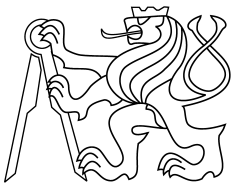
CENTER FOR
MACHINE PERCEPTION



CZECH TECHNICAL
UNIVERSITY IN PRAGUE

RESEARCH REPORT

ISSN 1213-2365

Learning data discretization via convex optimization

Vojtech Franc[1], Ondrej Fikar[1], Karel Bartos[2], Michal Sofka[2]

xfrancv@cmp.felk.cvut.cz

[1] Center for Machine Perception and [2] CISCO Systems

CTU–CMP–2016–01

January 28, 2016

Available at
ftp://cmp.felk.cvut.cz/pub/cmp/articles/franc/Franc-TR-2016-01.pdf

The project was sponsored by CISCO under project 830 1351C001.

Research Reports of CMP, Czech Technical University in Prague, No. 1, 2016

Published by

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

Learning data discretization via convex optimization

Vojtech Franc[1], Ondrej Fikar[1], Karel Bartos[2], Michal Sofka[2]

January 28, 2016

Abstract

Discretization of continuous input functions into piecewise constant or piecewise linear approximations is needed in many mathematical modeling problems. It has been shown that choosing the length of the piecewise segments adaptively based on data samples leads to improved accuracy of the subsequent processing such as classification. Traditional approaches are often tied to a particular classification model which results in local greedy optimization of a criterion function. This paper proposes a technique for learning the discretization parameters along with the parameters of a decision function in a convex optimization of the true objective. The general formulation is applicable to a wide range of learning problems. Empirical evaluation demonstrates that the proposed convex algorithms yield models with fewer number of parameters with comparable or better accuracy than the existing methods.

1 Introduction

Many mathematical modeling problems involve discretization of continuous input functions to convert them into their discrete counterparts (García et al., 2013; Liu et al., 2002). The discrete functions are then piece-wise constant (0th order) or piece-wise linear (1st order) approximations of the value function. For example, probability density functions are typically represented as multi-dimensional histograms in the discrete domain. In another example, the domain of input features is discretized in order to apply a linear decision function as in logistic regression or Support Vector Machine classification. Clearly, the accuracy of the decision functions directly depends on the discretization of the feature values. One common difficulty in the discretization process is the choice of the discretization step which then indicates the size of the piece-wise segments, e.g. histogram bins, or parameters of the feature representation quantization. The parameters of the decision function are estimated in a separate subsequent process (Dougherty et al., 1995; Pele et al., 2013).

Most algorithms employ the simplest unsupervised discretization by choosing a fixed number of bins with the same size (*equal interval width*) or the same number of samples in each bin (*equal frequency interval*) (Dougherty et al., 1995). The total number of bins is tuned for a specific application balancing two opposite considerations. Wider bins reduce the effects of noise in regions where the number of input samples is low. On the other hand, narrower bins result in

more accurate function approximation in regions where there are many samples and thus the effects of the noise are suppressed. Equal frequency intervals have been extended by using Shannon entropy over discretized space to adjust the bin boundaries (Dougherty et al., 1995). Apparently, using varying bin sizes during discretization can be beneficial.

Supervised discretization algorithms use sample labels to improve the binning (Kerber, 1992; Dougherty et al., 1995), often in an optimization step when learning a classifier (Boullé, 2006; Fayyad and Irani, 1992; Friedman and Goldszmidt, 1996). One widely-adopted approach is to initially start with a large number of bins and then merging neighboring bins while optimizing a criterion function (Boullé, 2006). In (Boullé, 2006), the discretization is based on a search algorithm to find a Bayesian optimal interval splits using a prior distribution an a model space. In (Fayyad and Irani, 1992), the recursive splitting is based on an information entropy minimization heuristic. The algorithm is extended with a Minimum Description Length stopping criteria in (Fayyad and Irani, 1993) and embedded into a dynamic programming algorithm in (Elomaa and Rousu, 1999). These techniques introduce supervision for finding the optimal discretization but are tied to a particular classification model (Naïve Bayes, decision trees, or Bayesian Networks (Friedman and Goldszmidt, 1996; Yang and Webb, 2008)). As a result, they rely on local greedy optimization of the criterion function (Hue and Boullé, 2007).

This paper proposes an algorithm for learning piece-wise constant or piece-wise linear embeddings from data samples along with the parameters of a decision function. Similarly to several previous techniques, the initial fine-grained discretization with many histogram bins is adjusted by optimizing a criterion function. In addition, our algorithm proposes several important contributions. First, when training a decision function, the algorithm optimizes the true objective (or its surrogate) which includes the discretization parameters with the parameters of the decision function. This is in contrast to previous methods that rely on two separate steps, discretization and classifier learning, which can deviate from the true objective (Pele et al., 2013). Second, the parameter learning is transformed into a convex optimization problem that can be solved effectively. Other techniques do not provide a global solution and resort to greedy strategy, where the features are processed sequentially (Hue and Boullé, 2007). Third, our formulation is general with piece-wise embeddings being used when training a linear decision function making it applicable to a wide range of linear models. Other methods are specific to a particular classification model (Boullé, 2006; Fayyad and Irani, 1992; Friedman and Goldszmidt, 1996; Yang and Webb, 2008).

Our experiments demonstrate that the learned discretization is effective when applied to various data representation problems. The first set of results combines the estimation of the representation parameters with learning a linear decision function in a joint convex optimization problem. The learned linear embedding has much lower dimensionality than equidistant discretization (by a factor of three on average) to achieve the same or better malware detection accuracy level. The second set of results shows piece-wise linear approximation of a probability density function using non-equidistant bins when estimating density from data samples. The proposed algorithm achieves lower KL-divergence between the estimate and the ground truth than histograms with equidistant bins. Comparison to previously published piece-wise linear embedding for non-

linear classification (Pele et al., 2013) shows higher accuracy of the proposed technique on a number of datasets from the UCI repository. These encouraging results show promise of the technique which could be extended to other linear embedding and classification problems.

2 Learning piece-wise constant functions

We consider learning a univariate piece-wise constant (PWC) function

$$f_{\text{pwc}}(x; \boldsymbol{w}, \boldsymbol{\theta}) = \sum_{i=1}^{B} [\![x \in [\theta_{i-1}, \theta_i)]\!] w_i = w_{k(x,\boldsymbol{\theta})} \tag{1}$$

where $x \in \mathbb{R}$ is the input variable, B is the number of bins, $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_B)^T \in \mathbb{R}^{B+1}$ is a vector defining the bin edges, $\boldsymbol{w} = (w_1, \ldots, w_B)^T \in \mathbb{R}^B$ is a vector of weights each of them associated to a single bin and the function

$$k(x, \boldsymbol{\theta}) = 1 + \min\{i \in \{0, \ldots, B-1\} \mid x \geq \theta_i\}$$

returns the bin number to which the variable x falls to. For notational convenience, we omit the additive scalar bias w_0 in the definition (1) which does not affect the discussion that follows. We denote set inclusive bracket as '[' and exclusive bracket as ')'. The operator $[\![A]\!]$ evaluates to 1 if the logical statement A is true and it is zero otherwise.

Let $g \colon \mathbb{R}^m \to \mathbb{R}$ be a convex function whose value $g(f(x^1), \ldots, f(x^m))$ measures how well the function $f \colon \mathbb{R} \to \mathbb{R}$ evaluated on m training inputs $\mathcal{T} = \{x_1, \ldots, x_m\} \in \mathbb{R}^m$ explains the data. For example, g can be the empirical risk or the maximum likelihood of f evaluated on $\mathcal{T}$. Assuming the bin edges $\boldsymbol{\theta}$ are fixed, the weights $\boldsymbol{w}$ of the PWC function (1) can be learned by solving the minimization problem

$$\boldsymbol{w}^* \in \operatorname*{argmin}_{\boldsymbol{w} \in \mathbb{R}^B} F_{\text{pwc}}(\boldsymbol{w}, \boldsymbol{\theta}) \tag{2a}$$

where

$$F_{\text{pwc}}(\boldsymbol{w}, \boldsymbol{\theta}) = g\big(f_{\text{pwc}}(x^1; \boldsymbol{w}, \boldsymbol{\theta}), \ldots, f_{\text{pwc}}(x^m; \boldsymbol{w}, \boldsymbol{\theta})\big) \tag{2b}$$

is convex in $\boldsymbol{w}$ since it is a composition of a convex function g and f_{pwc} which is linear in $\boldsymbol{w}$ (Boyd and Vandenberghe, 2004).

