain evidence that links nodes i and j, there is no justification for grouping them into the same cluster. This is reflected in the concept of a *minimal edge selector matrix*.

Definition 1. *A selector matrix Y and, equivalently, the corresponding graph clustering C_Y, is* minimal with respect to adjacency matrix X *if it satisfies*

$$Y = Y^+ \circ X, \tag{3}$$

where $(Y^+ \circ X)_{ij} = Y^+_{ij} X_{ij}$ is the Hadamard product that gives the intersection of the edges of Y^+ and X.

Intuitively, for every pair of nodes that are connected by the adjacency matrix, selector matrix Y decides whether they are assigned into the same cluster. Nodes that are not connected by the adjacency matrix X must not be linked by Y, but can still end up in the same cluster if they are connected by the transitive closure Y^+. Equation 3 also ensures that the transitive closure Y^+ does not differ from Y for any pair of nodes i, j that are connected by the adjacency matrix. This enforces that no two different minimal selector matrices have the same transitive closures and therefore induce identical clusterings, which would inflate the search space.

3.1. Probabilistic Model

This section derives a probabilistic model for the minimal graph clustering problem. Its most salient property is that it is not based on a generative model of the graph, but instead directly models the conditional probability of the clustering given the adjacency matrix X. This circumnavigates systematic errors caused by inaccurate distributional assumptions for the generation of the adjacency matrix X.

We define the posterior distribution over all reflexive and symmetric matrices Y that are minimal with respect to X.

Definition 2. *Let $X \in \{0,1\}^{n \times n}$ be a reflexive and symmetric adjacency matrix. Then, $\mathcal{Y}_X \subseteq \{0,1\}^{n \times n}$ is the set of matrices that are reflexive, symmetric, and minimal with respect to X.*

In our application, each node is an element of at most two cliques because each message is connected to all other messages that have been sent from the same IP address, and to all other messages that match the same campaign template. If a template or an address has been observed only once, either of these cliques may resolve to just the node itself. Let Q_X denote the set of cliques in X, and let C^q_Y be the projection of clustering C_Y to the elements of $q \in Q_X$. Within each clique $q \in Q_X$, any clustering C^q_Y is minimal with respect to X because $X_{ij} = 1$ for all $i, j \in q$, and therefore any reflexive, symmetric, and transitive clustering of q is possible. We model the probability distribution over clusterings of each clique $q \in Q_X$ as a Chinese Restaurant process (Pitman & Picard, 2006) with concentration parameter $\alpha_q > 0$:

$$P(C^q_Y|\alpha_q, n_q) = \alpha_q^{|C^q_Y|} \frac{\Gamma(\alpha_q)}{\Gamma(\alpha_q + n_q)} \prod_{c \in C^q_Y} \Gamma(|c|). \tag{4}$$

Equation 5 now defines the distribution over all partition matrices $Y \in \mathcal{Y}_X$ as a product over all cliques in Q_X, where the clique specific concentration parameters are collected into $\boldsymbol{\alpha} = \{\alpha_q : q \in Q_X\}$.

$$P(C_Y|X, \boldsymbol{\alpha}) \propto \begin{cases} \prod_{q \in Q_X} P(C^q_Y|\alpha_q, n_q) & \text{if } Y \in \mathcal{Y}_X \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Equation 5 can be seen in analogy to the factorization of the posterior over cliques in conditional random fields. However, because the minimality property has non-local effects on the possible values that edges can assume, this factorization is not equivalent to the assumption of the Markov property on which the factorization theorem for random fields is based (Hammersley & Clifford, 1971).

Normalization of Equation 5 is computationally intractable because it requires the enumeration of the elements of $\mathcal{Y}_X$. However, the Gibbs sampler that we will derive in the following only has to normalize over all values of the random variables that are reassigned in each step of the sampling process.

3.2. Inference

Computing the posterior distribution $P(C_Y|X, \boldsymbol{\alpha})$ is a generalization of the inference problem for conventional Chinese Restaurant process models. When all entries of X are one, the graph has only one clique and the special case of a Chinese Restaurant process is obtained. In this case, depending on the concentration parameter, the outcome may be one single cluster of all nodes. Maximization of the posterior as well as full Bayesian inference are intractable even for this special case because of the non-convexity of the posterior and the exponential number of possible clusterings. Hence, in this section we describe a Gibbs sampler that generates unbiased samples from the posterior.

Algorithm 1 Assignment space $\mathcal{Y}_i^Y$ for Gibbs sampler

Input: Current partitioning matrix Y
1: let $q_1, \ldots, q_k$ be the cliques with element i
2: let $\mathcal{Y}_i = \emptyset$
3: **for all** combinations $c_1 \in C_Y^{q_1} \cup \{\{i\}\}$, $\ldots$, $c_k \in C_Y^{q_k} \cup \{\{i\}\}$ **do**
4: let $Y'_{-i} = Y_{-i}$
5: let $Y'_{il} = Y'_{li} = 1$ if and only if $l \in c_j$ for any j
6: **if** $(Y'_{-i})^+ = Y'_{-i}$ **then**
7: add Y' to $\mathcal{Y}_i^Y$
8: **else**
9: discard Y'
10: **end if**
11: **end for**
Return: $\mathcal{Y}_i^Y$, all reflexive, symmetric, minimal partitioning matrices derived from Y by reassigning Y_i.

Gibbs samplers divide the set of random variables into smaller subsets and iteratively draw new values for one subset given the values of the remaining variables. For the observations to form an unbiased sample, the random variables have to be partitioned such that the sequence of assignments formes an ergodic Markov chain; that is, each state has to be reachable from each other state. In our case, perhaps the most obvious-seeming approach would be to factor the posterior over individual edges. However, since many matrices Y violate the minimality condition, the chain of alterations of single matrix entries would not in general be ergodic.

Therefore, we devise a sampling algorithm that jointly samples the i-th row and column (the i-th row and column are identical because Y is symmetric). Let Y'_i refer to the i-th row and column of the new matrix Y', and let $Y'_{-i} = Y_{-i}$ refer to the remaining matrix entries, such that $Y' = Y'_i \cup Y'_{-i}$. Equation 6 expands the definition of the conditional probability; Equation 7 factorizes over the cliques, according to Equation 5. Equation 8 omits all terms that are constant in Y'_i: the denominator, and all cliques in which node i does not occur. Normalization of the right hand side of Equation 8 is now over all values for Y'_i that render Y' reflexive, symmetric, and minimal with respect to X.

$$P(Y'_i|Y_{-i}, X, \boldsymbol{\alpha}) = \frac{P(Y'_i, Y'_{-i}|X, \boldsymbol{\alpha})}{P(Y_{-i}|X, \boldsymbol{\alpha})} \quad (6)$$

$$\propto \frac{\prod_{q \in Q_X} P(C^q_{Y'}|\alpha_q, n_q)}{P(Y_{-i}|X, \boldsymbol{\alpha})} \quad (7)$$

$$\propto \prod_{q \in Q_X : i \in q} P(C^q_{Y'}|\alpha_q, n_q) \quad (8)$$

The main computational challenge here is to determine the set $\mathcal{Y}_i^Y$ of reflexive, symmetric, minimal matrices that can be derived from Y by changing row and column i. Since Equation 8 has to be normalized, all of its elements have to be enumerated. An obvious but inefficient strategy would be to enumerate all up to 2^n assignments of Y_i and test the resulting matrix for reflexivity, symmetry, and minimality.

However, most values of Y'_i violate minimality and need not be enumerated. Algorithm 1 constructs the set $\mathcal{Y}_i^Y$ in $O(n^k)$, where k is the maximal number of cliques that each node is a member of. In our application, each node is an element of up to two cliques—the set of messages with a shared IP address, and the set of messages that follow the same campaign template. Hence, in our case, the algorithm has a worst-case execution time of $O(n^2)$. In most cases, the number of clusters in each of the two cliques is much lower than n, and thus much fewer than n^2 cases are considered.

Theorem 1. *Given an adjacency matrix X and an edge selector matrix Y, Algorithm 1 constructs a set $\mathcal{Y}_i^Y$ that contains all $Y' = Y'_i \cup Y_{-i}$ which are reflexive, symmetric, and minimal with respect to X. The execution time of Algorithm 1 is in $O(n^k)$ when each node is a member of at most k cliques in X.*

Proof. Let node i be an element of cliques $q_1, \ldots, q_k$. On these cliques, the current partitioning matrix Y induces clusterings $C_Y^{q_1}, \ldots, C_Y^{q_k}$ with at most n clusters each. When Y'_i links node i to more than one cluster from any $C_Y^{q_l}$, then by the definition of a clustering in Equation 1 these clusters are merged in $C_{Y'}$. However, when Y'_i links node i to two clusters with at least one other element in q_l each, say j and l with $Y_{jl} = 0$, the transitive closure Y'^{+} has to add at least an edge to Y'_{-i} that links j and l. Since j and l are in clique q_l, they have to be connected by the adjacency matrix, $X_{jl} = 1$. But $Y'_{jl} = 0$, $Y'^{+}_{jl} = 1$ and $X_{jl} = 1$ violates the minimality condition defined in Equation 3. Therefore, Y' must only merge clusters that have elements in different cliques, and so at most n^k combinations of clusters can lead to minimal matrices Y' when merged. Reflexivity and symmetry of Y' follow from reflexivity and symmetry of X. The execution time is dominated by the enumeration of all n^k many combinations of clusters in Line 3. □

The Gibbs sampler iteratively samples Y^{t+1} according to $P(Y'_{i_t}|Y^t_{-i_t}, X, \boldsymbol{\alpha})$, given by Equation 8. Each Y^{t+1} is created from the predecessor by cycling over the rows that are resampled—that is, $i_t = t \mod n$. The conditional is defined over the set $\mathcal{Y}_{i_t}^{Y^t}$. We will now argue that a sequence of matrices created by the Gibbs sampler is an ergodic Markov chain.

Theorem 2. *For $\alpha_q > 0$, the sequence $Y^0, \ldots, Y^T$ with $Y^{t+1} \sim P(Y^{t+1}_{(t \mod n)}|Y^t_{-(t \mod n)}, X, \boldsymbol{\alpha})$ is an ergodic Markov chain.*

Proof. The sequence is a Markov chain because each element is sampled from a distribution that is parameterized only with the preceding matrix and the row that is to be resampled. For it to be ergodic we have to prove that from any state Y, every other state Y' can be reached. With the $\alpha_q > 0$, Equation 5 is positive for all states in $\mathcal{Y}_X$ that are reflexive, symmetric, and minimal with respect to X. In each step the sampler can only change row and column i. Hence, any chain of states with $Y^{t+1} \in \mathcal{Y}^{Y^t}_{(t \mod n)}$ can be reached because by Theorem 1, all elements of $\mathcal{Y}^{Y^t}_{(t \mod n)}$ are reflexive, symmetric, minimal with respect to X and differ from Y^t only in row and column i.

To begin with, we argue that from any state Y the identity matrix I can be reached which connects each node only to itself. To prove this, it suffices to show that for any i and any Y, a state $I^{Y,i}$ with ${I^{Y,i}}_{ii} = 1$ for all i, ${I^{Y,i}}_{ij} = 0$ for all $j \neq i$, and ${I^{Y,i}}_{jk} = Y_{jk}$ for all $j, k \neq i$ can be reached directly from state Y by sampling row and column Y_i. By the definition of $\mathcal{Y}_i^Y$, the Gibbs sampler can directly reach state $I^{Y,i}$ from Y if $I^{Y,i}$ is symmetric, reflexive, minimal with respect to X, and differs from Y only in the i-th row and column. By its definition, it is clear that $I^{Y,i}$ differs from Y only in the i-th column and row, and that it is reflexive. Since Y is symmetric and the i-th row and column of $I^{Y,i}$ are identical, $I^{Y,i}$ has to be symmetric as well. It remains to be shown that $I^{Y,i} = I^{Y,i^+} \circ X$. We split the proof of this claim into two parts. First, we show that the i-th row and column of $I^{Y,i}$ are equal to the i-th row and column of $I^{Y,i^+} \circ X$. Intuitively, because $I^{Y,i}$ connects node i only to itself, the transitive closure adds nothing, and the Hadamard product has no effect because X is reflexive. Formally, this can be shown via an inductive proof along the following construction of the transitive closure of $I^{Y,i}$. Let $R^0 = I^{Y,i}$. For all $l > 0$, let $R^l_{ij} = 1$ if $R^{l-1}_{ij} = 1$ or if there is a k such that $R^{l-1}_{ik} = 1$ and $R^{l-1}_{kj} = 1$; otherwise, $R^l_{ij} = 0$. When R^{l-1} contains an open triangle of edges $R^{l-1}_{ik} = 1$ and $R^{l-1}_{kj} = 1$, then R^l is defined to add an edge $R^l_{ij} = 1$. Then the limit $\lim_{l\to\infty} R^l$ is the transitive closure $(I^{Y,i})^+$. Now inductively, if for all $j : R^{l-1}_{ij} = 0$, then for all $j : R^l_{ij} = 0$, and from ${I^{Y,i}}_i = 0$ it follows that ${I^{Y,i^+}}_i = {I^{Y,i}}_i$, and $(I^{Y,i^+} \circ X)_i = {I^{Y,i}}_i$.