In practice the bin edges $\boldsymbol{\theta}$ are often selected before learning $\boldsymbol{w}$. The simplest way is to use equidistantly spaced bins,

$$\theta_i = i(\text{Max} - \text{Min})/B + \text{Min}\,, \quad \forall i \in \{0, \ldots, B\}\,,$$

where Min and Max denote the minimal and the maximal attainable value of the input variable x, respectively. The bin edges are constructed for different values of B and the optimal setting is typically tuned on validation examples. This procedure involves minimization of $F_{\text{pwc}}(\boldsymbol{w}, \boldsymbol{\theta})$ for all proposal discretizations $\boldsymbol{\theta}$. In principle, one could optimize the width of individual bins as well, however, this would generate exponentially many proposal discretizations making this naive approach intractable.

We propose a method which can simultaneously learn the discretization $\boldsymbol{\theta}$ and the weights $\boldsymbol{w}$ of the PWC function (1). The important feature is that the

resulting bins do not have to be equidistant. We only assume that the bin edges are selected from a finite monotonically increasing sequence of real numbers $\nu_0 < \nu_1 < \cdots < \nu_D$, in further text represented by a vector $\boldsymbol{\nu} = (\nu_0, \ldots, \nu_D)^T \in \mathbb{R}^{D+1}$ and denoted as the initial discretization. The set of admissible discretizations $\Theta_B \subset \mathbb{R}^{B+1}$ of the variable $x \in \mathbb{R}$ into B bins contains all vectors $\boldsymbol{\theta}$ satisfying:

$$\theta_i = \nu_{l_i} \,, i \in \{0, \ldots, B\}, \tag{3a}$$

$$\left.\begin{array}{l} l_0 = 0 \,, \\ l_i < l_{i+1} \,, \quad i \in \{1, \ldots, B-1\} \,, \\ l_B = D \,, \end{array}\right\} \tag{3b}$$

The equation (3a) says that the bin edges $\boldsymbol{\theta} \in \Theta_B$ form a subset of the initial discretization $\boldsymbol{\nu}$. The indices $\{l_0, \ldots, l_B\}$ define which edges are selected from $\boldsymbol{\nu}$. The equations (3b) state that the left most bin edge is ν_0, the right most edge is ν_D, and the intermediate edges form an increasing sequence.

Given a convex learning algorithm (2), we propose to learn the discretization $\boldsymbol{\theta}^* \in \Theta_B$ simultaneously with the weights $\boldsymbol{w}^* \in \mathbb{R}^B$ by solving

$$(\boldsymbol{w}^*, \boldsymbol{\theta}^*) \in \operatorname*{argmin}_{\boldsymbol{w} \in \mathbb{R}^B, \boldsymbol{\theta} \in \Theta_B} F_{\text{pwc}}(\boldsymbol{w}, \boldsymbol{\theta}) \,. \tag{4}$$

Unlike (2) where $\boldsymbol{\theta}$ is fixed, the problem (4) is almost always non-convex and hard to optimize. In the sequel we derive a convex approximation of (4) which can be solved efficiently provided the original problem (2) can be solved efficiently.

Let us define a function $c\colon \mathbb{R}^D \to \{0, \ldots, D-1\}$

$$c(\boldsymbol{v}) = \sum_{i=1}^{D-1} [\![v_i \neq v_{i+1}]\!]$$

returning the total number of different neighboring elements of the vector $\boldsymbol{v} \in \mathbb{R}^D$. Let Θ_B be induced by the initial discretization $\boldsymbol{\nu}$ as defined by (3). It is easy to see that for any $(\boldsymbol{w}, \boldsymbol{\theta}) \in (\mathbb{R}^B \times \Theta_B)$ there exists a unique $\boldsymbol{v} \in \mathrm{V}_B = \left\{ \boldsymbol{v} \in \mathbb{R}^D \middle| c(\boldsymbol{v}) \leq B - 1 \right\}$ such that

$$f_{\text{pwc}}(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\theta}) = f_{\text{pwc}}(\boldsymbol{x}; \boldsymbol{v}, \boldsymbol{\nu}) \,, \quad \forall \boldsymbol{x} \in \mathbb{R}^n \,, \tag{5}$$

holds. In particular, $\boldsymbol{v}$ is constructed from $(\boldsymbol{w}, \boldsymbol{\theta})$ by

$$v_l = f_{\text{pwc}}(\nu_l; \boldsymbol{w}, \boldsymbol{\theta}) \,, \qquad l \in \{1, \ldots, D\} \,. \tag{6}$$

Computing $(\boldsymbol{w}, \boldsymbol{\theta})$ from $(\boldsymbol{v}, \boldsymbol{\nu})$ is also straightforward but not unique in general. It can be seen that for any $\boldsymbol{v} \in \mathrm{V}_B$ it is possible to construct a finite number of pairs $\mathcal{W}(\boldsymbol{v}) = \{(\boldsymbol{w}_1, \boldsymbol{\theta}_1), \ldots, (\boldsymbol{w}_L, \boldsymbol{\theta}_L)\} \in (\mathbb{R}^B \times \Theta_B)^L$, such that the equality (5) holds for any $(\boldsymbol{w}, \boldsymbol{\theta}) \in \mathcal{W}(\boldsymbol{v})$. Provided $c(\boldsymbol{v}) = B - n$ the conversion from $\boldsymbol{v}$ and $(\boldsymbol{w}, \boldsymbol{\theta})$ is unique, i.e. $|\mathcal{W}(\boldsymbol{v})| = 1$. The parametrization $(\boldsymbol{w}, \boldsymbol{\theta})$ is a shortened representation of $(\boldsymbol{u}, \boldsymbol{\nu})$ which is composed of sub-sequences of the equal components. Therefore we will denote $(\boldsymbol{v}, \boldsymbol{\nu})$ as the uncompressed parametrization and $(\boldsymbol{w}, \boldsymbol{\theta})$ as the compressed parametrization. The equivalence between the compressed parametrization and the uncompressed one is illustrated in Figure 1. The equivalence implies that

$$\min \left\{ F_{\text{pwc}}(\boldsymbol{w}; \boldsymbol{\theta}) \mid (\boldsymbol{w}, \boldsymbol{\theta}) \in (\mathbb{R}^B \times \Theta_B) \right\} = \min \left\{ F_{\text{pwc}}(\boldsymbol{v}; \boldsymbol{\nu}) \mid \boldsymbol{v} \in \mathrm{V}_B \right\}$$

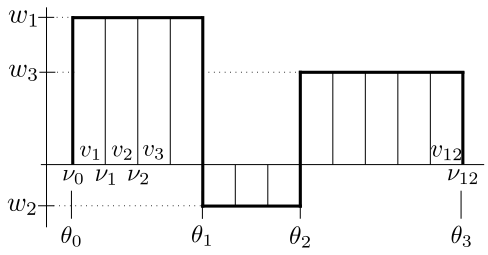


Figure 1: The figure shows an example of the PWC function and its two equivalent parametrizations which guarantee that $f_{\text{pwc}}(x; \boldsymbol{w}, \boldsymbol{\theta}) = f_{\text{pwc}}(x; \boldsymbol{v}, \boldsymbol{\nu})$, $\forall x \in \mathbb{R}$. The function $f_{\text{pwc}}(x; \boldsymbol{w}, \boldsymbol{\theta})$ has 3 bins of heights $\boldsymbol{w} = (w_1, w_2, w_3)^T$ and 4 edges $\boldsymbol{\theta} = (\theta_0, \theta_1, \theta_2, \theta_3)^T = (\nu_0, \nu_4, \nu_7, \nu_{12})^T$ selected out of the sequence $\boldsymbol{\nu} = (\nu_0, \ldots, \nu_{12})^T$. The function $f_{\text{pwc}}(x; \boldsymbol{v}, \boldsymbol{\nu})$ has 11 bins with 12 edges $\boldsymbol{\nu}$ and heights $\boldsymbol{v} = (w_1, w_1, w_1, w_1, w_2, w_2, w_2, w_3, w_3, w_3, w_3, w_3)^T$. Though $\boldsymbol{v}$ has 11 components there are only two places where they differ, hence $c(\boldsymbol{v}) = 2$.

and therefore the problem (4) can be solved by finding

$$\boldsymbol{v}^* \in \operatorname*{argmin}_{\boldsymbol{v} \in \mathbb{R}^D} F_{\text{pwc}}(\boldsymbol{v}, \boldsymbol{\nu}) \quad \text{s.t.} \quad c(\boldsymbol{v}) \leq B - 1 \,, \tag{7}$$

and solving (6) for $(\boldsymbol{w}^*, \boldsymbol{\theta}^*)$ while $\boldsymbol{v}$ is set to $\boldsymbol{v}^*$. Methods for efficient computation of the compressed parametrization $(\boldsymbol{w}^*, \boldsymbol{\theta}^*)$ from the uncompressed one $(\boldsymbol{v}^*, \boldsymbol{\nu})$ are subject of Section 4.