Secondly, we show that all elements in I^{Y,i^+} *except* the i-th row and column remain unchanged from Y^+: From the monotonicity of the transitive closure operator and $I^{Y,i} \leq Y$ it follows that $(I^{Y,i^+} \circ X)_{-i} \leq (Y^+ \circ X)_{-i}$. Furthermore, since the transitive closure operator only adds positive edges, $(I^{Y,i^+} \circ X)_{-i} \geq (I^{Y,i} \circ X)_{-i} = (Y \circ X)_{-i}$, which is in turn equal to $(Y^+ \circ X)_{-i}$ because Y itself is minimal with respect to X. Both inequalities together give us $(I^{Y,i^+} \circ X)_{-i} = (Y^+ \circ X)_{-i}$, and because ${I^{Y,i}}_{-i} = Y_{-i}$ we have that ${I^{Y,i}}_{-i} = (I^{Y,i^+} \circ X)_{-i}$. Together with the first part finally $I^{Y,i} = I^{Y,i^+} \circ X$.

This establishes that $I^{Y,i}$ can be reached by the Gibbs sampler from any state Y for any i, and thus by repeatedly using this state transition for *all* i, I is reachable. The reachability relation is symmetric because Y^{t+1} is constructed from Y^t by reassigning one column and row which can be reversed, and Y^t is required to be in $\mathcal{Y}_X$, and therefore can be reached from Y^{t+1}. Hence, from any state Y, every other state Y' can be reached via the state I, and ergodicity holds. □

3.3. Prediction

The Gibbs sampler creates a chain $Y^0, \ldots, Y^T$ of matrices, governed by the posterior $P(Y|X, \boldsymbol{\alpha})$. In order to predict which campaign s a given IP address a will participate in, we can approximate the Bayesian infer-

ence of c (Equation 9) using the chain (Equation 10).

$$P(s|a, X, \boldsymbol{\alpha}) = \sum_{Y \in \mathcal{Y}_X} P(C_Y|X, \boldsymbol{\alpha}) P(s|a, C_Y) \quad (9)$$

$$\approx \sum_{Y \in \{Y^0, \ldots, Y^T\}} P(s|a, C_Y) \quad (10)$$

Equation 1 decomposes $P(s|a, C_Y)$ into two multinomial distributions that can be estimated from the available data.

4. Case Study

In this section, we conduct a case study on botnet detection. Since the gound truth about which botnets are currently active and which hosts they are composed of is not available, we evaluate the model in terms of its accuracy of predicting which spam campaign a given IP address will participate in.

We record incoming spam emails over a period of 11 days in January 2012 at a large email service provider. We select only emails that have been blacklisted on the grounds of three content-based filtering techniques: The first is a set of manually maintained regular expressions, each tailored to match against all spams of one particular campaign. The second is a list of semi-automatically generated, campaign-specific feature sets (Haider & Scheffer, 2009). A feature set consists of words and structure flags and is the intersection of all previously observed emails from the campaign. The third is a blacklist of URLs that spam emails link to. Thus, we have a reliable partitioning of all emails into spam campaigns.

We exclude IP addresses of known legitimate forwarding servers that relay inbound emails according to their users' personal filtering policies. To this end, we track the IP address from the last hop in the transmission chain. If the address has a valid reverse DNS entry that matches a domain from a list of well-known email service providers, we omit the message.

Equation 5 allows for individual values of the concentration parameters α_q for each clique q in the email graph X. We use two distinct values: a value of α_a for all cliques that share a joint IP address, and a value of α_s for all cliques that match a joint campaign. Parameters α_a and α_s are tuned to maximize the AUC metric on the data recorded on the first day. The data of the remaining ten days is then used for evaluation. Within each day, the Gibbs sampler infers a chain of clusterings on the data of the first 16 hours. The emails of the last 8 hours with a sender IP address that has previously occurred are used as test data. Emails from IP addresses that have not been seen before are excluded, since no informed decision can be made for them. The proportion of IP addresses that have not previously been observed depends on the proportion of the global email traffic that the server gets to observe. Also, we exclude emails from campaigns that appear less than 100 times. In total, this data collection procedure results in 701,207 unique pairs of campaigns and IP addresses in the training sets and 71,528 in the test sets. Each test email serves as a positive example for its campaign and a negative example for all other campaigns.

4.1. Reference Methods

We compare the Minimal Graph Clustering model to three baselines. The first, *threshold-based* baseline is an agglomerative clustering algorithm based on a threshold heuristic, adapted from Zhuang et al. (2008). It operates on the assumption that each campaign is sent by only one botnet. Initially, every campaign constitutes its own cluster. Clusters c and c' are greedily merged if their fraction of overlapping IP addresses exceeds a threshold. This fraction is defined as

$$\frac{\sum_{i \in c} \mathbb{I}(\exists j \in c' : s_i = s_j)}{2|c|} + \frac{\sum_{j \in c'} \mathbb{I}(\exists i \in c : s_j = s_i)}{2|c'|},$$

where $\mathbb{I}$ is the indicator function and s_i the campaign of the i-th email. Given a clustering C of emails, $P(s|a, C) = \sum_{c \in C} P(s|c, C) P(c|a, C)$ is inferred after multinomial distributions $P(s|c)$ and $P(c|a, C)$ have been estimated on the training data. The clustering threshold is tuned for performance on the first day.

The second baseline is *spectral clustering*, where we tune the number of clusters and similarity values for emails with matching campaign or IP address. We use the implementation of Chen et al. (2011).

The third baseline is a straightforward generative clustering model for email graphs with a Chinese Restaurant process prior and a likelihood function that factorizes over the edges of the email graph X, assuming independence for the edges in X. The likelihood function has a set of four parameters $\boldsymbol{\theta} = \{\theta_s^{in}, \theta_s^{in}, \theta_a^{out}, \theta_a^{out}\}$ that quantify the probability of the presence of a link when the nodes are and are not elements of a joint botnet. The likelihood for an edge that connects two emails from the same campaign is given as

$$P(X_{ij} = 1|C, \boldsymbol{\theta}) = \begin{cases} \theta_s^{in}, & \text{if } C(i) = C(j) \\ \theta_s^{out}, & \text{if } C(i) \neq C(j), \end{cases}$$

where $C(i)$ denotes the cluster that clustering C assigns email i to. The likelihood of an edge between two emails from the same IP address is defined analogously,

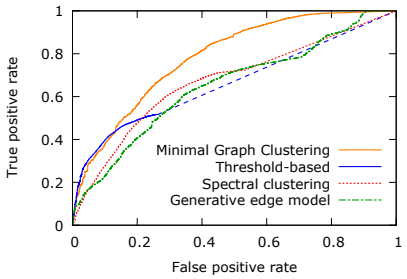

Figure 1. ROC-curves for campaign prediction.

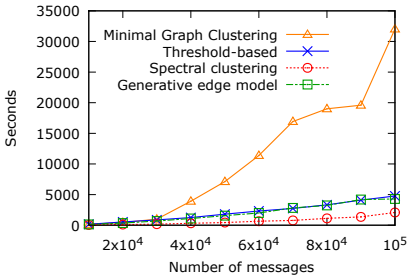

Figure 2. Execution times for inferring the clustering.

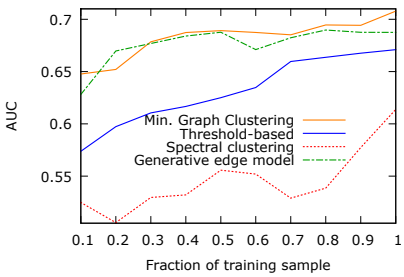

Figure 3. AUC depending on training set size.

using the paramters θ_a^{out} and θ_a^{out}. The joint probability of the email matrix and a clustering C is then given as $P(X, C|\alpha, \boldsymbol{\theta}) = P_{CRP}(C|\alpha) \prod_{i,j<i} P(X_{ij}|C, \boldsymbol{\theta})$. The parameters are adjusted using gradient-ascent on the joint data likelihood.

4.2. Results

We measure ROC curves; IP addresses for which the likelihood $P(s|a, X, \boldsymbol{\alpha})$ of the correct campaign exceeds the threshold are counted as true positives, addresses for which the likelihood of an incorrect campaign exceeds the threshold as false positives. Figure 4 shows ROC curves for the four methods under study.

Minimal graph clustering attains the highest area under the ROC curve of 0.784, compared to 0.675 for threshold-based clustering, 0.671 for spectral clustering, and 0.651 for the generative edge model baseline. The threshold-based method is marginally more accurate than the Minimal Graph Clustering model for low threshold values, and less accurate for all other threshold values. Threshold-based clustering infers $P(s|c)$ and $P(c|a, C)$ and consequently $P(s|a, C)$ to be zero for many values of s and a. Therefore, a large interval of points on the ROC curve cannot be attained by any threshold value; this is indicated by a dashed line.

Typically, the number of requests during a DDoS attack exceeds the capacity to serve requests by far. An unprotected system will serve only a small, random fraction of requests and will be unable to serve all others; this amounts to a false positive rate of close to one. In order to defend against such an attack, one has to select a small proportion of requests which can be served. Therefore, in defending against DDoS attacks, the right hand side of the ROC curve that allows high true positive rates is practically relevant.

Figure 4 shows execution times for running all four methods until convergence depending on the number of examples. The Minimal Graph Clustering model is computationally more expensive than the baselines. For continuously maintaining a clustering that subsequently incorporates newly available messages, it is thus advisable to use the previous clustering as a starting point of the Gibbs sampler in order to reduce the number of necessary iterations until convergence.

Figure 4 shows area under ROC curve depending on what fraction of the training sample is used. For testing, only emails with IP addresses that are present in the smallest subset are used. The plots indicate that having access to a larger sample of the overall email traffic could increase performance considerably.

5. Conclusion and Discussion

The identification of spam-disseminating botnets can be reduced to the problem of clustering the graph of email messages in which messages are linked if they originate from the same IP address or match the same campaign template. We devised a probabilistic model that directly describes the conditional probability of a clustering given the input graph without making distributional assumptions about the generation of the observable data. We derived a Gibbs sampler; we showed that resampling rows and edges of the output matrix creates an ergodic Markov chain, and that each sampling step can be carried out in $O(n^2)$. We argue that botnet models can be evaluated in terms of their ability to predict which spam campaign a given IP address is going to participate in. From a case study carried out with an email service provider we conclude that the minimal graph clustering model outperforms a number of reference methods—spectral clustering, a generative model, and a threshold-based, agglomerative clustering model—in terms of its area under the ROC curve.

The botnet model draws a picture of the current size

and activity of botnets. From the IP addresses, the geographical distribution of each botnet can be derived. The botnet model can be used to select particularly prolific botnets for in-depth analysis and possible legal action. Widespread botnet software is versatile and supports both, dissemination of email spam and the staging of network attacks (Stern, 2008). When both, a mailing campaign and a network attack are carried out by a single network within the typical IP-address reassignment interval of one day, then the botnet model which has been trained on email data can score HTTP requests by the likelihood that their sender IP address is part of an attacking botnet. This allows to prioritize requests and to maintain a service during an attack. Alternatively, the botnet model can be trained with HTTP requests instead of emails; the recipient domain of an HTTP request plays the role of the campaign template. Again, the botnet model allows to infer the likelihood that an individual sender IP address acts as part of an attacking botnet.

Direct evaluation of the model's ability to decide whether an IP request is part of a network attack would require evaluation data in the form of a collection of individual HTTP requests labeled with the botnet that has sent the request. While it is relatively easy to collect the entire stream of legitimate and attacking HTTP requests that reach a domain during an attack, there is no practical means of labeling individual requests. In general, HTTP requests contain no information that allows even a human expert to decide whether a request is part of an attack, let alone which botnet a request has really been sent from.