The problem (7) has a convex objective but its single constraint remains non-convex. By introducing a vector $\boldsymbol{d} = (v_1 - v_2, \ldots, v_{D-1} - v_D)^T$ we see that the function $c(\boldsymbol{v})$ can be written as the L_0-norm of the vector $\boldsymbol{d} = (v_1 - v_2, \ldots, v_{D-1} - v_D)^T$, i.e. $c(\boldsymbol{v}) = \|\boldsymbol{d}\|_0$. It has been demonstrate that the L_1-norm is often a good convex proxy of the L_0-norm (see e.g. (Candes and T.Tao, 2005; Candes et al., 2006; Donoho, 2006)). Therefore we propose to approximate the non-convex function $c(\boldsymbol{v}) = \|\boldsymbol{v}\|_0$ by the convex function $\tilde{c}(\boldsymbol{v}) = \|\boldsymbol{d}\|_1$. A convex relaxation of the problem (7) then reads

$$\boldsymbol{v}^* \in \operatorname*{argmin}_{\boldsymbol{v} \in \mathbb{R}^D} F_{\text{pwc}}(\boldsymbol{v}; \boldsymbol{\nu}) \quad \text{s.t.} \quad \sum_{i=1}^{D-1} |v_i - v_{i+1}| \leq B - 1 \,,$$

or equivalently we can solve an unconstrained problem

$$\boldsymbol{v}^* \in \operatorname*{argmin}_{\boldsymbol{v} \in \mathbb{R}^D} \left[F_{\text{pwc}}(\boldsymbol{v}; \boldsymbol{\nu}) + \gamma \sum_{i=1}^{D-1} |v_i - v_{i+1}| \right] , \tag{8}$$

where the constant $\gamma \in \mathbb{R}_+$ is monotonically dependent on $B - 1$.

3 Learning piece-wise linear functions

The previous section shows how to transform learning of PWC functions to a convex optimization problem. A similar idea can be applied to learning piece-wise linear (PWL) functions which provide more accurate approximation of the

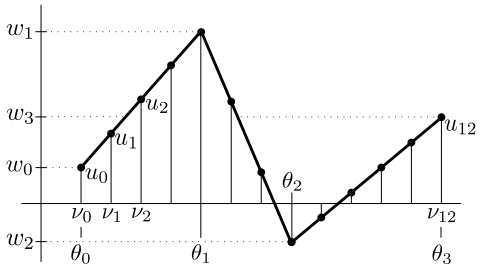


Figure 2: The figure shows an example of the PWL function and its two equivalent parametrizations which guarantee that $f_{\text{pwl}}(x; \boldsymbol{w}, \boldsymbol{\theta}) = f_{\text{pwl}}(x; \boldsymbol{u}, \boldsymbol{\nu}), \forall x \in \mathbb{R}$. The function $f_{\text{pwl}}(x; \boldsymbol{w}, \boldsymbol{\theta})$ is described by 3 line segments connecting 4 points $\{(\theta_0, w_0), \ldots, (\theta_3, w_3)\}$ whose coordinates are stored in $\boldsymbol{w} = (w_0, \ldots, w_3)^T$ a $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_3)^T$. The bin edges $\boldsymbol{\theta} = (\theta_0, \theta_1, \theta_2, \theta_3)^T = (\nu_0, \nu_4, \nu_7, \nu_{12})^T$ form a subset of $\boldsymbol{\nu} = (\nu_0, \ldots, \nu_{12})^T$. The function $f_{\text{pwl}}(x; \boldsymbol{u}, \boldsymbol{\nu})$ is described by 11 line segments connecting points $\{(\nu_0, u_0), \ldots, (\nu_{12}, u_{12})\}$ whose coordinates are stored in $\boldsymbol{u} = (u_0, \ldots, u_{12})^T$ and $\boldsymbol{\nu}$. Note, however, that only 4 components of $\boldsymbol{u}$ cannot be expressed as an average of its neighbors, hence $e(\boldsymbol{u}) = 4$.

input data. In this case, we want to learn a function

$$f_{\text{pwl}}(x; \boldsymbol{w}, \boldsymbol{\theta}) = w_{k(x,\boldsymbol{\theta})-1} \cdot (1 - \alpha(x, \boldsymbol{\theta})) + w_{k(x,\boldsymbol{\theta})} \cdot \alpha(x, \boldsymbol{\theta})\big) \tag{9}$$

where $x \in \mathbb{R}$ is the input variable as for before, $\boldsymbol{\theta} \in \mathbb{R}^{B+1}$ is a vector defining the bin edges, B is the number of bins and $\boldsymbol{w} \in \mathbb{R}^{B+1}$ is the vector of weights [1]. The function $\alpha \colon \mathbb{R} \times \mathbb{R}^{B+1} \to [0, 1]$, defined as

$$\alpha(x, \boldsymbol{\theta}) = \frac{x - \theta_{k(x,\boldsymbol{\theta})-1}}{\theta_{k(x,\boldsymbol{\theta})} - \theta_{k(x,\boldsymbol{\theta})-1}}$$

is a normalized distance between x and the right edge of the $k(x, \boldsymbol{\theta})$-th bin.

Analogically to the previous section we want to learn simultaneously the discretization $\boldsymbol{\theta}^* \in \Theta_B$ and the weights $\boldsymbol{w}^* \in \mathbb{R}^{B+1}$ by solving

$$(\boldsymbol{w}^*, \boldsymbol{\theta}^*) \in \operatorname*{argmin}_{\boldsymbol{w} \in \mathbb{R}^{B+1}, \boldsymbol{\theta} \in \Theta_B} F_{\text{pwl}}(\boldsymbol{w}, \boldsymbol{\theta}) \tag{10}$$

where

$$F_{\text{pwl}}(\boldsymbol{w}, \boldsymbol{\theta}) = g\big(f_{\text{pwl}}(x^1; \boldsymbol{w}, \boldsymbol{\theta}), \ldots, f_{\text{pwl}}(x^m; \boldsymbol{w}, \boldsymbol{\theta})\big)$$

is the learning objective depending on the responses of the PWL function (9) evaluated on the training inputs $\{x^1, \ldots, x^m\} \in \mathbb{R}^m$. In the sequel, we derive a convex relaxation of (10).

Let us define a function $e \colon \mathbb{R}^{D+1} \to \{0, \ldots, D+1\}$,

$$e(\boldsymbol{u}) = \sum_{i=1}^{D-1} [\![u_i \neq \frac{1}{2}(u_{i-1} + u_{i+1})]\!]$$

[1] Note that the PWC function has B weights associated with the bins while the PWL function has $B + 1$ weights associated with the bin edges.

which retuns the number of weights that cannot be expressed as an average of neighboring weights. Let Θ_B be induced by the initial discretization $\boldsymbol{\nu}$ as defined by (3). It can be seen that for any $(\boldsymbol{w}, \boldsymbol{\theta}) \in (\mathbb{R}^{B+1} \times \Theta_B)$ we can construct a unique $\boldsymbol{u} \in \mathcal{U}_B = \{\boldsymbol{u} \in \mathbb{R}^{D+1} \mid e(\boldsymbol{u}) \leq B-1\}$ such that

$$f_{\text{pwl}}(x; \boldsymbol{w}, \boldsymbol{\theta}) = f_{\text{pwl}}(x; \boldsymbol{u}, \boldsymbol{\nu}), \quad \forall x \in \mathbb{R}^n \tag{11}$$

holds, in particular, $\boldsymbol{u}$ is constructed from $(\boldsymbol{w}, \boldsymbol{\theta})$ by

$$u_l = f_{\text{pwl}}(\nu_l; \boldsymbol{w}, \boldsymbol{\theta}), \quad l \in \{0, \dots, D\}. \tag{12}$$

In addition, for any $\boldsymbol{u} \in \mathcal{U}_B$ it is possible to construct a finite number of pairs $\mathcal{W}'(\boldsymbol{u}) = \{(\boldsymbol{w}_1, \boldsymbol{\theta}_1), \cdots, (\boldsymbol{w}_L, \boldsymbol{\theta}_L)\} \in (\mathbb{R}^{B+1} \times \Theta_B)^L$ such that the equality (11) hold for any $(\boldsymbol{w}, \boldsymbol{\theta}) \in \mathcal{W}'(\boldsymbol{u})$. For $e(\boldsymbol{u}) = B-1$ the conversion from $\boldsymbol{u}$ to $(\boldsymbol{w}, \boldsymbol{\theta})$ is unique, i.e. $|\mathcal{W}(\boldsymbol{u}) = 1|$. The equivalence between the compressed parametrization $(\boldsymbol{w}, \boldsymbol{\theta})$ and the uncompressed parametrization $(\boldsymbol{u}, \boldsymbol{\nu})$ is illustrated in Figure 2. The equivalence implies that

$$\min\left\{F_{\text{pwl}}(\boldsymbol{w}, \boldsymbol{\theta}) \mid (\boldsymbol{w}, \boldsymbol{\theta}) \in (\mathbb{R}^{B+1} \times \Theta_B)\right\} = \min\left\{F_{\text{pwl}}(\boldsymbol{u}, \boldsymbol{\nu}) \mid \boldsymbol{u} \in \mathcal{U}_B\right\}.$$