6. Acknowledgments

This work was funded by a grant from STRATO AG.

References

Chen, Wen-Yen, Song, Yangqiu, Bai, Hongjie, Lin, Chih-Jen, and Chang, Edward Y. Parallel spectral clustering in distributed systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):568–586, 2011.

Choi, H., Lee, H., and Kim, H. Botgad: detecting botnets by capturing group activities in network traffic. In *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*, pp. 2. ACM, 2009.

DiBenedetto, S., Gadkari, K., Diel, N., Steiner, A., Massey, D., and Papadopoulos, C. Fingerprinting custom botnet protocol stacks. In *Secure Network Protocols (NPSec), 2010 6th IEEE Workshop on*, pp. 61–66. IEEE, 2010.

Dittrich, D. and Dietrich, S. Discovery techniques for p2p botnets, 2008.

Goebel, J. and Holz, T. Rishi: Identify bot contaminated hosts by irc nickname evaluation. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pp. 8–8. USENIX Association, 2007.

Gu, G., Perdisci, R., Zhang, J., and Lee, W. Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *Proceedings of the 17th conference on Security symposium*, pp. 139–154. USENIX Association, 2008.

Haider, P. and Scheffer, T. Bayesian clustering for email campaign detection. In *Proceeding of the International Conference on Machine Learning*, 2009.

Hammersley, J. and Clifford, P. Markov fields on finite graphs and lattices, 1971.

Holz, T., Steiner, M., Frederic, D., Biersack, E., and Freiling, F. Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. In *Proceedings of the USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2008.

John, J.P., Moshchuk, A., Gribble, S.D., and Krishnamurthy, A. Studying spamming botnets using botlab. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, pp. 291–306. USENIX Association, 2009.

Mori, T., Esquivel, H., Akella, A., Shimoda, A., and Goto, S. Understanding large-scale spamming botnets from internet edge sites. In *Proceedings of the Conference on E-Mail and Anti-Spam*. CEAS, 2010.

Pitman, J. and Picard, J. *Combinatorial stochastic processes*. Springer, 2006. ISBN 354030990X.

Stern, H. A survey of modern spam tools. In *Proceedings of the Conference on Email and Anti-Spam*, 2008.

Xie, Y., Yu, F., Achan, K., Panigrahy, R., Hulten, G., and Osipkov, I. Spamming botnets: Signatures and characteristics. *ACM SIGCOMM Computer Communication Review*, 38(4):171–182, 2008.

Yu, F., Xie, Y., and Ke, Q. Sbotminer: Large scale search bot detection. In *Proceedings of the third ACM international conference on Web search and data mining*, pp. 421–430. ACM, 2010.

Zhuang, L., Dunagan, J., Simon, D.R., Wang, H.J., and Tygar, JD. Characterizing botnets from email spam records. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, pp. 1–9. USENIX Association, 2008.

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 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

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

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

$R^*(\mathbf{c})$ 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.

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

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*

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

subject to the constraints

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

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

$$\begin{aligned}
&a\mathbf{K}(\bar{p})\bar{\mathbf{c}} - \nabla R^*(\bar{\mathbf{c}}) + \sum_i \kappa_i \nabla g_i^*(\bar{\mathbf{c}}) + \sum_j \lambda_j \nabla h_j^*(\bar{\mathbf{c}}) = 0 \\
&\forall_i \;\; g_i^*(\bar{\mathbf{c}}) \leq 0, \quad \forall_j \;\; h_i^*(\bar{\mathbf{c}}) = 0, \quad \forall_i \;\; \kappa_i g_i^*(\bar{\mathbf{c}}) = 0
\end{aligned}$$

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,

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

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

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

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

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

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	$\mathbf{0.8612 \pm 0.0019}$ $*$	$\mathbf{0.8583 \pm 0.0020}$ $*$	$\mathbf{0.8686 \pm 0.0020}$ $*$

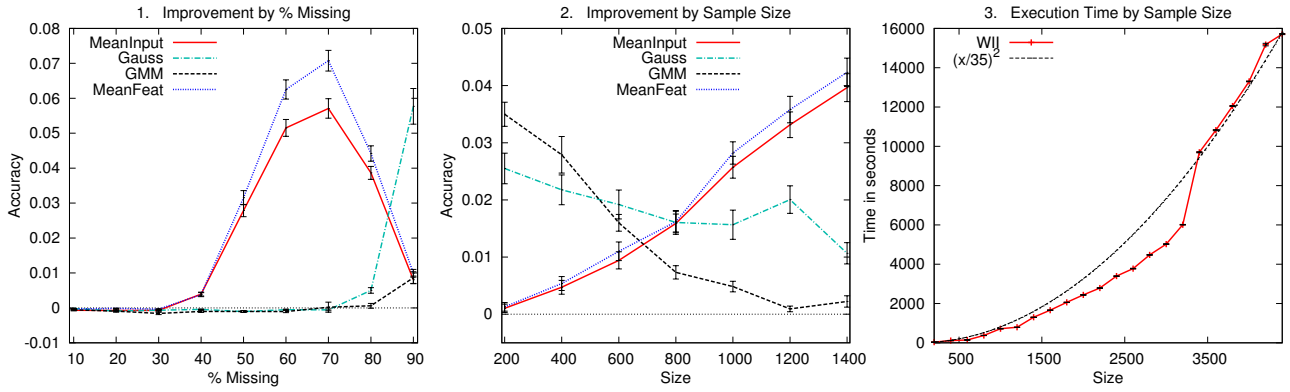

Figure 1. Detailed results on USPS classification task.

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 $\hat{\boldsymbol{\omega}}$ 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 $\hat{\boldsymbol{\omega}}$ 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 $\hat{\boldsymbol{\omega}}$. On the other hand, we see that it works for all training set sizes, again depending on $\hat{\boldsymbol{\omega}}$. 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.*

Discriminative Clustering for Market Segmentation

Peter Haider*
Dep. of Computer Science
Univ. of Potsdam, Germany
haider@cs.uni-potsdam.de

Luca Chiarandini†
Web Research Group
Universitat Pompeu Fabra
Barcelona, Spain
chiarluc@yahoo-inc.com

Ulf Brefeld
University of Bonn
Bonn, Germany
brefeld@uni-bonn.de

ABSTRACT

We study discriminative clustering for market segmentation tasks. The underlying problem setting resembles discriminative clustering, however, existing approaches focus on the prediction of univariate cluster labels. By contrast, market segments encode complex (future) behavior of the individuals which cannot be represented by a single variable. In this paper, we generalize discriminative clustering to structured and complex output variables that can be represented as graphical models. We devise two novel methods to jointly learn the classifier and the clustering using alternating optimization and collapsed inference, respectively. The two approaches jointly learn a discriminative segmentation of the input space and a generative output prediction model for each segment. We evaluate our methods on segmenting user navigation sequences from Yahoo! News. The proposed collapsed algorithm is observed to outperform baseline approaches such as mixture of experts. We showcase exemplary projections of the resulting segments to display the interpretability of the solutions.

Categories and Subject Descriptors

I.5.3 [**Clustering**]: Algorithms

General Terms

Algorithms, Experimentation

Keywords

Discriminative Clustering, Market Segmentation

1. INTRODUCTION

Market segmentation reveals divisions in a given market, where a market refers to a population of interest such as people, customers, or organizations. A market segment is a subset of a market that is characterized by similar demands and/or needs based on qualities of a given product such as price or function.

Every segment is required to meet the following criteria. (i) It is homogeneous within the segment, so that individuals within the same segment exhibit common needs and can be targeted jointly with the same marketing strategy. (ii) It is easily distinguishable from other segments to guarantee that different segments have different demands and (iii) serves as a blueprint for distinct targeting strategies. The requirements are often summarized as *homogeneity*, *identifiability*, and *interpretability* (19).

Besides frequently deployed self-organizing maps (SOMs) (13; 7), market segmentation matches the problem setting of model-based clustering approaches. Clustering techniques either minimize the within-cluster similarity (1; 14), maximize the between-cluster similarity (7), or optimize a combination of both (18), and thus aim to produce homogeneous and identifiable solutions. Once segments have been computed, new customers need to be assigned to one of the subgroups to advertise the product accordingly.

Unfortunately, mapping new instances to an existing clustering is often difficult in practice. Intuitively, the new customer should be grouped to the closest segment with respect to some similarity measure. Closeness can for instance be computed as the distance in feature space to the median or the nearest member of a segment (11; 4). However, often at the time of classifying a new instance, not all features are known. Even using the similarity measure that is used by the clustering method itself on all features is frequently observed to perform surprisingly inaccurate, see e.g., (20) and Section 4. Other difficulties are for instance distribution-based clusterings, such as Expectation Maximization (6), which assign probabilities for cluster-memberships. By doing so, customers are probabilistically related to *every* segment. Converting this soft assignment into a hard assignment by taking a *maximum a posteriori* or *winner-takes-all* decision is often suboptimal if the memberships deviate from a point distribution, in which case more than one segments are likely candidates (14).

Optimally, the segmentation is therefore learned together with a classifier that discriminatively maps new instances to clusters; a problem setting which is also known as discriminative clustering (5; 20; 24; 8). The idea is to have the clustering provide the labels for the classifier which is trained in a supervised manner. The joint optimization alters the clustering so that the segments can be easily dis-

*This work was performed during an internship at Yahoo! Research, Barcelona, Spain.

†Also Yahoo! Research, Barcelona, Spain.

KDD'12, August 12–16, 2012, Beijing, China.

criminated from each other by the classifier. Combining the two criteria thus guarantees concise clusterings and accurate classifiers. Existing approaches focus on clustering a population and predicting a cluster label for a new instance. By contrast, market segmentation is more complex. In market segmentation tasks, we need to differentiate between the data that characterizes individuals and the data that characterizes their future behavior. The clustering clearly needs to take all available information into account to generate meaningful segments. However, the classifier does not have access to future events and needs to take a decision on the available information such as gender, income, etc. This observation renders existing approaches to discriminative clustering too restrictive for market segmentation tasks.

In this paper we generalize discriminative clustering for market segmentation tasks using the structured prediction framework. We differentiate between *attributes* of a customer and her *interests/behavior*. *Attributes* are *a priori* available features of individuals of the population such as gender or income. Her *behavior* is a collection of interacting variables describing a segment. As segments need to be interpretable, we model the output data as a complex and structured variable which can be represented as a graphical model. The distinction allows for learning a classifier only on the attributes, computing the clustering on both attributes and behavior, and finally summarizing the segments only in terms of the behavior.

We devise two solutions which are based on the regularized empirical risk minimization framework. The first is a straightforward adaptation of mixtures of experts. Classifier and clustering are optimized using an alternating strategy where we fix one component while optimizing the other. The second solution uses approximations and integrates out parameters of the classifier using collapsed inference for efficiency. Both approaches use generative models for the output structure and, in contrast to conventional discriminative clustering approaches, do not involve trade-off parameters for classification accuracy and cluster consistency (class balance) because the optimization problems are not prone to trivial and degenerate solutions.

Use cases of our methods contain traditional market segmentation tasks. Consider for instance a company that aims at promoting a new product or a hotel chain that intends to lure visitors with special offers. Our methods not only compute a meaningful segmentation of the customers but also allow for devising appropriate targeting strategies from the graphical models. Moreover, our method serves as discriminative clustering for structured variables, where the task is not to output a single class/cluster label but the average structure for every segment. The differentiation between attributes and behavior increases the range of applications that can be addressed. A special – but still novel – case is obtained when attributes and behavior partially overlap.

Empirically, we study our methods on another interesting use case: Segmenting user navigation sessions on the Web for displaying segment-specific website layouts. We experiment on a large click log from Yahoo! News. The attribute data is assembled from meta-information about the session such as the timestamp, the referrer domain, and the first page request. The behavior consists of subsequent navigation actions given by click sequences. The generative representation of the behavior data is interpretable and can be easily transformed into segment-specific layouts.

The remainder of the paper is structured as follows. Section 2 discusses the relationship of our problem setting with previously studied settings and methods. In Section 3 we derive two algorithms to optimize the empirical counterpart of the expected segmented log-likelihood. Section 4 reports on empirical results using a large click log from a commercial news provider and Section 5 concludes.