Consequently, the problem (10) can be solved by

$$\boldsymbol{u}^* \in \operatorname*{argmin}_{\boldsymbol{u} \in \mathbb{R}^{D+1}} F_{\text{pwl}}(\boldsymbol{u}, \boldsymbol{\nu}) \quad \text{s.t.} \quad e(\boldsymbol{u}) \leq B-1, \tag{13}$$

and computing $(\boldsymbol{w}^*, \boldsymbol{\theta}^*)$ from $\boldsymbol{u}^*$ by solving the equation (12) for $(\boldsymbol{w}, \boldsymbol{\theta})$ with $\boldsymbol{u} = \boldsymbol{u}^*$. Efficient methods computing the compressed parametrization $(\boldsymbol{w}^*, \boldsymbol{\theta}^*)$ from the uncompressed one $(\boldsymbol{u}^*, \boldsymbol{\nu})$ are discussed in Section 4. As before, replacing the L_0-norm in the non-convex constraint $e(\boldsymbol{u}) = B-1$ by the L_1-norm, we obtain a convex relaxation of (10) which reads

$$\boldsymbol{u}^* \in \operatorname*{argmin}_{\boldsymbol{u} \in \mathbb{R}^{D+1}} F_{\text{pwl}}(\boldsymbol{u}, \boldsymbol{\nu}) \quad \text{s.t} \quad \sum_{i=1}^{D-1} \left|u_i - \frac{1}{2}u_{i-1} - \frac{1}{2}u_{i+1}\right| \leq B-1,$$

or equivalently we can solve an unconstrained problem

$$\boldsymbol{u}^* \in \operatorname*{argmin}_{\boldsymbol{u} \in \mathbb{R}^{D+1}} \left[F_{\text{pwl}}(\boldsymbol{u}, \boldsymbol{\nu}) + \gamma \sum_{i=1}^{D-1} \left|u_i - \frac{1}{2}u_{i-1} - \frac{1}{2}u_{i+1}\right|\right]. \tag{14}$$

where the constant $\gamma \in \mathbb{R}_+$ is monotonically dependent on $B-1$.

4 Rounding of piece-wise functions

The uncompressed parametrization $(\boldsymbol{v}, \boldsymbol{\nu})$ of the PWC function (e.g. found by the proposed algorithm (8)) can be converted to the compressed one $(\boldsymbol{w}, \boldsymbol{\theta})$ by splitting $\boldsymbol{v}$ into sub-vectors of equal weights, i.e. $\boldsymbol{v} = (\boldsymbol{v}_1^T, \boldsymbol{v}_2^T, \dots, \boldsymbol{v}_B^T)$ where each $\boldsymbol{v}_i$ can be written as $\boldsymbol{v}_i = w_i[1, \dots, 1]^T$. Obtaining the compressed parametrization $(\boldsymbol{w}, \boldsymbol{\theta})$ is then straightforward (see Figure 1). An analogical procedure can be applied for the PWL parametrizations in which case we are searching for sub-vectors whose intermediate components can be expressed as an average of its neighbors. In practice, however, the components of the uncompressed solution can be noisy thanks to the used convex relaxation and

usage of approximate solvers to find the uncompressed parameters. For this reason, it is useful to round the uncompressed solution before its conversion to the compressed one. The rounding procedures for the PWC and the PWL parametrization are described in the next sections.

4.1 Rounding of PWC parametrization

A very crude rounding is obtained by splitting $\boldsymbol{v}$ into the sub-vectors $\boldsymbol{v} = (\boldsymbol{v}_1^T, \boldsymbol{v}_2^T, \dots, \boldsymbol{v}_B^T)$ such that all components of each $\boldsymbol{v}_i$ have the same sign. The rounded uncompressed solution is then

$$\bar{\boldsymbol{v}} = (\mathrm{mean}(\boldsymbol{v}_1)[1, \dots, 1]^T, \dots, \mathrm{mean}(\boldsymbol{v}_B)[1, \dots, 1]^T)^T \tag{15}$$

where the operator $\mathrm{mean}(\boldsymbol{v}')$ returns the mean value of the components of the vector $\boldsymbol{v}'$. Note that this procedure does not require any parameter.

Another method is to find the parameters $\bar{\boldsymbol{v}}$ of the PWC function with B bins which have the shortest Euclidean distance to given $\boldsymbol{v}$ by solving

$$\bar{\boldsymbol{v}} \in \operatorname*{argmin}_{\boldsymbol{v}' \in \mathbb{R}^D} \|\boldsymbol{v} - \boldsymbol{v}'\|^2 \quad \text{s.t.} \quad c(\boldsymbol{v}') = B - 1\,. \tag{16}$$

The problem (16) is the Euclidean projection of $\boldsymbol{v}$ onto the set $\mathrm{V}_B = \left\{\boldsymbol{v} \in \mathbb{R}^D \middle| c(\boldsymbol{v}) \leq B - 1\right\}$. The optimal solution of (16) can be found by the Dynamic programming. To see this, we write the distance $\|\boldsymbol{v} - \boldsymbol{v}'\|^2$ as

$$\|\boldsymbol{v} - \boldsymbol{v}'\|^2 = \sum_{i=1}^{D} (v_i - v'_i)^2 = \sum_{j=1}^{B} \sum_{i=\theta_{j-1}+1}^{\theta_i} (v_i - v'_i)^2 = \sum_{j=1}^{B} R_j(v'_{\theta_{j-1}}, v'_{\theta_j}, \theta_{j-1}, \theta_j)\,.$$

The solution of $\min_{v'_{\theta_{j-1}} = \dots = v'_{\theta_j}} R_j(v'_{\theta_{j-1}}, v'_{\theta_j}, \theta_{j-1}, \theta_j)$ is trivial. The minimization w.r.t. $\boldsymbol{\theta} \in \Theta_B$ can be solved by the dynamic programming since the sub-problems form a chain and each sub-problem shares only a single variable with its neighboring sub-problems. The overall computational time is $\mathcal{O}(D \cdot B^2)$.

Because the number of bins is often unknown a priori, we can search for the minimal number of bins which explain the given solution $\boldsymbol{v}$ with a prescribed precision ε, i.e. we can solve

$$\bar{\boldsymbol{v}} = \operatorname*{argmin}_{B \in \mathcal{N}, \boldsymbol{v}' \in \mathbb{R}^D} B \quad \text{s.t.} \quad \|\boldsymbol{v} - \boldsymbol{v}'\|^2 \leq \varepsilon \quad \text{and} \quad c(\boldsymbol{v}') = B - 1\,. \tag{17}$$

The solution of (17) can be converted to solving the problem (16) with increasing B until the constraint $\|\boldsymbol{v} - \boldsymbol{v}'\|^2 \leq \varepsilon$ is satisfied.

4.2 Rounding PWL function

The parameter vector $\bar{\boldsymbol{u}}$ describing the PWL function with B bins which has the shortest Euclidean distance to given $\boldsymbol{u}$ can be found by solving

$$\bar{\boldsymbol{u}} \in \operatorname*{argmin}_{\boldsymbol{u}' \in \mathbb{R}^{D+1}} \|\boldsymbol{u} - \boldsymbol{u}'\|^2 \quad \text{s.t.} \quad e(\boldsymbol{u}') = B - 1\,. \tag{18}$$

If the number of bins is unknown, we can search for the minimal number of bins which explain the given solution $\boldsymbol{u}$ with a prescribe precision ε by solving

$$\bar{\boldsymbol{u}} = \operatorname*{argmin}_{B \in \mathcal{N}, \boldsymbol{u}' \in \mathbb{R}^{D+1}} B \quad \text{s.t.} \quad \|\boldsymbol{u} - \boldsymbol{u}'\|^2 \leq \varepsilon \quad \text{and} \quad e(\boldsymbol{v}') = B - 1\,. \tag{19}$$

The problems (18) and (19) can be solved exactly by procedures analogical to the ones for the PWC function which were described in previous section and hence omitted for the sake of space.

5 Examples of the proposed framework

The previous sections desribe a generic framework that allows to modify a wide class of convex algorithms so that they can learn the PWC and the PWL functions. In this section, we give three instances of the proposed framework. We also show how the same idea can be applied to learning multi-variate PWC and PWL functions. In particular, we consider learning of linear classifiers of sequential data represented by PWC histograms (Section 5.1), estimation of PWL probability density models (Section 5.2) and learning non-linear classifier via PWL data embedding (Section 5.3).

5.1 Classification of histograms

In many applications the object to be classified is described by a set of sequences sampled from some unknown distributions. A simple yet efficient representation of the sequential data is the normalized PWC histogram which is used as an input of a linear classifier. This classification model has been successfully is used e.g. in computer vision (Dalal and Triggs, 2005) or in the computer security (Bartos and Sofka, 2015) as will be described in Section 6.1.

Assume we are given a training set $\mathcal{T} = \{(\boldsymbol{X}^1, y^1), \ldots, (\boldsymbol{X}^m, y^m)\} \in (\mathbb{R}^{n \times d} \times \{-1, +1\})^m$ where the input matrix $\boldsymbol{X}^i$ describes n sequences each having d elements. The linear classifier $h(\boldsymbol{X}; \boldsymbol{w}, \boldsymbol{\theta}) = \text{sign}(f_{\text{pwc}}(\boldsymbol{X}; \boldsymbol{w}, \boldsymbol{\theta}))$ assigns $\boldsymbol{X}$ into a class based on the sign of the discriminant function

$$f_{\text{pwc}}(\boldsymbol{X}; \boldsymbol{w}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{b_i} \frac{1}{d} \sum_{k=1}^{d} [\![X_{i,k} \in [\theta_{i,j-1}, \theta_{i,j})]\!] w_{i,j} \tag{20}$$

where $\boldsymbol{\theta}_i = (\theta_{i,0}, \ldots, \theta_{i,b_i})^T \in \mathbb{R}^{b_i+1}$ is a vector defining bin edges of the i-th histogram and $\boldsymbol{\theta} = (\boldsymbol{\theta}_1^T, \ldots, \boldsymbol{\theta}_n^T)^T \in \mathbb{R}^{n+B}$ is their concatenation, $B = \sum_{i=1}^{n} b_i$ denotes the total number of bins, $\boldsymbol{w}_i = (w_{i,1}, \ldots, w_{i,b_i})^T \in \mathbb{R}^{b_i}$ are bin heights of the i-th histogram and $\boldsymbol{w} = (\boldsymbol{w}_1^T, \ldots, \boldsymbol{w}_n^T) \in \mathbb{R}^B$ is their concatenation.

In order to learn the bin edges $\boldsymbol{\theta}$ and the weights $\boldsymbol{w}$ simultaneously from examples $\mathcal{T}$ we apply the general framework described in Section 2. In particular, we show how to adapt the Support Vector Machine (SVM) algorithm. First, we define the initial discretization $\boldsymbol{\nu}_i = (\nu_{i,0}, \ldots, \nu_{i,D})^T \in \mathbb{R}^{D+1}$ for each of n histograms. For example, we place $D+1$ edges equidistantly between the minimal Min_i and the maximal Max_i value that can appear in the i-th sequence, so that $\nu_{i,j} = j(\text{Max}_i - \text{Min}_i) + \text{Min}_i$. Second, we combine the SVM objective function

$$F_{\text{pwc}}^{\text{svm}}(\boldsymbol{w}, \boldsymbol{\theta}; \lambda) = \frac{\lambda}{2} \|\boldsymbol{w}\|^2 + \frac{1}{m} \sum_{i=1}^{m} \max\left\{0, 1 - y_i f_{\text{pwc}}(\boldsymbol{x}^i, \boldsymbol{w}, \boldsymbol{\theta})\right\}$$

with the L_1-norm approximation of the function

$$c_n(\boldsymbol{v}) = \sum_{i=1}^{n} \sum_{j=1}^{D-1} [\![v_{i,j} \neq v_{i,j+1}]\!]$$

the value of which equals to the total number of different neighboring components of $\boldsymbol{v}$. Analogically to the general formulation (8) we obtain the following convex program

$$\boldsymbol{v}^* \in \operatorname*{argmin}_{\boldsymbol{v}\in\mathbb{R}^{nD}} \left[F^{\mathrm{svm}}_{\mathrm{pwc}}(\boldsymbol{v}, \boldsymbol{\nu}; \lambda) + \gamma \sum_{i=1}^{n} \sum_{j=1}^{D-1} |v_{i,j} - v_{i,j+1}| \right]. \tag{21}$$

The constant $\lambda \in \mathbb{R}_+$ controls the empirical error. The constant $\gamma \in \mathbb{R}_+$ influences the number of equal neighboring in $\boldsymbol{v}^*$ and thus the number of emerging bins. The uncompressed parametrization $(\boldsymbol{v}^*, \boldsymbol{\nu})$ is converted to the compressed one $(\boldsymbol{w}^*, \boldsymbol{\theta}^*)$ via the rounding methods from Section 4 which are applied to each pair $(\boldsymbol{v}_i^*, \boldsymbol{\nu}_i)$, $i \in \{1, \ldots, n\}$, separately.

5.2 Estimation of PWL histograms

Given a training sample $\mathcal{T} = \{x^1, \ldots, x^m\} \in \mathbb{R}^m$ drawn from i.i.d. random variables with unknown distribution $p(x)$, the goal is to find $\hat{p}(x)$ well approximating $p(x)$ based on the samples $\mathcal{T}$. Assume we want to model the unknown p.d.f. by the PWL function

$$\hat{p}_{\mathrm{pwl}}(x; \boldsymbol{w}, \boldsymbol{\theta}) = \big(1 - \alpha(x, \boldsymbol{\theta})\big) w_{k(x,\boldsymbol{\theta})-1} + \alpha(x, \boldsymbol{\theta}) w_{k(x,\theta)} \tag{22}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{B+1}$ are the bin edges and $\boldsymbol{w} \in \mathbb{R}_+^{B+1}$ is a vector of non-negative weights selected such that $\int \hat{p}_{\mathrm{pwl}}(x; \boldsymbol{w}, \boldsymbol{\theta}) \mathrm{d}x = 1$. To learn the unknown parameters $(\boldsymbol{w}, \boldsymbol{\theta})$ from the sample $\mathcal{T}$, we are going to instantiate the framework proposed in Section 3 to the Maximum Likelihood method. Let the initial discretization $\boldsymbol{\nu} \in \mathbb{R}^{D+1}$ be equidistantly spaced between the minimal and the maximal value, i.e. $\nu_j = j(\mathrm{Max} - \mathrm{Min}) + \mathrm{Min}$, $\forall j \in \{0, \ldots, D\}$, where D is set to be sufficiently high. We substitute the negative log-likelihood

$$F^{\mathrm{nnl}}_{\mathrm{pwl}}(\boldsymbol{w}, \boldsymbol{\theta}) = -\sum_{i=1}^{m} \log \hat{p}_{\mathrm{pwl}}(x^i; \boldsymbol{w}, \boldsymbol{\theta})$$

to the general formulation (14) which yields the following convex problem

$$\boldsymbol{u}^* = \operatorname*{argmin}_{\boldsymbol{u}\in\mathbb{R}^{D}} \left[F^{\mathrm{nnl}}_{\mathrm{pwl}}(\boldsymbol{u}, \boldsymbol{\nu}) + \gamma \sum_{j=1}^{D-1} \left| u_j - \frac{1}{2} u_{j-1} - \frac{1}{2} u_{j+1} \right| \right] \tag{23a}$$

subject to

$$u_0 + u_D + 2 \sum_{i=1}^{D-1} u_i = \frac{2D}{\mathrm{Max} - \mathrm{Min}}\,, \qquad u_i \geq 0\,, \quad i \in \{0, \ldots, D\}\,, \tag{23b}$$

where $\gamma \in \mathbb{R}_+$ is a constant controlling the number of bins. The introduced constraints (23b) ensure that $\int_{\mathrm{Min}}^{\mathrm{Max}} \hat{p}_{\mathrm{pwl}}(x; \boldsymbol{u}, \boldsymbol{\nu}) \mathrm{d}x = 1$ holds for any feasible $\boldsymbol{u}$. The found $\boldsymbol{u}^*$ defines a PWL probability density model $\hat{p}_{\mathrm{pwl}}(x^i; \boldsymbol{u}^*, \boldsymbol{\nu})$. If necessary the compressed parameters $(\boldsymbol{w}^*, \boldsymbol{\theta}^*)$, describing the non-equidistant bins, can be recovered from $\boldsymbol{u}^*$ by the rounding methods described in Section 4.

5.3 PWL embedding for non-linear classification

The PWL embedding (Pele et al., 2013) is a simple yet efficient way to learn highly non-linear decision function by a linear algorithm like for example the Support Vector Machines. Let $\mathcal{T} = \{(\boldsymbol{x}^1, y^1), \ldots, (\boldsymbol{x}^m, y^m)\} \in (\mathbb{R}^n \times \{-1, +1\})^m$ be a training set of input features and output binary labels. A linear classifier $h(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\theta}) = \operatorname{sign}(f_{\text{pwl}}(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\theta}))$ assigns the input $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ into classes based on the sign of the discriminant function

$$f_{\text{pwl}}(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \left(w_{i,k(x_i,\boldsymbol{\theta}_i)-1} \cdot (1 - \alpha(x_i, \boldsymbol{\theta}_i)) + w_{i,k(x_i,\boldsymbol{\theta}_i)} \cdot \alpha(x_i, \boldsymbol{\theta}_i) \right) \quad (24)$$

which is a sum of n PWL functions each defined for a single input feature. The vector $\boldsymbol{\theta}_i \in \mathbb{R}^{b_i+1}$ contains the bin edges of the i-th feature and $\boldsymbol{\theta} = (\boldsymbol{\theta}_1^T, \ldots, \boldsymbol{\theta}_n^T)^T \in \mathbb{R}^{n+B}$ is their concatenation where $B = \sum_{i=1}^{n} b_i$ denotes the total number of bins. The vector $\boldsymbol{w}_i \in \mathbb{R}^{b_i+1}$ contains the weights associated to the edges $\boldsymbol{\theta}_i$ and $\boldsymbol{w} = (\boldsymbol{w}_1^T, \ldots, \boldsymbol{w}_n^T) \in \mathbb{R}^{B+n}$ is their concatenation.