2. RELATED WORK

Market segmentation tasks are often solved using neural networks such as self-organizing maps (13; 7). Kiang et al. (13) for instance extend self-organizing maps to group customers according to their attitude towards different communications modes. D'Urso and de Giovanni (7) use the natural clustering property of self-organizing maps together with dissimilarity measures which capture temporal structure of the data. In general, clustering data with self-organizing maps and variants thereof inherently implements the homogeneity assumption of market segmentation. However, classifying new instances into the clustering is often difficult and it is not possible to output generative models to summarize the resulting clusters. Additionally, the optimization criterion of self-organizing maps is highly sensitive to the actual initialization and usually converges to different local optima.

Related to market segmentation is the task of estimating a mixture model for observations (6). Introducing selector variables encoding probabilistic cluster-memberships, maximizing the log-likelihood by marginalizing over the selector is usually straightforward. The selector can be modeled in a data-dependent or data-independent fashion but the probabilistic nature of the cluster-memberships render a direct application for market segmentation tasks impossible.

Discriminative clustering simultaneously computes a segmentation of the data at hand *and* a classifier that discriminates the resulting clusters well. Existing approaches include projections into lower-dimensional subspaces (5), joint optimization of max-margin classifiers and clusterings (20; 24), the optimization of scatter metrics (21), and the maximization of an information theoretic criterion to balance class separation and classifier complexity (8). Sinkkonen et al. (17) aim to find clusters that are homogeneous in auxiliary data given by additional discrete variables. The above mentioned approaches do not predict any output variable but focus on the discrete cluster variable. Moreover, in our generalized problem setting, instances are represented as input-output pairs. The classifier discriminates the clusters given only the input, whereas the cluster parameters need to accurately estimate the outputs of the contained instances. Previous work on discriminative clustering does not split instances into two parts. They represent instances as a single input which consequently allows the classifiers to access the whole example at decision time. The same assumption is commonly being made in market segmentation studies that involve model-based clustering approaches, (9; 16; 22) but prohibits a natural solution for market segmentation tasks.

This problem setting can be seen as an alteration of the setting which the Mixture of Experts approach (10; 12) aims to solve, where the behavior y is predicted given the attributes x as a mixture model where the mixture component weights depend again on x. In our case, mixture component weights have to be always point distributions as demanded by the application. Framing the distribution of y given the mixture component as a pure generative model allows us to

derive a more efficient algorithm than that of the Mixture of Experts approach.

Zhao et al. (23) proposed a maximum-margin clustering for multivariate loss functions. Minimizing the complex losses allows for capturing structural differences that cannot be expressed in terms of standard misclassification rates. In principle, by defining a loss function that captures the differences of two clusterings, one could possibly solve market segmentation tasks as their approach implicitly favors clusterings that are easily discriminable. However, the loss function cannot be expressed in terms of the contingency table of the two clusterings, and the decoding problem in the inner loop of the algorithm, that is finding the most violated constraint, becomes intractable in practice.

Also related to our problem setting are multi-view clustering approaches, where the data is split into two disjoint feature sets, which are sometimes also called views. Bickel and Scheffer (2) present an intertwined Expectation Maximization algorithm to compute the maximum likelihood solution for one view using the expectations provided by its peer view. The two data views are modeled generatively and the algorithm maximizes the joint marginal likelihood. By contrast, we aim to find a discriminative classifier on the input view and instead of maximizing the marginal likelihood of the output view we seek to maximize the likelihood conditioned on a hard cluster assignment.

3. DISCRIMINATIVE SEGMENTATION

We now present our main contribution, the generalization of discriminative clustering for structured output variables to solve market segmentation problems. We introduce the problem setting in the next section and present a straightforward solution in terms of mixtures of experts in Section 3.2. An efficient approximation is devised in Section 3.3 and Section 3.3.3 discusses scalability issues.

3.1 Preliminaries

We are given a sample $\mathcal{M}$ from a market where individuals are represented by tuples $(x, y) \in \mathcal{X} \times \mathcal{Y}$ encoding attributes x and behavior y. Attributes x may encompass individual features like gender, income, etc while the expressed historic behavior is captured by $y \in \mathcal{Y}$ and represented as a graphical model. The behaviors y are governed by a family of distributions denoted by $P(y|\theta)$ with natural parameter θ.

In our running example on segmenting user navigation on the Web, attributes x encode meta-information about the session such as the timestamp, the referrer domain, and the first page request and is represented as a feature vector. The behavior y encodes sequences of the subsequent Web navigation and can for instance be represented as a Markov-chain where nodes correspond to pageviews and connecting edges visualize clicks.

We aim to find an appropriate segmentation of the market $\mathcal{M}$. Formally, the goal is to find a classifier $h : \mathcal{X} \to \{1, \ldots, k\}$ that maps attributes x to one of k clusters, parameterized by $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$, where the θ_j are chosen to maximize the likelihood of the observed behaviors y of the respective segment. The number of clusters k is assumed to be given by the application at hand, because it constitutes a trade-off between predictive power and effort spent for developing multiple market strategies. As h and $\boldsymbol{\theta}$ are not independent they need to be optimized jointly. Hence, over all classifiers h and parameter collections $\boldsymbol{\theta}$, we aim at maximizing the expected risk functional R that is defined in terms of the segmented log-likelihood

$$R(h, \boldsymbol{\theta}) = \int \log P(y|\theta_{h(x)}) dP(x, y). \tag{1}$$

Since the true joint distribution $P(x, y)$ is unknown, we replace Equation (1) by its empirical counterpart on the finite market sample of size n given by $\mathcal{M} = \{(x_i, y_i)\}_{i=1}^n$

$$\hat{R}(\boldsymbol{\theta}, h) = \sum_{i=1}^{n} \log P(y_i|\theta_{h(x_i)}). \tag{2}$$

Directly maximizing Equation (2) in terms of the component parameters $\boldsymbol{\theta}$ and the classifier h is infeasible, since the objective function is not only highly non-convex and non-continuous but an NP-hard problem because of combinatorial assignments. However, if the classifier h was fixed, the θ_j could be optimized directly, as h provides the segmentation and for each segment j optimal parameters $\hat{\theta}_j$ are trivially computed by

$$\hat{\theta}_j = \operatorname*{argmax}_{\theta} \sum_{i:h(x_i)=j} \log P(y_i|\theta). \tag{3}$$

For many common distribution families, the maximum likelihood estimates $P(y|\theta)$ can be computed easily by counting or averaging over the observations in the segment, respectively. Vice versa, keeping the segment parameters $\theta_1, \ldots, \theta_k$ fixed, learning the classifier h results in a standard multi-class classification scenario. Using linear models, h can be written as

$$h(x) = \operatorname*{argmax}_{j \in \{1,\ldots,k\}} w_j^\top x, \tag{4}$$

where each segment has its own weight vector w_j. In the remainder, we will use h and $\boldsymbol{w} = (w_1, \ldots, w_k)^\top$ interchangeably. The next section exploits this observation and presents a joint alternating optimization scheme.

3.2 An Alternating Optimization Scheme

A straightforward approach to solve market segmentation problems is to alternate the optimization of the classifier and the clustering while fixing the other, respectively. As shown in Equation (3), keeping the classifier fixed allows to apply standard maximum likelihood techniques to compute the natural parameters of the segments. We thus focus on deriving the classifier h for a fixed clustering. We make use of the maximum-margin framework and deploy a re-scaled variant of the hinge loss to deal with uncertain cluster-memberships (or class labels).

The idea is as follows. Intuitively, an individual (x, y) should be assigned to the segment that realizes the highest log-likelihood with respect to y. However, two or more segments might be competing for the instance and realize

Algorithm 1 Alternating Optimization

1: Input: $(x_1, y_1), \ldots, (x_n, y_n), \lambda > 0, k > 1$
2: Initialize $\boldsymbol{\theta}$ randomly
3: **repeat**
4: E-step: $\boldsymbol{w} \leftarrow \operatorname{argmin}_{\boldsymbol{w}', \boldsymbol{\xi} \geq \mathbf{0}} \ \frac{\lambda}{2}\|\boldsymbol{w}'\|^2 + \frac{1}{n}\sum_{i=1}^n \xi_i$
5: s.t. $(w'_{j_i^*} - w'_j)^\top x \geq 1 - \frac{\xi_i}{s(y)}$
6: M-step: $\boldsymbol{\theta} \leftarrow \operatorname{argmax}_{\boldsymbol{\theta}'} \sum_j \sum_{i:h(x_i)=j} \log P(y_i|\theta')$
7: **until** convergence

similar log-likelihoods, in which case a winner-takes-all decision is prohibitive. We thus treat the difference of the log-likelihoods between the most likely segment j^* and cluster $j' \neq j^*$ as a *misclassification score*, given by

$$s(y) = \log P(y|\theta_{j^*}) - \log P(y|\theta_{j'}). \tag{5}$$

These scores can be incorporated in a support vector machine by re-scaling the hinge loss and act like example-dependent costs (3). The re-scaled hinge loss becomes a convex upper bound of the difference of the log-likelihoods,

$$\ell(x) = s(y) \max\left(0, 1 - (w_{j^*} - w_{j'})^\top x\right). \tag{6}$$

Stacking up $\boldsymbol{w} = (w_1, \ldots, w_k)^\top$ and $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)^\top$, we arrive at the following maximum-margin optimization problem

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{\xi}\geq\mathbf{0}} \quad & \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & (w_{j_i^*} - w_j)^\top x \geq 1 - \frac{\xi_i}{s(y)}, \end{aligned}$$

for $1 \leq i \leq n$, $j \in \{1, \ldots, k\}$, $j \neq j^*$, $j_i^* = \arg\max_j P(y_i|\theta_j)$, and regularizaion parameter $\lambda > 0$.

Hence, Equation (2) can be optimized by randomly initializing $\boldsymbol{\theta}$ and subsequently alternating between optimizing $\boldsymbol{w}$ and θ_j while fixing the respective peer. Algorithm 1 instantiates the pseudo-code as a member of the Expectation Maximization framework. The alternating optimization scheme can also be interpreted as an adaptation of the EM algorithm for a mixture of experts (12). The classifier h, given as the weight vectors w_j, corresponds to the gating networks. In contrast to the conventional mixture of experts model, it is trained using an SVM to output deterministic decisions, instead of soft decisions. The generative distribution over behaviors differs from the expert networks only in the fact that in our problem setting the prediction of the behavior y is not allowed to depend directly on the attributes x, only via the cluster assignment.

A major drawback of the alternating approach is however the discrete assignment of individuals to only a single segment, even during the intermediate optimization steps. As a consequence, the algorithm is prone to degenerate solutions and poor local optima. Additionally, the optimization is expensive in terms of computational time as it requires the computation of a multi-class SVM until convergence in every step.

3.3 Collapsed Optimization

We now present an efficient approximation of the discriminative segmentation problem by using a continuous relaxation to the original problem formulation. We first show that the parameters of the classifier can be computed in closed-form so that the joint optimization problem depends only on the segment parameters. Second, we devise an EM-like algorithm (6) to optimize the remaining objective.

3.3.1 Eliminating the Classifier Parameters

As the discrete cluster assignments cause many difficulties, we now replacing them by soft assignments using an adjustable soft-max function. The parameter ρ controls the degree of reproducing the maximum, that is for $\rho \to \infty$ we precisely obtain the maximum operator. Incorporating the soft-max in Equation (2) yields the optimization problem

$$\max_{\boldsymbol{\theta},\boldsymbol{w}} \sum_{i=1}^{n} \log \sum_{j=1}^{k} P(y_i|\theta_j) \frac{\exp(\rho\, w_j^\top x_i)}{\sum_{j'} \exp(\rho\, w_{j'}^\top x_i)}, \tag{7}$$

which still contains the mutually dependent variables $\boldsymbol{\theta}$ and $\boldsymbol{w}$. To obtain an efficiently solvable optimization problem, we express the objective as a continuous function of $\boldsymbol{w}$ so that $\boldsymbol{w}$ can be eliminated using collapsed inference. Instead of the hinge loss in Equation (6), we employ another tight convex upper bound in terms of the squared loss,

$$\ell(x) = (\log P(y|\theta_j) - w_j^\top x)^2.$$

Implicitly, introducing the squared loss converts the classifier into a regressor that aims at predicting the log-likelihood for an individual (x, y) and the j-th segment as accurate as possible. Assuming the log-likelihoods were predicted perfectly, the parameters $\boldsymbol{w}$ would not only be optimal for the regression but also for Equation (2) as the classifier h in Equation (4) would still return the most likely segment. Changing the loss function also has the advantage that now the optimal solution for $\boldsymbol{w}$ can be computed analytically. The corresponding regularized optimization problem is also known as regularized least squares regression (RLSR) or ridge regression and is given by

$$\min_{\boldsymbol{w}} \ \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{n}\sum_{j=1}^{k}\left(\log P(y_i|\theta_j) - w_j^\top x_i\right)^2, \tag{8}$$

for $\lambda > 0$. Setting the derivative with respect to $\boldsymbol{w}$ to zero and solving for w_j, we obtain the optimal solution that can be computed in closed-form

$$w_j = X^\top (XX^\top + \lambda I)^{-1} \pi(\theta_j), \tag{9}$$

where $X \in \mathbb{R}^{n\times d}$ contains the stacked attribute vectors and $\pi(\theta_j) = (\log P(y_1|\theta_j), \ldots, \log P(y_n|\theta_j))^\top$ is the vector of log-likelihoods for the j-th segment. The computation of the inner product

$$w_j^\top x = \pi(\theta_j)^\top (XX^\top + \lambda I)^{-1} Xx$$

can effectively be sped-up by precomputing the linear transformation of the attributes. Introducing auxiliary variables $\bar{x}$ given by

$$\bar{x} = (XX^\top + \lambda I)^{-1} Xx,$$

allows to efficiently rewrite the inner product as $w_j^\top x = \pi(\theta_j)^\top \bar{x}$. Further note that $\bar{x}$ depends only on inner products of individual attributes. Hence, the kernel trick can be applied to incorporate non-linearity. Introducing a Mercer kernel κ, the auxiliary matrix $\bar{X}$ can be written in terms of the corresponding kernel matrix K with elements $K_{ij} = \{\kappa(x_i, x_j)\}_{ij}$ as $(K + \lambda I)^{-1} K$. The classifier can then be expressed in terms of a set of dual multipliers α and the kernel function as $h(x) = \arg\max_j \sum_i \alpha_{ji} \kappa(x_i, x_j)$. The dual multipliers can be obtained explicitly as a function of the component parameters as

$$\alpha_j(\theta_j) = \pi(\theta_j)^\top (K + \lambda I)^{-1}. \tag{10}$$

Substituting the obtained observations in Equation (7) results in a simplified optimization problem that does no

longer depend on $\boldsymbol{w}$ and that has only the θ_j as free parameters,

$$\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log \sum_{j=1}^{k} P(y_i|\theta_j) \frac{\exp(\rho\, \pi(\theta_j)^\top \bar{x}_i)}{\sum_{j'} \exp(\rho\, \pi(\theta_j)^\top \bar{x}_i)}. \quad (11)$$

3.3.2 Optimizing the segment parameters

The optimization problem in Equation (11) can be solved with an EM-like algorithm (6) using auxiliary variables $z_{i,j} \geq 0$, with $\sum_j z_{i,j} = 1$, encoding the belief that the i-th example belongs to the j-th cluster. EM-like algorithms consist of two phases which assign instances to generating segments (E-step) and maximize the segment parameters with respect to the associated instances (M-step), respectively. In the E-step, it therefore suffices to identify the auxiliary variables with the true posterior probabilities given the current $\boldsymbol{\theta}$,

$$z_{i,j} \propto P(y_i|\theta_j) \frac{\exp(\rho\, \pi(\theta_j)^\top \bar{x}_i)}{\sum_{j'} \exp(\rho\, \pi(\theta_j)^\top \bar{x}_i)}.$$

Following the general EM framework, we express the empirical risk functional in Equation (11) in terms of the expectations $z_{i,j}$. This allows us to effectively pull the logarithm into the sum over segments for the M-step; we arrive at the optimization problem

$$\max_{\boldsymbol{\theta}} \sum_i \sum_j z_{i,j} \log \left[P(y_i|\theta_j) \frac{\exp(\rho\, \pi(\theta_j)^\top \bar{x}_i)}{\sum_{j'} \exp(\rho\, \pi(\theta_j)^\top \bar{x}_i)} \right],$$

which can be rewritten as

$$\max_{\boldsymbol{\theta}} \sum_{i,j} z_{i,j} \Big[\rho\pi(\theta_j)^\top \bar{x}_i - \log \sum_{j'} \exp(\rho\pi(\theta_{j'})^\top \bar{x}_i) + \log P(y_i|\theta_j) \Big].$$

The above Equation is to be maximized with respect to $\boldsymbol{\theta}$. Compared to conventional M-steps for mixture model estimation problems, $\boldsymbol{\theta}$ appears not only in $P(y_i|\theta_j)$, but also in what are usually the segment weights for each example. This renders the objective function non-concave and, consequentially, there is no exact analytical solution.

As a remedy, we derive an approximation that is linear and non-decreasing in the $\pi(\theta_j)$, rendering the objective concave in θ_j and thus analytically solvable for common choices of $P(y|\theta_j)$. We begin with approximating the normalization term $Z(\boldsymbol{\theta}) = \log \sum_{j'} \exp(\rho\pi(\theta_{j'})^\top \bar{x}_i)$ by its first-order Taylor expansion around the current $\boldsymbol{\theta}^{old}$ which is given by

$$Z(\boldsymbol{\theta}) \approx \sum_{j'} t_{ij'} \rho\pi(\theta_{j'})^\top \bar{x}_i + C, \quad (12)$$

where the $t_{ij'} = \frac{\exp(\rho\pi(\theta_{j'}^{old})^\top \bar{x}_i)}{\sum_j \exp(\rho\pi(\theta_j^{old})^\top \bar{x}_i)}$ are the Taylor coefficients and C is a constant. Substituting Equation (12) into the objective function of the M-step and collecting the coefficients gives us

$$\operatorname*{argmax}_{\boldsymbol{\theta}} \sum_i \sum_j \log P(y_i|\theta_j) \left[z_{i,j} + \rho \sum_{i'} \bar{x}_{i'i} \Big(z_{i',j} - \sum_{j'} z_{i',j'} t_{i'j'} \Big) \right], \quad (13)$$

Algorithm 2 Collapsed Optimization Algorithm

```
1: Input: (x_1, y_1), ..., (x_n, y_n), λ
2: ρ ← 1, t ← 1, initialize θ^(0) randomly
3: repeat
4:    E-step: Q(z_i = j) ← P(z_i = j | x_i, y_i, ρ, λ, θ^(t−1))
5:    M-step: θ^(t) ← arg max_θ Σ_i Σ_j Q(z_i = j) ×
6:                                  log P(y_i, z_i = j | x_i, ρ, λ, θ)
7:    ρ ← ρ × 1.1, t ← t + 1
8: until convergence
9: θ̂ = argmax_θ R̂(θ, α(θ))
10: α ← α(θ̂)
```

which is a linear function of $\log P(y_i|\theta_j)$. For increasing ρ, the Taylor coefficients for each instance approach a point distribution exponentially fast, i.e. $t_{i\cdot} \to (0, \ldots, 0, 1, 0, \ldots, 0)^\top$. The same holds for the auxiliary variables $z_{i,j}$, which approach t_{ij}. Thus, the second summands of the coefficients in Equation (13) approach zero, and hence for large ρ all the $\log P(y_i|\theta_j)$ have either positive or very small negative coefficients. We clip coefficients below zero to guarantee that the objective function is non-decreasing in the $\log P(y_i|\theta_j)$ and obtain an approximation that approaches the exact solution with increasing ρ.

The softmax factor ρ therefore constitutes a trade-off between accurately approximating the original optimization problem of maximizing $\hat{R}(\boldsymbol{\theta}, h)$ and smoothing the objective function to facilitate finding good local optima.

We deal with this tradeoff by starting the EM-algorithm with $\rho = 1$, and multiplying it by a constant factor each iteration. Preliminary experiments have shown the factor 1.1 to work well. Due to the approximation of the hard decisions with a soft-max, the algorithm is not guaranteed to monotonically increase the true objective value. We thus select the intermediate result with the highest objective value as the final solution for the cluster parameters.

Algorithm 2 shows the collapsed optimization algorithm in pseudo code. In line 2, the cluster parameters $\boldsymbol{\theta}$ are initialized randomly. Line 4 performs the expectation step of the EM-algorithm, computing the current posterior estimates, given the soft-max factor ρ, the cluster parameters, and implicitly also the classifier h. The maximization step in line 5 boils down to an optimization problem of the form $\sum_i \sum_j c_{ij} \log P(y_i|\theta_j)$, which for non-negative coefficients c_{ij} can be solved analytically for many choices of distribution families. For example if $P(y|\theta)$ is a multinomial distribution over $y \in \{1, \ldots, m\}$ with $P(y = q|\theta_j) = \theta_{j,q}$, the maximum is attained for $\theta_{j,q} = \sum_i c_{ij} [\![y_i = q]\!] / \sum_i c_{ij}$ for all j, q, where $[\![\cdot]\!]$ is the indicator function. Finally, lines 9 and 10 select the best intermediate solution in terms of the true objective, and re-obtain the explicit classifier using Equation (10).

3.3.3 Scalability

The computational bottlenecks of the collapsed optimization algorithm are the computation of the $\bar{x}_i$, involving matrix inversions and multiplications, and the computation of the coefficients in Equation (13), where we have to sum over all pairs of examples, leading to an overall complexity of the algorithm of $O(n^{2.376} + n^2 kT)$.

For applications with a large number of examples the super-quadratic dependence on the sample size n makes the

algorithm effectively intractable. We can alleviate this by randomly partitioning the examples in the least-squares estimation in Equation (8) into s disjoint subsets $S^{(1)}, \ldots, S^{(s)}$ of size m. For each subset the weight vectors $\boldsymbol{w}^{(l)}$ are estimated separately, and thus within each subset the vectors $\pi(\theta_j)$ and the transformed examples $\bar{x}$ have only m components. Consequently, in Equation (13) the inner summation over the examples only runs over the m examples in the subset to which example (x_i, y_i) belongs. Finally, we obtain the parameters of the classifier h by averaging the weight vectors over all subsets, $w_j = \frac{1}{s}\sum_l w_j^{(l)}$.

Mixing the separately learned weight vectors is identical to the mixture weight method by Mann et al. (15) that has been shown, theoretically and empirically, to yield results close to learning a weight vector using all examples at once. Note however that $\boldsymbol{\theta}$ is still learned from the whole, unpartitioned training sample. Using the partitioned estimation for the weight vectors, the overall complexity of the algorithm becomes $O(nm^{1.376} + nmkT)$. For $m \ll n$, the computation becomes tractable even for very large sample sizes n.

4. EMPIRICAL EVALUATION

In this section, we evaluate the proposed algorithm using a large data sample from Yahoo! News United Kingdom[1]. The click log contains browsing session logs, where events from the same user are identified by their browser cookies and sessions are split after 25 minutes of inactivity. We use all sessions from June 2011 for training and the first week of July 2011 for testing. Figure 1 shows the categories, such as *politics/world*, *politics/uk*, *science/space*, in the training set, averaged over the four weeks where different colors correspond to different categories[2].

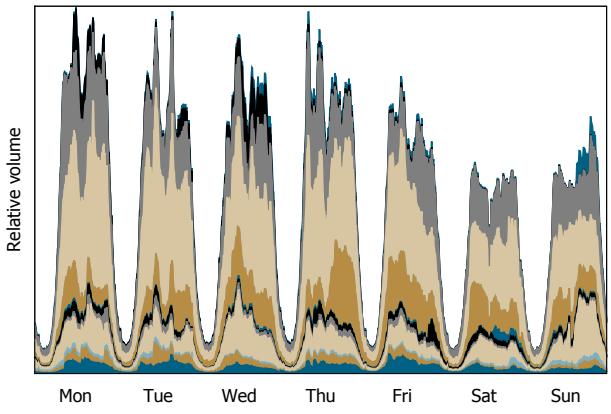

Figure 1: Click volume of categories over time.

The goal of our study is threefold: Firstly, we aim to segment the user sessions of Yahoo! News according to their interests expressed by clicks. Secondly, new sessions need to be classified to one of the segments so that every segment accurately predicts subsequent clicks of the respective user. Finally, the segments need to be interpretable to allow for devising target strategies from the segment description. A typical targeting strategy in our Yahoo! News example could for instance be a dynamic layout of the Web site to advertise news articles of categories that the respective user is probably interested in.