We can learn the weights $\boldsymbol{w}$ as well as the discretization $\boldsymbol{\theta}$ using the SVM algorithm adapted by the framework from Section 3. First, for each input vector we define the initial discretization $\boldsymbol{\nu}_i = (\nu_{i,0}, \ldots, \nu_{i,D})^T \in \mathbb{R}^{D+1}$, e.g. as before by setting $\nu_{i,j} = j(\text{Max}_i - \text{Min}_i) + \text{Min}_i$ where Min_i and Max_i is the minimal and the maximal value of the i-the feature and D is the maximal number of bins per feature. Let $\boldsymbol{\nu} = (\boldsymbol{\nu}_1^T, \ldots, \boldsymbol{\nu}_n^T)^T \in \mathbb{R}^{n(D+1)}$ be the concatenation of initial discretizations for all feature. The SVM objective function

$$F_{\text{pwl}}^{\text{svm}}(\boldsymbol{w}, \boldsymbol{\theta}; \lambda) = \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m} \max\left\{0, 1 - y_i f_{\text{pwl}}(\boldsymbol{x}^i, \boldsymbol{w}, \boldsymbol{\theta})\right\}$$

is then combined with the L_1-norm approximation of the function

$$e_n(\boldsymbol{u}) = \sum_{i=1}^{n}\sum_{j=1}^{D-1} [\![u_{i,j} \neq \frac{1}{2}(u_{i,j-1} + u_{i,j+1})]\!]$$

the value of which equals to the total number of weights that can be expressed as the average of its neighbors. Analogically to the general formulation (14) we obtain the following convex program

$$\boldsymbol{u}^* = \operatorname*{argmin}_{\boldsymbol{u} \in \mathbb{R}^{n(D+1)}} \left[F_{\text{pwl}}^{\text{svm}}(\boldsymbol{u}, \boldsymbol{\nu}; \lambda) + \gamma \sum_{i=1}^{n}\sum_{j=1}^{D-1} \left| u_{i,j} - \frac{1}{2}u_{i,j-1} - \frac{1}{2}u_{i,j+1} \right| \right]. \quad (25)$$

The constant $\lambda \in \mathbb{R}_+$ controls the empirical error like in the standard SVM. The constant $\gamma \in \mathbb{R}_+$ controls the number of emerging bins, and thus the complexity (or smoothness) of the decision function. The uncompressed parameters $\boldsymbol{u}^*$ cane b converted to the compressed ones $(\boldsymbol{w}^*, \boldsymbol{\theta}^*)$ by the projection methods from Section 4 which are applied to each pair $(\boldsymbol{u}_i^*, \boldsymbol{\nu}_i)$, $i \in \{1, \ldots, n\}$, separately.

6 Experiments

This section provides empirical evaluation of the algorithms proposed in Section 5. Section 6.1 describes learning of malware detector representing the network communication by PWC histograms. Section 6.2 evaluates the proposed

PWL density estimator on synthetic data. Section 6.3 evaluates the proposed algorithm for learning PWL data embedding on classification benchmarks selected from the UCI repository.

6.1 Malware detection by classification of histograms

The proposed approach was applied in the network security domain to classify unseen malware samples from HTTP traffic. The data was obtained from several months (January - July 2015) of real network traffic of companies of various sizes in form of proxy log records. The logs contain anonymized HTTP/HTTPS web requests, where one request represents one connection defined as a group of packets from a single user and source port with a single server IP address, port, and protocol. We grouped all connections into bags, where one bag contains all connections of the same user going the same domain. Finally, we computed a histogram representation of each bag and used the histograms as input to a linear two-class classifier (20) as described in Bartos and Sofka (2015). We compare two methods learning the classifier parameters:

1. The linear SVM using histograms with equidistantly spaced bins. The number of bins per feature varied from $\{8, 12, \ldots, 256\}$.

2. The proposed algorithm learning non-equidistant bins from examples. The uncompressed weights $\boldsymbol{u}^*$ are obtained by solving (21) with the initial discretisation $\boldsymbol{\nu}$ set to split each feature equidistantly to $D = 256$ bins. The constant γ controlling the number of bins varied from 10^{-1} to 10^{-6}. The compressed weights $(\boldsymbol{w}^*, \boldsymbol{\theta}^*)$ were obtained by the rounding procedure (15). Finally, the linear SVM was re-trained on the learned bins $\boldsymbol{\theta}^*$.

The optimal value of the SVM constant γ used by both compared methods was selected from $\{10^{-1}, \ldots, 10^{-5}\}$ based on the validation error.

The data consists of 7,028 positive (malware) and 44,338 negative (legitimate) samples. The positive samples are of 32 classes representing 32 different malware types. The samples were split into training, validation and testing set in ratio 3/1/1. It is ensured that the same malware class never appears simultaneously in the training, validation and test part. We report the average performance and its standard deviation computed over the five test splits.

We analyzed the effect of the number of bins on the precision and recall of the linear SVM classifier. Figure 3 shows precision recall curves (PRCs) for the representation with different number of equidistant bins. Figure 4 shows PRCs of the representation with different number of non-equidistant bins positions of which are learned by the proposed method. It is seen that the representation with non-equidistant bins achieves higher precision using substantially smaller number of bins. In Figure 6 we show the recall at the precision 95% (a common operating point of the detector used in real-life deployment) as the function of the number of bins in order to emphasize the difference between the equidistant and learned non-equidistant representation. The figures are numerically summarized in Table 1.

Figure 5 illustrates the weights $\mathbf{w}$ of the linear SVM classifier trained with three different representations. First we used a histograms with 256 equidistant bins (black line), resulting in a large complexity of weights. Second, we learned

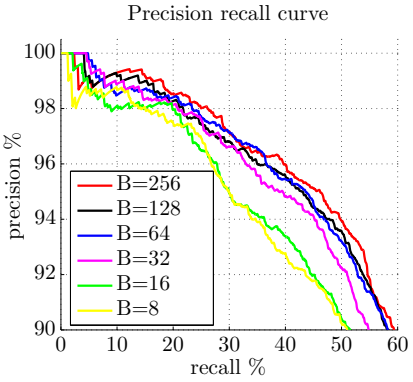


Figure 3: Precision recall curves for histogram representation with different number of equidistant bins. The performance increases with the number of bins, however higher number of bins show comparable results.

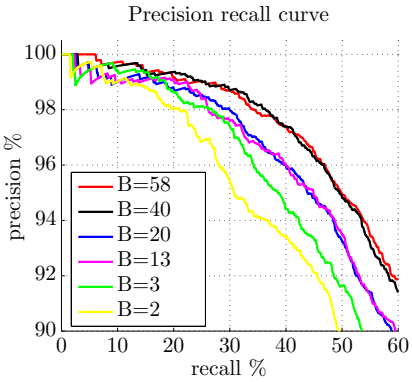


Figure 4: Precision recall curves for histogram representation with non-equidistant bins learned by the proposed method. The performance is increased when compared to Figure 3, and with smaller number of bins.

the weights and bins simultaneously according to the proposed algorithm (red line), which significantly decreased the number of bins without sacrificing the efficacy. Finally, we applied the second step to define new bins by rounding the uncompressed solution and then re-trained a new linear SVM classifier on the top of it (blue line). It means that the discretization for the second and third method is the same, but the third method retrained the classifier and boosted the weights to further maximize the separability of the positive and negative samples.

6.2 Non-parametric distribution estimation

We evaluated the algorithm finding PWL approximation of an unknown distribution described in Section 5.2. The samples were drawn from a mixture of two Gaussians: $p(x) = 0.4 \cdot \mathcal{N}(x; -2, 1) + 0.6 \cdot \mathcal{N}(x; 2, 0.5)$ with mean and standard deviation $(-2, 1)$ and $(2, 0.5)$, respectively. We compared three methods:

1. The proposed algorithm estimating non-equdistant PWL histogram. The uncompressed weights $\boldsymbol{u}^*$ are obtained by solving (23). The initial $D = 100$ bins $\boldsymbol{\nu}$ were placed equidistantly between the minimal and the maximal value in the training set. The optimal value of the constant γ was selected from $\{10, \ldots, 10000\}$ based on the log-likelihood evaluated on the validation set. The compressed parameters $(\boldsymbol{w}^*, \boldsymbol{\theta}^*)$ of the PWL histogram (22) are computed from $\boldsymbol{u}^*$ by the rounding procudere (19) with the precision parameter $\varepsilon = 0.001$.