From a data perspective, modeling sequences of clicked categories by Markov processes is straightforward. However, Markov processes, e.g., visualized by transition matrices, are difficult to interpret as the entries encode interests with respect to the previous category. Taking the inferred Markov model properly into account would imply changing the website layout within a session depending on the previous category. A simpler way to obtain interpretable clusters is to use multinomial distributions for the output variables of interest. We use the sequences of user clicks enriched with the respective locations of the clicks. That is, the behavior y consists of the multi-set of subsequently clicked categories c and link sections s. The distribution $P(y|\theta_j)$ is defined as the product of multinomial distributions

$$P(y|\theta) = \prod_l P(c_i|\mu) \prod_j P(s_j|\nu),$$

where μ and ν are the parameter vectors governing the distributions over categories and link sections, respectively.

The attributes x of a session is represented as a binary feature vector encoding the most common referrer domains, the respective category of the first pageview, as well as features encoding the timestamp; we use binary indicators for every day of the week, for each hour of the day, and for each hour of the week. For the collapsed algorithm, we use a linear kernel and randomly partition the training data into disjoint subsets of size 1,000 for computing the predicted log-likelihoods.

4.1 Baselines

We compare the collapsed algorithm with three baselines, the alternating optimization scheme in Section 3.2, a mixture of experts model and a k-means based solution. The *mixture of experts model* (12) minimizes the squared error in terms of the within-cluster log-likelihoods and optimizes the marginal likelihood

$$\sum_i \log \sum_j P(y_i|\theta_j) P(z_i = j|x).$$

Instead of the prior $P(z = j)$ we have a conditional distribution $P(z = j|x)$ which is defined in analogy to the collapsed algorithm as

$$P(z = j|x) \propto \exp(\sum_i \alpha_{j,i} k(x_i, x)).$$

The mixture of experts model is optimized with a standard EM-algorithm and therefore provides only probabilistic cluster assignments and does not take into account that sessions need to be assigned to only a single cluster.

The third baseline is derived from the straightforward, yet somewhat naïve, approach to segment the input space first and only then optimize the generative model in each cluster. The drawback of this non-iterative approach is that it does generally not lead to homogeneous behavior within clusters because the segments are fixed when estimating the generative models. We use k-means for finding the clustering, and estimate segment paramters $\boldsymbol{\theta}$ by maximum likelihood based on the hard cluster assignments. The classifier h classifies a new instance into the cluster with the nearest centroid.

[1]All processing is anonymous and aggregated

[2]Colors occur more than once due to the large number of categories.

In each setting, every algorithm is deployed 10 times with random parameter initializations and in the remainder we only report the results of the run with highest training likelihood.

4.2 Convergence

In this section, we evaluate the convergence behavior of the collapsed algorithm. Recall that the collapsed algorithm optimizes an approximate objective, where the hard cluster assignments are replaced by a soft-max controlled by an increasing factor ρ. To cancel out effects caused by the approximation, we substitute the resulting $\boldsymbol{\theta}$ into the exact optimization criterion in Equation (2) and measure the respective objective value. Note that the results do not necessarily increase monotonically.

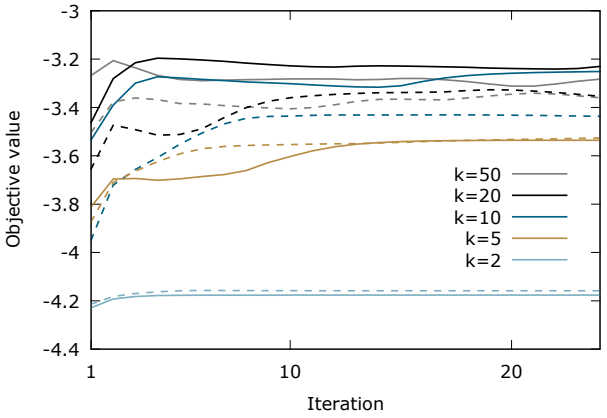

Figure 2: Objective values for the collapsed algorithm (solid) and the mixture of experts baseline (dashed), for different numbers of clusters k.

Figure 2 shows the results for different numbers of clusters for the collapsed algorithm (solid curves). For comparison, we also added the mixture of experts baseline (dashed curves). As expected, the true objective value is not monotonic, since both algorithms optimize an approximation to the exact optimization criterion. The figure also shows that the best values are obtained after at most 20 iterations.

4.3 Predictive Performance

To evaluate the performance of the collapsed algorithm, we measure its predictive accuracy in terms of how well future behavior can be predicted. The classifier and the segmentation are learned jointly as described in Section 3 using the training set and then deployed to the test set. The sessions in the test set are first classified by the classifier in one of the segments which is then used to predict the future clicks of the user. Since the final prediction is a complex variable, we refrain from expressing the performance in terms of error rates and measure the predictive log-likelihood $\log P(y|\theta_{h(x)})$ instead. We compare the collapsed algorithm to the alternating optimization scheme, the mixture of experts model, and the k-means based solution. We report on averages and standard errors over 10 repetitions with different random initializations.

Figure 3 shows the predictive performance for varying numbers of clusters. Not surprisingly, all methods perform

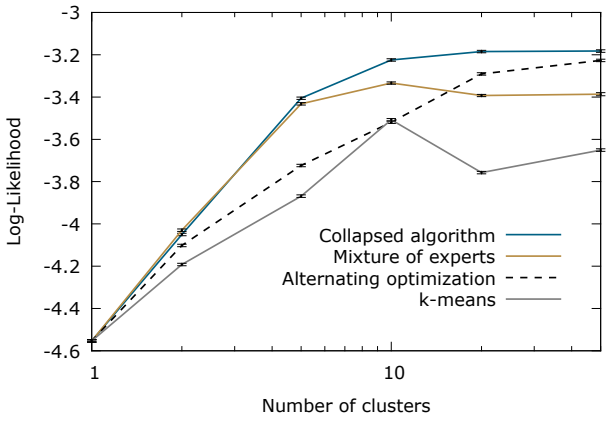

Figure 3: Averaged predictive performance and standard error.

equally worse for only a single cluster. For only a few clusters, the mixture of experts baseline performs about as well as the collapsed algorithm. We credit this finding to the existence of easy-to-reach solutions that do not necessarily require hard cluster assignments in the $\boldsymbol{\theta}$-steps. However, when the number of clusters grows, the performance of the mixture of experts approach decreases slightly while that of the collapsed model increases. Here it becomes more and more important to select the parameters in a way that allows to discriminate well between the clusters, and thus the collapsed algorithm outperforms the baselines significantly. The alternating algorithm and the k-means baseline perform significantly worse than the collapsed algorithm. Only for 20 and more clusters the alternating algorithm produces better results than the mixture of experts model. Note that the k-means performs worst as it does not use an alternating update schema but first learns the clustering and then estimates the generative models using the fixed segments.

It is apparent that the predictive performance levels off after increasing the number of clusters beyond 10. Intuitively, this observation can be explained by a trade-off between classification and segmentation: even if a more fine-grained clustering would be able to predict the future behavior more accurately, the classifier cannot discriminate well between a larger number of similar clusters to identify the best-matching segment. We observe a natural trade-off between predictive power and the effort that has to be spent for developing and maintaining target strategies for a large number of market segments.

The execution time of the collapsed algorithm for a solution with 10 clusters is within the range of 3 hours, compared to about an hour each for the mixture of experts and the k-means baselines. The alternating optimization however takes about 11 hours which renders its application infeasible in practice.

4.4 Discussion

Market segmentation aims at grouping similar individuals of a population together that share the same needs or that have similar demands. The goal is to target individuals within the same segment jointly e.g., to advertise a new product. To this end, the segments need to be interpretable

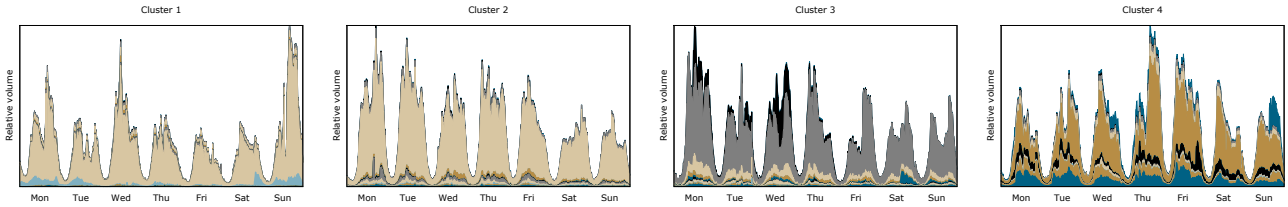

Figure 5: Click volumes of categories over time for the four clusters.

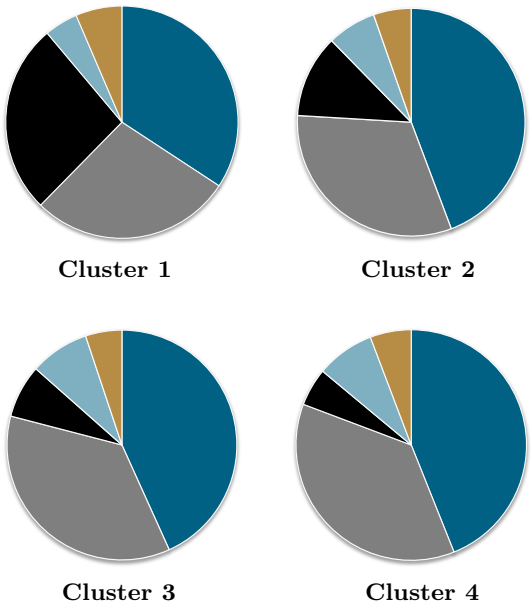

Figure 4: Visualization of click frequencies for the five most frequent link locations using four clusters.

to derive a concise description of the segments that can be converted into a segment-specific targeting strategy.

In our collapsed algorithm, generative models in each segment encode the contained behavior and interest. The flexibility of the probabilistic inference machinery allows us to project the behavior onto discriminative variables to visualize different characteristics of the clusters. In this section we give two examples for such projections to visualize differently distributed user behavior across the clustering. For simplicity, we use a solution with four clusters.

The first example shows a visualization of segment-specific user clicks in terms of their location on the Web page. Including the location of clicks is necessary for altering the layout dynamically as changes in frequently clicked areas will have impact the behavior more than substituting a redundant and less clicked widget. We focus on the five modules of the Web site that receive the highest number of clicks in the data.

Figure 4 shows the results. Segments 2, 3, and 4 exhibit very similar click behavior in terms of the clicked modules. By contrast, cluster 1 differs significantly in the usage of the Web components. On average, users in cluster 1 prefer the location visualized in black over the alternatives compared to users in the other segments. This observation could be exploited to directly devise target strategies. While members of cluster 2–4 should be addressed by changing the content of the modules visualized in gray or dark blue, users in the first segment could also be triggered by the module encoded in black.

Analogously, the behavior could be projected on the categories to visualize the respective distribution of categories for each segment. However, we choose to show a more interesting projection for lack of space. The incorporation of the timestamps of the sessions allows us to visualize the clusters in time. As the feature representation of timestamps encompasses one week, Figure 5 shows the average category distribution across the days of the week where different colors correspond to different categories.[3]

Apparently, the clusters do not only differ in terms of the categories but also specialize on certain periods in time because the segments are optimized using *all* available data, that is, attribute *and* behavior encoding variables. The first cluster clearly specializes on Sundays and is characterized by a clean topic distribution. The three other cluster also possess dominant categories but focuses more on working days than on weekends. Cluster 4 contains the most diverse set of categories and acts like a basin for categories that are not as easy to discriminate. Here it becomes obvious that a solution with only four clusters may not be optimal for the task at hand. When we increase the maximal number clusters, the category distribution of clusters becomes cleaner that is less categories are likely. Additionally, clusters adapt better to specialized periods such as working days or weekends for larger k.

Taking various such projections into account describes segments from different angles and helps to find a concise targeting strategy. For instance, knowing the articles that are likely to be read in an ongoing session helps to address the respective user in various ways including displaying ads. Incorporating context informations such as the click behavior of the segments, finally allows for tailoring web pages to each segment and to increase the overall user experience.

5. CONCLUSION

We studied discriminative clustering for structured and complex response variables that can be represented as generative models. The problem setting matches market segmentation tasks where populations are to be segmented into disjoint groups. Solving market segmentation-like problems appropriately not only involves a clustering of the individuals but also learning a classifier that discriminates well be-

[3]Colors are again reused due to the large number of categories.

tween the segments, for instance to allow for classifying new customers to one of the groups. The two components need to be learned jointly and have access to different pieces of information about the individuals: the classifier needs to group individuals on the basis of *a priori* available informations while the clustering aims at grouping people with similar (future) needs or behavior.