2. The PWL histogram with bin edges $\boldsymbol{\theta}$ placed equidistantly between the minimal and the maximal value in the training set. The weights $\boldsymbol{w}$ are found by maximizing the likelihood function which is equivalent to solving (23) with $\gamma = 0$. The optimal number of bins was selected from

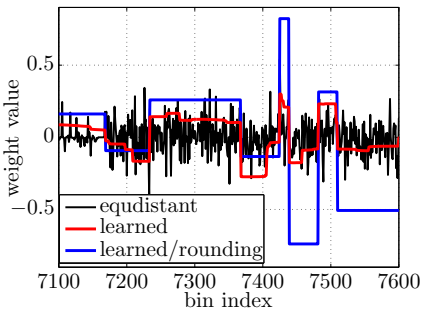


Figure 5: SVM weights learned with three different algorithms: a) linear SVM with 512 equidistant bins, b) proposed simultaneous learning of weights and bins, c) same as b) to define new bins (rounding) that are used for learning new linear SVM classifier.

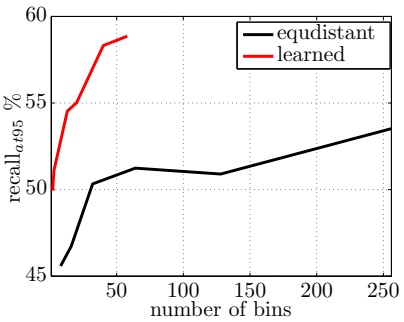


Figure 6: Recall is influenced by the complexity of the baseline histogram representation as well as of the representation learned with the proposed approach. However, the proposed optimization achieves higher recall with significantly less number of bins.

$\{5, 10, \ldots, 100\}$ based on the log-likelihood evaluated on the validation set.

3. The standard PWC histogram with equidistant bins whose number was selected from $\{5, 10, \ldots, 100\}$ based on the log-likelihood evaluated on the validation set. The bin heights were found by the ML method.

We used the distribution $p(x)$ to generate training and validation set the size of which varied from 100 to 10000. For each method we recorded the optimal number of bins and the KL-divergence between the estimated and the ground distribution $p(x)$. The results are averages and standard deviations computed over ten generated data sets.

Figure 7(a) shows the KL-divergence and Figure 7(b) the number of bins as a function of the training set size. As expected the equidistant PWC histogram provides the least precise (high KL divergence) and the most complex (high number of bins) model. We also see that PWL model with equidistantly spaced bins provides the same accurate model as the model with non-equidistant bins learned from example, however, the compactness (the number of bins) of the non-equidistant model is consistently smaller.

Figure 8 shows examples of PWC historams and PWL histograms with learned non-equdistant bins. It is seen that the non-equdistant PWL histogram can closely approximate the ground truth model even from small training sets.

6.3 PWL embedding for non-linear classification

In this section we evaluate the algorithm for learning the PWL data embedding proposed in Section 5.3. We learned a two-class classifier $h(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\theta}) = \text{sign}(f_{\text{pwl}}(\boldsymbol{x}; \boldsymbol{w}; \boldsymbol{\theta}))$ with the discriminant function $f_{\text{pwl}}(\boldsymbol{x}; \boldsymbol{w}; \boldsymbol{\theta})$ defined by (24) for a set of classification problems selected from the UCI repository (Lichman, 2013) which are summarized in Table 2. We evaluated three methods:

Equidistant bins			Learned soft bins		Learned rounded bins		
bins per feature	B	recall at 95 [%]	γ	recall at 95 [%]	bins per feature	B	recall at 95 [%]
256	58,880	53.5 (25.4)	$5 \cdot 10^7$	58.2 (24.4)	58	13,316	58.9 (23.6)
128	29,440	51.0 (27.9)	$1 \cdot 10^6$	56.4 (23.9)	40	9,196	58.3 (22.9)
64	14,720	51.2 (26.5)	$5 \cdot 10^6$	56.4 (22.6)	20	4,634	55.0 (20.6)
32	7,360	50.3 (26.3)	$1 \cdot 10^5$	56.2 (24.3)	13	2,991	54.5 (22.2)
16	3,680	46,7 (26.9)	$5 \cdot 10^5$	54.6 (25.4)	3	741	51.1 (25.7)
8	1,840	45.6 (28.5)	$1 \cdot 10^4$	51.2 (22.5)	2	510	50.0 (27.5)

Table 1: Performance comparison of a linear SVM classifier trained from a histogram representation with equidistant bins with the two proposed methods: learned soft bins (when the bins and SVM weights are learned simultaneously) and learned rounded bins (retraining new SVM weights once the bins are learned from the samples). All approaches have comparable recall, but the proposed algorithms significantly reduced the number of bins.

1. The proposed algorithm learning simultaneously $\boldsymbol{\theta}$ and $\boldsymbol{w}$. The uncompressed parameters $\boldsymbol{u}^*$ are found by solving (25) with the initial discretization $\boldsymbol{\nu}$ equidistantly splitting each feature to $D = 100$ bins. The compressed parameters $(\boldsymbol{w}^*, \boldsymbol{\theta}^*)$ are computed from $(\boldsymbol{u}^*, \boldsymbol{\nu})$ by the rounding procedure (19) with the precision parameter $\varepsilon = 0.1$. Finally, a linear SVM is re-trained on the learned bins $\boldsymbol{\theta}^*$. The constant γ controlling the number of bins is varied from 0.1 to 0.0001.

2. The parameters $\boldsymbol{w}$ are trained by the linear SVM on top of equdistantly constructed bins $\boldsymbol{\theta}$. The number of bins per feature varied from 5 to 20.

3. Method of (Pele et al., 2013) which was shown to outperform the non-linear SVM with many state-of-the-art kernels and data embeddings. The non-equidistant bins are found for each feature independently by constructing edges as the mid-points between the cluster centers obtained from the k-means algorithm. The number of bins is varied from 3 to 20.

The optimal value of the SVM-constant λ used by all three methods is selected from $\{0.1, \ldots, 0.00001\}$ based on the validation error.

We used the same evaluation protocol as in (Pele et al., 2013). Each data set was ten times randomly split into training, validation and test part in the ratio 60/20/20. The reported results are averages and the standard deviations computed on the test part over the ten splits.

Figure 9 shows the test classification accuracy of the compared methods as a function of the number of bins. The baseline PWL embedding with equidistant bins provides slightly but consistently worse accuracy compared to the other two methods learning the non-equidistant bins. The proposed method and the approach of (Pele et al., 2013) yield statistically similar results both in terms of the classification accuracy and the complexity of the embedding (number of bins). However, the proposed method relies solely on solving convex optimization problems unlike the method of (Pele et al., 2013) which involves highly non-convex clustering problem.

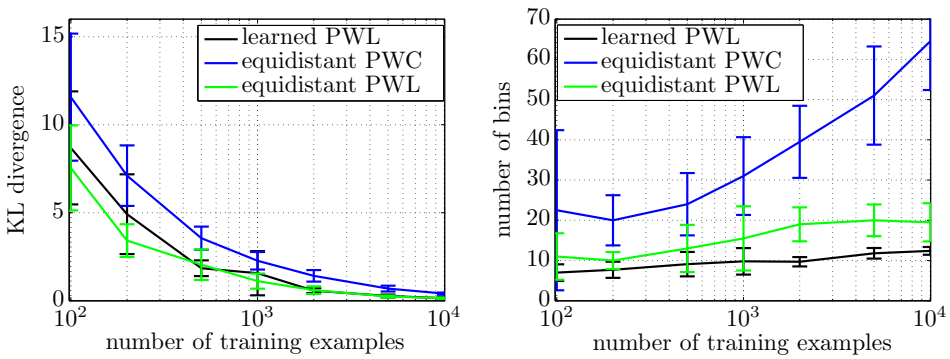


Figure 7: Figures compare the proposed method learning non-equdistant PWL histograms (black), the equdistant PWL histogram (green) and the equdistant PWC histogram (blue). Figure (a) depicts the KL-divergence between the ground truth and the estimated model as a function of the training set size. Figure (b) shows the optimal number of bins as the function of training set size.

name	number of examples	number of features
eyestate	14,980	15
magic	19,020	11
miniboo	130,065	50
musk	6,598	167
skin	245,057	4
wilt	4,889	6

Table 2: A summary of two-class classification problems selected from the UCI repository (Lichman, 2013) and used to evaluate the linear embedding algorithms.