We devised two algorithms based on alternating optimization and collapsed inference, respectively. Empirical results showed that the collapsed variant is not only more efficient but also predicts accurately the click behavior of users for Yahoo! News. The generative nature of the clustering led to interpretable clusters. We showed how projections of the clustering on only a few variables allowed for targeting the detected segments individually and contributed to user understanding.

Our approach is not restricted to Yahoo! News and can generally be applied to arbitrary market segmentation tasks and other Web sites to improve the overall user experience. As our approach is orthogonal to personalized approaches, future work will study the integration of both frameworks.

Acknowledgements

Part of this work was supported by the German Science Foundation under the reference number GA 1615/1-1.

References

[1] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

[2] S. Bickel and T. Scheffer. Multi-view clustering. In *Proceedings of the IEEE international conference on data mining*. Citeseer, 2004.

[3] U. Brefeld, P. Geibel, and F. Wysotzki. Support vector machines with example dependent costs. In *Proceedings of the European Conference on Machine Learning*, 2003.

[4] R. D'Andrade. U-statistic hierarchical clustering. *Psychometrika*, 4:58–67, 1978.

[5] F. De la Torre and T. Kanade. Discriminative cluster analysis. In *Proceedings of the 23rd international conference on Machine learning*, pages 241–248. ACM, 2006.

[6] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[7] P. D'Urso and L. D. Giovanni. Temporal self-organizing maps for telecommunications market segmentation. *Neurocomput.*, 71:2880–2892, 2008.

[8] R. Gomes, A. Krause, and P. Perona. Discriminative clustering by regularized information maximization. In *Advances in Neural Information Processing Systems*, 2010.

[9] J. Huang, G. Tzeng, and C. Ong. Marketing segmentation using support vector clustering. *Expert systems with applications*, 32(2):313–317, 2007.

[10] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[11] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 2:241–254, 1967.

[12] M. Jordan and R. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

[13] M. Y. Kiang, M. Y. Hu, and D. M. Fisher. An extended self-organizing map network for market segmentation – a telecommunication example. *Decision Support Systems*, 42:36–47, 2006.

[14] S. P. Lloyd. Least square quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[15] G. Mann, R. McDonald, M. Mohri, N. Silberman, and D. Walker. Efficient large-scale distributed training of conditional maximum entropy models. *Advances in Neural Information Processing Systems*, 22:1231–1239, 2009.

[16] M. Namvar, M. Gholamian, and S. KhakAbi. A two phase clustering method for intelligent customer segmentation. In *2010 International Conference on Intelligent Systems, Modelling and Simulation*, pages 215–219. IEEE, 2010.

[17] J. Sinkkonen, S. Kaski, and J. Nikkilä. Discriminative clustering: Optimal contingency tables by learning metrics. *Machine Learning: ECML 2002*, pages 109–137, 2002.

[18] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the International Conference on Machine Learning*, 2001.

[19] M. Wedel and W. Kamakura. *Market segmentation: conceptual and methodological foundations*, volume 8. Springer, 2000.

[20] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. Maximum margin clustering. *Advances in neural information processing systems*, 17:1537–1544, 2005.

[21] J. Ye, Z. Zhao, and M. Wu. Discriminative k-means for clustering. In *Advances in Neural Information Processing Systems*, 2007.

[22] W. Yu and G. Qiang. Customer segmentation of port based on the multi-instance kernel k-aggregate clustering algorithm. In *Management Science and Engineering, 2007. ICMSE 2007. International Conference on*, pages 210–215, 2007.

[23] B. Zhao, J. Kwok, and C. Zhang. Maximum margin clustering with multivariate loss function. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 637–646. IEEE, 2009.

[24] B. Zhao, F. Wang, and C. Zhang. Efficient multiclass maximum margin clustering. In *Proceedings of the 25th international conference on Machine learning*, pages 1248–1255. ACM, 2008.

Chapter 6

Discussion

All presented papers advance the state of the art in prediction with mixture models in specific ways. The following chapter discusses how the thesis as a whole relates to previous literature and how the contributions of the individual papers fit into the overall picture. At first a discussion about existing general approaches to prediction with mixture models is necessary.

Mixture of experts [Jordan 94] is a generic framework for learning predictive mixture models, used e.g. successfully for EEG signal classification [Subasi 07], pedestrian classification [Enzweiler 11], and protein structure prediction [MacDonald 12]. It makes use of the fact that the models in each mixture component are efficiently tractable, while the learned model overall is complex. The mixture components are optimized with respect to the predictive accuracy of the overall model. However, this generic approach can not take into account domain-specific knowledge about the actual cluster structure in the data without additional feature engineering.

Deep neural networks [Hinton 06], which have recently been successfully applied to various applications, e.g. [Mohamed 11], [Mohamed 12], [Socher 12], and [Yu 12], can also be interpreted as mixture models. They consist of multiple layers, each with a predetermined number of binary feature detectors. The bottom-most layer encodes the input data. Within each layer, the feature detectors are trained to maximize the probablity of the states of the lower layers. Finally, the top-most layer has additional weights that are optimized to predict a target variable. Each feature detector in the top layer is a predictor for the target variable. Viewing the approach in terms of mixture models provides an insight into why deep belief networks work so well. One can regard each subset of feature detectors within a layer of the network as a mixture component c. Predictions for the activations in the layer above y given the layer below x are computed with the usual $P(y|x) = \sum_c P(c|x)P(y|c)$, using sampling or mean-field approximation [Lee 09] to approximate the sum over c. Now finding good parameters for both $P(c|x)$ and $P(y|c)$ is a challenge, because in general the optimization problem is highly non-convex, and optimization procedures can easily get stuck in poor local optima. But because the subsets of feature detectors in deep belief networks share parameters with each other, in effect one can utilize a number of components that is exponential in the number of detectors. Thus even if most components have little predictive power due to poor local optima, there is a high chance that at least some of the components are useful. Furthermore, $P(c|x)$ is defined in terms of the joint probability distribution $P(x, c)$, which can be optimized using unlabeled data, which is usually available in much greater abundance than

labeled data. Thus, unsupervised pre-training can be leveraged to obtain a good initialization for the component parameters. The recently studied improvement of deep neural networks called dropout provides an indication for the correctness of this hypothetical explanation [Hinton 12]. Here, for each stochastic gradient update step, each detector is removed with probability 0.5. This makes training different subsets of detectors more explicit, and consequently improves accuracy. Thus far deep neural networks have been used for applications like visual object recognition and speech recognition. Adapting them to other types of data like text is still an open challenge. Though they have been used for textual data with some success [Socher 11], the problem of capturing the complex sequential dependencies of natural language is still not generally solved.

A conceptually similar, but methodically different approach are support vector machines for structured outputs with latent variables [Yu 09]. Here, a set of hidden variables is introduced, where each possible realization c can be seen as a mixture component. For prediction, the decision function is maximized over the components, instead of averaged. Compared to deep belief networks, the relationship between input and hidden variables can not be pre-trained using unlabeled data, because it is a purely discriminative approach and there is no notion of a distribution over inputs. This does, however, make it easier to utilize all sorts of data types.

A large class of generative mixture models can be subsumed under the term topic models, overviewed in [Steyvers 07] and [Teh 10]. Often-used frameworks are hierarchical dirichlet processes [Teh 06] and Indian Buffet processes [Griffiths 06]. Mostly used for textual data, mixture components are topics c, which are defined as distributions over words in the vocabulary. Words y in a document x are predicted in the usual fashion as $P(y|x) = \sum_c P(c|x)P(y|x)$. For example in the application of document retrieval, the match of a document x to a query word y can be computed via $P(y|x)$ [Wei 06]. In order for topic models to work as intended, the distributions over topics given a document $P(c|x)$ are often enforced to be sparse via a Dirichlet prior [Blei 03]. Each document is thus associated with only a small number of topics, while not being as restrivtive as requiring a single topic for each document.

In most cases, learning mixture models involves either solving a non-convex optimization problem, for which the globally optimal solution can not be found efficiently, or in the fully Bayesian setting, computing an intractable integral. There are several general approaches to address this problem. One of them is the EM algorithm [Dempster 77], which alternates between maximizing a lower bound of the complete data likelihood and constructing a new lower bound at the current solution. It is guaranteed to find at least a local optimum of the optimization problem. Another class of approaches are variational methods [Attias 99], which approximate the joint data distribution with a combination of simpler distributions for which the solution can be found efficiently. Furthermore there have been developed Markov chain Monte Carlo algorithms such as the Gibbs sampler [Smith 93] which in the limit of infinitely many sampling iterations generates samples from the true posterior distribution over the model parameters. The methods in this thesis use several variants of those approaches, as well as a simple greedy procedure when it is appropriate such as in Section 2.5.

6.1 Application-oriented Modelling

In order to use the available training data as effectively as possible, the used model has to be tailored to each application specifically. The papers of this thesis follow this principle for four different applications and thereby provide new methods to deal with them more effectively.

In Chapter 2 [Haider 09], the framework of Bayesian model-based clustering is extended to handle vectors of binary features that are not independent. For textual data, the independence assumption is clearly unjustified. Hence the demonstrated increase in predictive accuracy through allowing dependencies in the input space was expected. Tractability of the model is maintained by using a novel pretrained feature transformation, which is specifically optimized in order to minimize the approximation error induced by the independence assumption in the transformed space. Thus the paper broadens the spectrum of classes of observations for which a proper mixture model can be formulated and efficiently inferred.

For observations that can be abstracted into a graph where the presence of edges constitutes evidence for shared cluster membership and the absence of edges consitutes only weak evidence against, Chapter 3 [Haider 12b] provides a novel way of modeling the posterior distribution over clusterings. It depends only on the cliques of the graph, and does not have to make any assumptions about the process generating the observations. Compared to previous approaches for graph partitioning [Ng 02], it does not provide only a point estimate, but a complete distribution. Furthermore, the number of clusters does not have to be specified in advance.

In Chapter 4 [Dick 08] the main advance is that the developed method not only estimates a single set of imputations for the missing values, but instead a distribution over imputation sets. This is more appropriate because it reflects the remaining uncertainty about the imputed values. This is similar to replacing a maximum-a-posteriori point estimate with full Bayesian inference.

The optimization problem stated in Chapter 5 [Haider 12a] is the first formulation of the market segmentation problem that neither needs to define the classification criteria or the cluster parameters beforehand nor neglects the fact that each training example has to be predicted by a single cluster. By combining the goals of separability and homogeneity into a single objective function, there is no intermediate step that relies on heuristics involved.

6.2 Improvement of Predictive Power

In full Bayesian approaches of mixture models, there is no set of parameters that gets optimized. Thus there is only one way of pursuing that the resulting mixture model (respectively distribution over mixture models) is useful for prediction: the constructor of the model has to ensure that it matches the real generative process or the dependencies between observables and latent variables as closely as possible. However if the models include hyperparameters, those can be optimized with respect to the predictive accuracy. In Chapter 2 [Haider 09] this is accomplished through the use of the precision parameter σ that governs the probability of deviations of feature frequencies compared to the training set. Indirectly this parameter controls the granularity of the resulting clusterings, which greatly influences predictions. By

tuning it on a hold-out dataset, see Section 2.6, we can maximize predictive accuracy.

The concentration hyperparameters α_a and α_s in Chapter 3 [Haider 12b] play a similar role. They control how likely it is that a new email originated from a different botnet than all previous emails from the same IP address or spam campaign, respectively. They, too, can be tuned on a hold-out set as in Section 3.4 to optimize predictive power of a model that by itself does not include an optimization criterion that takes prediction accuracy into account.

Approaches based on expected risk minimization provide the possibility to define the risk and the resulting objective funtion in terms of the prediction accuracy. In Section 4.3 [Dick 08] the risk is defined as a function of a decision function which depends on the estimated mixture, such that the mixture itself is optimized with regard to accuracy. The same is true for the optimization problem in Chapter 5 [Haider 12a]. There the relationship between the clustering and the resulting prediction is more direct than in previous work.

All four presented papers include evidence that in their respective applications the use of mixture models leads to an improvement compared to the corresponding single-component models. Single-component baselines are either explicitly mentioned or covered as special cases of one of the reference methods:

In the spam campaign detection application of Chapter 2 [Haider 09], the hyperparameter σ controls the granularity of the clustering. In the limit of $\sigma \to \infty$ the solution degrades to a single cluster, and thus to prediction with a single-component model. Since σ is tuned with respect to predictive accuracy, this case is implicitly included in the evaluation and shown to perform worse than the mixture model. Another single-component baseline is the one-class support vector machine, which is explicitly shown to perform worse, see Section 2.6.

The case with the application of botnet detection in Chapter 3 [Haider 12b] is similar. Here the hyperparameters also enforce in the limit of $\alpha_a \to 0$ and $\alpha_s \to 0$ a single-cluster solution. As with σ above, they are tuned with respect to predictive accuracy, such that the single-component model is implicitly part of the evaluation.

In the application of classifying emails with missing values in Chapter 4 [Dick 08], we compare the method of finding a mixture of imputations for the missing values to several methods for finding a single set of imputations. In each case, the single set of imputations is used as a prior for the distribution over imputations. The evaluation in Section 4.6 shows that for almost all combinations of single imputation methods and datasets learning a distribution over imputations increases predictive accuracy significantly.

In Chapter 5 [Haider 12a] about market segmentation, we evaluate the method for different numbers of clusters, including one. The results in Section 5.4 show that predictive accuracy rises steeply with increasing number of clusters in the beginning, and then levels off.

6.3 Evaluability

The third aspect of the overall research question of this thesis is how prediction with mixture models can serve as a means to evaluating the cluster structure underlying the mixture. One of the approaches that can be taken in order to evaluate the accuracy of clusterings found by a method is to use synthetic data [Thalamuthu 06]. One simply

generates artificial data that is assumed to have the same characteristics as the real data, and compares the found clustering to the clusters in the generated data. But this approach suffers from the flaw that one has to make strong assumptions about the characteristics of the real data. If those assumptions are wrong, the obtained evaluation is also incorrect.

Generative models such as topic models can be evaluated by computing the joint likelihood of the training set, see e.g. [Heller 05]. In the same fashion one can select different models or optimize their hyperparameters. This is reasonable for choosing few parameters of a single generative model family, but it can not be reliably used for choosing the family of the model: It is always possible to construct a model with a prior that assigns maximum probability mass to all training examples and zero probability to all others. Thus it compares favorably to all other models under the criterion of training set likelihood, but is actually useless. Consider for example a mixture model that assigns each example to its own cluster, with parameters that assign probability one to the example in the cluster, and a prior over cluster parameters that is only supported on the values taken by those clusters. Here, the sum of marginal likelihoods over training examples is one. Consequentially, the marginal likelihood of an example not included in the training set is zero. It is more accurate to evaluate a mixture model by computing (approximately) the probability of held-out documents [Wallach 09]. This is a special case of assessing the accuracy of a clustering by measuring its predictive power. However this works only for generative models, and thus can not be used to compare, e.g., a generative model to a heuristic clustering algorithm.

In Chapter 2 [Haider 09], the found clustering of emails into campaigns could in principle be evaluated by itself, although that would involve costly manual labeling of the test data. Using the usefulness of the clustering as an intermediate step of spam filtering is much cheaper, since emails labeled as spam or non-spam are readily available. And this is better than using some intrinsic measure of clustering quality, like Bayesian information criterion [Schwarz 78]. Intrinsic measures are not application-specific, and there is no guarantee that a particular criterion works well in a certain domain.

The application of finding botnets in Chapter 3 [Haider 12b] does not lend itself to a supervised evaluation of the found clusterings by itself. Emails arrive from arbitrary hosts that are not under the experimenter's control, and thus he or she cannot obtain the information to which botnet each host actually belongs. In Section 3.4 we evaluate the clusterings using only observable information, by measuring the accuracy of predicting the campaign of an email given its IP address, utilizing the distributions over campaigns and botnets arising from the inferred clustering. This method of evaluation is the only alternative to intrinsic evaluation criteria, which have the aforementioned drawbacks. Prior published work does not evaluate the found clusterings at all, and instead is restricted to qualitative analysis.

For the application of learning a classifier from data with missing values studied in Chapter 4 [Dick 08], there exists the theoretical possibility of evaluating the process that generates the distribution over imputations, but not the imputations per se. If one would take a subset of the training examples with complete data, eradicate some of the observations, and then apply any method that deals with missing values by imputing one or more values, then one could compare the found imputations with the original, erased values. We have not done so in our experiments because,

in the application, only the resulting predictive accuracy is of importance. In this case, measuring predictive accuracy not only allows us to indirectly evaluate the distribution over imputations for training examples which are actually incompletely observed, but the evaluation criterion is even more relevant to the application than a direct evaluation of the imputations.

A different case presents itself with the abstract problem setting of market segmentation in Chapter 5 [Haider 12a]. There exist many suggestions of a priori criteria to segment markets. But these are mostly concerned about homogeneity and not separability. Even homogeneity is only directly pursued with respect to for example demographic attributes, when in fact homogeneity is desired with respect to future behavior that is relevant to the vendor-client relationship. In Section 5.3 the segments are optimized with respect to exactly the observed behavior of the training examples. Also, separability is taken into account in the only natural way. Instead of trading off separability and homogeneity in an ad hoc way, the predictive power of the model when actually applied to separating the training examples is optimized. Thus the optimization criterion itself takes into account the usefulness of the found partitioning.

Bibliography

[Attias 99] Hagai Attias. *Inferring parameters and structure of latent variable models by variational Bayes.* In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, pages 21–30. Morgan Kaufmann Publishers Inc., 1999.

[Blei 03] David M Blei, Andrew Y Ng & Michael I Jordan. *Latent dirichlet allocation.* Journal of Machine Learning Research, vol. 3, pages 993–1022, 2003.

[Dempster 77] Arthur P Dempster, Nan M Laird & Donald B Rubin. *Maximum likelihood from incomplete data via the EM algorithm.* Journal of the Royal Statistical Society. Series B (Methodological), pages 1–38, 1977.

[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.

[Enzweiler 11] Markus Enzweiler & Dariu M Gavrila. *A multilevel mixture-of-experts framework for pedestrian classification.* IEEE Transactions on Image Processing, vol. 20, no. 10, pages 2967–2979, 2011.

[Geweke 11] John Geweke & Gianni Amisano. *Hierarchical Markov normal mixture models with applications to financial asset returns.* Journal of Applied Econometrics, vol. 26, no. 1, pages 1–29, 2011.

[Griffiths 06] Tom Griffiths & Zoubin Ghahramani. *Infinite latent feature models and the Indian buffet process.* In Advances in Neural Information Processing Systems 18, pages 475–482. MIT Press, Cambridge, MA, 2006.

[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.

[Haider 12a] Peter Haider, Luca Chiarandini & Ulf Brefeld. *Discriminative clustering for market segmentation.* In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 417–425. ACM, 2012.

[Haider 12b] Peter Haider & Tobias Scheffer. *Finding Botnets Using Minimal Graph Clusterings*. In Proceedings of the 29th International Conference on Machine Learning, pages 847–854, 2012.

[Heller 05] Katherine A Heller & Zoubin Ghahramani. *Bayesian hierarchical clustering*. In Proceedings of the 22nd International Conference on Machine Learning, pages 297–304. ACM, 2005.

[Hinton 06] Geoffrey E Hinton & Ruslan R Salakhutdinov. *Reducing the dimensionality of data with neural networks*. Science, vol. 313, no. 5786, pages 504–507, 2006.

[Hinton 12] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever & Ruslan Salakhutdinov. *Improving neural networks by preventing co-adaptation of feature detectors*. CoRR, vol. abs/1207.0580, 2012.

[Jordan 94] Michael I Jordan & Robert A Jacobs. *Hierarchical mixtures of experts and the EM algorithm*. Neural Computation, vol. 6, no. 2, pages 181–214, 1994.

[Lee 09] Honglak Lee, Roger Grosse, Rajesh Ranganath & Andrew Y Ng. *Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations*. In Proceedings of the 26th Annual International Conference on Machine Learning, pages 609–616. ACM, 2009.

[MacDonald 12] Ian M MacDonald. Protein structural domain boundary prediction: A mixture of experts approach. AV Akademikerverlag, 2012.

[McNicholas 10] Paul D McNicholas & Thomas Brendan Murphy. *Model-based clustering of microarray expression data via latent Gaussian mixture models*. Bioinformatics, vol. 26, no. 21, pages 2705–2712, 2010.

[Mohamed 11] A-R Mohamed, Tara N Sainath, George Dahl, Bhuvana Ramabhadran, Geoffrey E Hinton & Michael A Picheny. *Deep belief networks using discriminative features for phone recognition*. In IEEE International Conference on Acoustics, Speech and Signal Processing, pages 5060–5063. IEEE, 2011.

[Mohamed 12] Abdel-rahman Mohamed, George E Dahl & Geoffrey Hinton. *Acoustic modeling using deep belief networks*. IEEE Transactions on Audio, Speech, and Language Processing, vol. 20, no. 1, pages 14–22, 2012.

[Ng 02] Andrew Y Ng, Michael I Jordan, Yair Weiss*et al. On spectral clustering: Analysis and an algorithm*. Advances in Neural Information Processing Systems, vol. 2, pages 849–856, 2002.

[Povey 10] Daniel Povey, Lukáš Burget, Mohit Agarwal, Pinar Akyazi, Kai Feng, Arnab Ghoshal, Ondrej Glembek, Nagendra Kumar Goel, Martin Karafiát, Ariya Rastrow*et al. Subspace Gaussian mixture*

models for speech recognition. In IEEE International Conference on Acoustics Speech and Signal Processing, pages 4330–4333. IEEE, 2010.

[Schwarz 78] Gideon Schwarz. *Estimating the dimension of a model*. The Annals of Statistics, vol. 6, no. 2, pages 461–464, 1978.

[Smith 93] Adrian FM Smith & Gareth O Roberts. *Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods*. Journal of the Royal Statistical Society. Series B (Methodological), pages 3–23, 1993.

[Socher 11] Richard Socher, Cliff C. Lin, Andrew Y. Ng & Christopher D. Manning. *Parsing Natural Scenes and Natural Language with Recursive Neural Networks*. In Proceedings of the 26th International Conference on Machine Learning (ICML), 2011.

[Socher 12] Richard Socher, Brody Huval, Bharath Bath, Christopher D Manning & Andrew Ng. *Convolutional-recursive deep learning for 3D object classification*. In Advances in Neural Information Processing Systems, pages 665–673, 2012.

[Steyvers 07] Mark Steyvers & Tom Griffiths. *Probabilistic topic models*. Handbook of latent semantic analysis, vol. 427, no. 7, pages 424–440, 2007.

[Subasi 07] Abdulhamit Subasi. *EEG signal classification using wavelet feature extraction and a mixture of expert model*. Expert Systems with Applications, vol. 32, no. 4, pages 1084–1093, 2007.

[Teh 06] Yee Whye Teh, Michael I Jordan, Matthew J Beal & David M Blei. *Hierarchical dirichlet processes*. Journal of the American Statistical Association, vol. 101, no. 476, 2006.

[Teh 10] Yee Whye Teh & Michael I Jordan. *Hierarchical Bayesian nonparametric models with applications*. Bayesian Nonparametrics: Principles and Practice, pages 158–207, 2010.

[Thalamuthu 06] Anbupalam Thalamuthu, Indranil Mukhopadhyay, Xiaojing Zheng & George C Tseng. *Evaluation and comparison of gene clustering methods in microarray analysis*. Bioinformatics, vol. 22, no. 19, pages 2405–2412, 2006.

[Wallach 09] Hanna M Wallach, Iain Murray, Ruslan Salakhutdinov & David Mimno. *Evaluation methods for topic models*. In Proceedings of the 26th Annual International Conference on Machine Learning, pages 1105–1112. ACM, 2009.

[Wei 06] Xing Wei & W Bruce Croft. *LDA-based document models for ad-hoc retrieval*. In Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 178–185. ACM, 2006.

[Yu 09] Chun-Nam John Yu & Thorsten Joachims. *Learning structural SVMs with latent variables*. In Proceedings of the 26th Annual International Conference on Machine Learning, pages 1169–1176. ACM, 2009.

[Yu 12] Dong Yu, Frank Seide & Gang Li. *Conversational Speech Transcription Using Context-Dependent Deep Neural Networks*. In Proceedings of the 29th International Conference on Machine Learning, 2012.

[Zhang 11] Cha Zhang, Michael Cohen, Yong Rui & Ting Yu. *Image segmentation using spatial-color Gaussian mixture models*, February 8 2011. US Patent 7,885,463.