7 Conclusions

We proposed a generic framework which allows to modify a wide class of convex learning algorithms such that they can learn parameters of the piece-wise constant (PWC) and the piece-wise linear (PWL) functions from examples. The learning objective of the original algorithm is augmented by a convex term which enforces compact bins to emerge from an initial fine discretization. In contrast to existing methods, the proposed approach learns the discretization and the parameters of the decision function simultaneously. In addition, learning is converted to a convex problem which is solvable efficiently by global methods. We instantiated the proposed framework for three problems, namely, learning PWC histogram representation of sequential data, estimation of the PWL probability density function and learning PWL data embedding. The proposed algorithms were evaluated on synthetic and standard public benchmarks and applied to malware detection in network traffic data. It was demonstrated that the proposed convex algorithms yield models with fewer number of parameters with

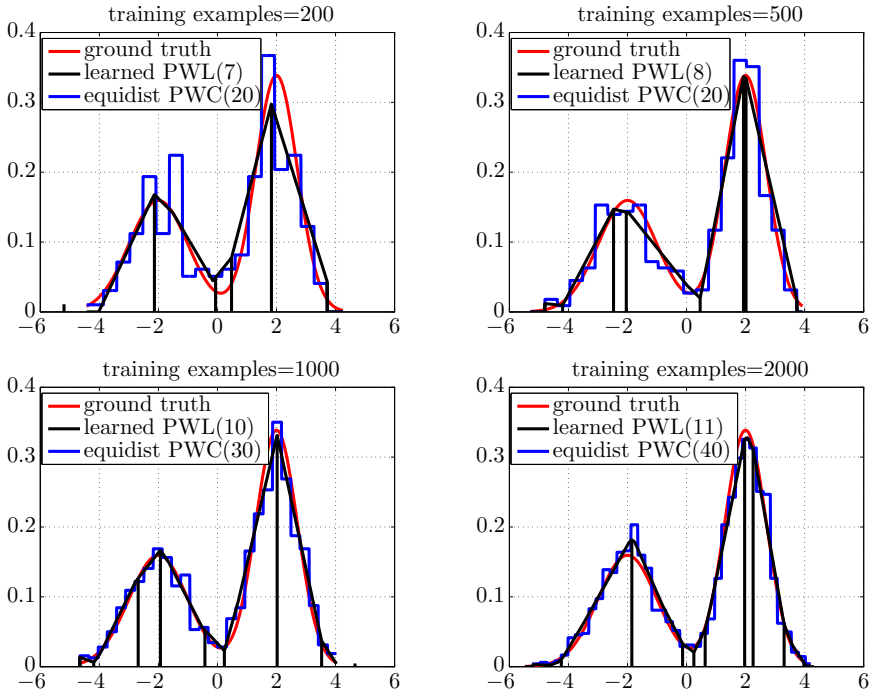


Figure 8: Figures show the ground truth p.d.f. (red), the learned PWL histogram (black) and the PWC histogram with equidistant bins (blue) estimated from training sets of different sizes. The number of bins of a particular histogram is shown in brackets. The black vertical lines denote the learned bin edges.

comparable or better accuracy than the existing methods.

References

Bartos, K. and Sofka, M. (2015). Robust representation for domain adaptation in network security. In *In proc. of ECML/PKDD*, volume 3, pages 116–132.

Boullé, M. (2006). Modl: A bayes optimal discretization method for continuous attributes. *Machine Learning*, 65(1):131–165.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Candes, E., Romberg, J., and Tao, T. (2006). Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8).

Candes, E. and T.Tao (2005). Decoding by linear programming. *IEEE Transactions on Infromation Theory*, 51(12):4203–4215.

Dalal, N. and Triggs, B. (2005). Histogram of oriented gradients for human detection. In *Proc. of Computer Vision and Pattern Recognition*, volume 1, pages 886–893.

Donoho, D. (2006). Compressed sensing. *IEEE Transactions on Infromation Theory*, 52(4):1289–1306.

Dougherty, J., Kohavi, R., and Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In *Proc. of International Conference on Machine Learning*, pages 194–202. Morgan Kaufmann.

Elomaa, T. and Rousu, J. (1999). General and efficient multisplitting of numerical attributes. *Machine Learning*, 36(3):201–244.

Fayyad, U. M. and Irani, K. B. (1992). On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8(1):87–102.

Fayyad, U. M. and Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. of International Joint Conference on Artificial Intelligence*, pages 1022–1029.

Friedman, N. and Goldszmidt, M. (1996). Discretizing continuous attributes while learning bayesian networks. In *Proc. of Internatinal Conference on Machine Learning*, pages 157–165.

García, S., Luengo, J., Sáez, J. A., López, V., and Herrera, F. (2013). A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):734–750.

Hue, C. and Boullé, M. (2007). A new probabilistic approach in rank regression with optimal bayesian partitioning. *Journal of Machine Learning Research*, 8:2727–2754.

Kerber, R. (1992). Chimerge: Discretization of numeric attributes. In *Proc. of the Tenth National Conference on Artificial Intelligence*, AAAI'92, pages 123–128.

Lichman, M. (2013). UCI machine learning repository.

Liu, H., Hussain, F., Tan, C. L., and Dash, M. (2002). Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, 6(4):393–423.

Pele, O., Taskar, B., Globerson, A., and Werman, M. (2013). The pairwise piecewise-linear embedding for efficient non-linear classification. In *Proc. of the International Conference on Machine Learning*, pages 205–213.

Yang, Y. and Webb, G. I. (2008). Discretization for naive-bayes learning: managingdiscretization bias and variance. *Machine Learning*, 74(1):39–74.

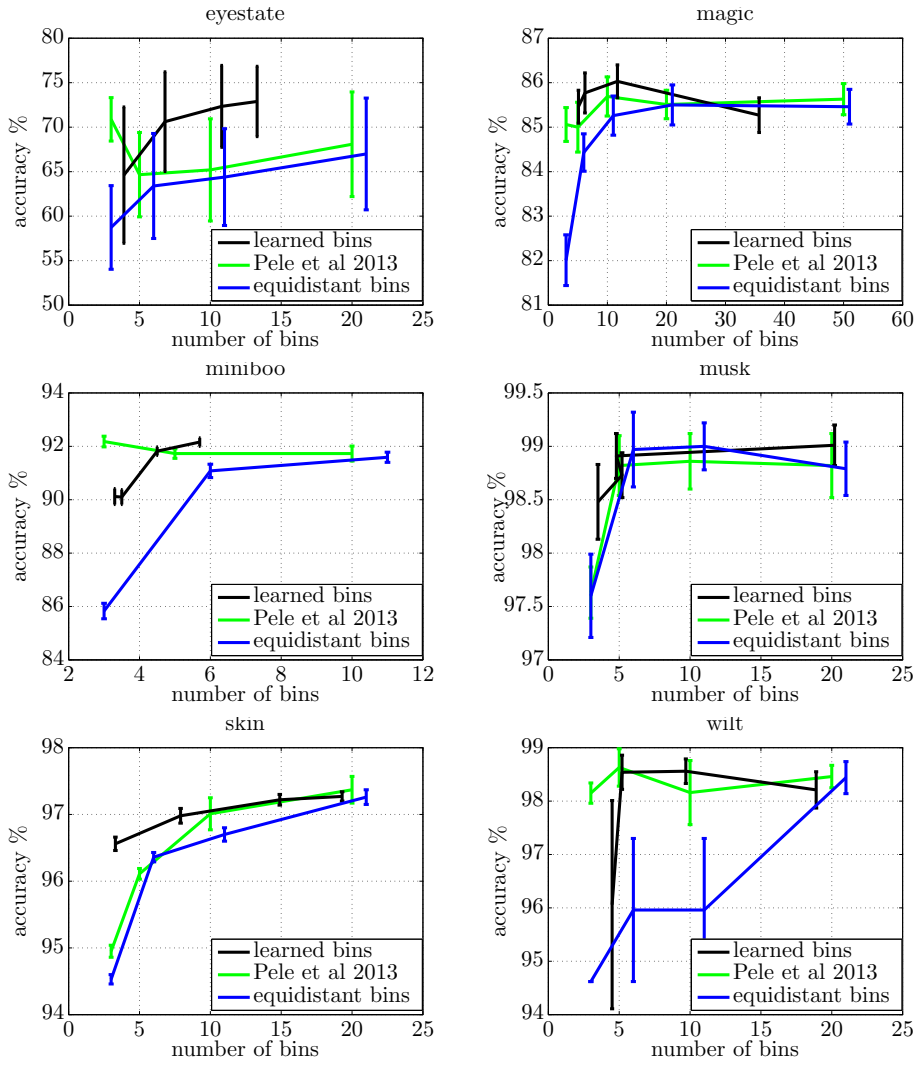


Figure 9: Figures show the classification accuracy (mean and std computed over ten splits) of the linear classifier using PWL data embedding evaluated on six two-class classification problems selected from UCI repository. The accuracy is shown as a function of the average number of bins used to discretize each feature. The results are shown for three compared methods. The baseline using equidistantly placed bins (blue), the method of (Pele et al., 2013) finding non-equidistant bins by the k-means algorithm (green) and the proposed methods learning the non-equidistant bins by a convex programming (black).