Interpretable Preference Learning: A Game Theoretic Framework for Large Margin On-Line Feature and Rule Learning

Mirko Polato,[1] **Fabio Aiolli**[1]
[1]Department of Mathematics, University of Padova, Italy
mpolato@math.unipd.it, aiolli@math.unipd.it

Abstract

A large body of research is currently investigating on the connection between machine learning and game theory. In this work, game theory notions are injected into a preference learning framework. Specifically, a preference learning problem is seen as a two-players zero-sum game. An algorithm is proposed to incrementally include new useful features into the hypothesis. This can be particularly important when dealing with a very large number of potential features like, for instance, in relational learning and rule extraction. A game theoretical analysis is used to demonstrate the convergence of the algorithm. Furthermore, leveraging on the natural analogy between features and rules, the resulting models can be easily interpreted by humans. An extensive set of experiments on classification tasks shows the effectiveness of the proposed method in terms of interpretability and feature selection quality, with accuracy at the state-of-the-art.

Introduction

Nowadays, the connection between game theory (GT) and machine learning (ML) is heavily studied in the field of ML for security in which security challenges are faced using integration frameworks between GT and ML (Yufei Liu 2017; Xu et al. 2017). Many of these approaches can also be considered part of the more general literature concering Adversarial Learning (Goodfellow et al. 2014) that is now a hot topic in ML. Less recently, the GT-ML duo has been also investigated in the mainstream ML literature. For instance, it is well known that hard margin SVM can be cast into a two-players zero-sum game (Aiolli, Da San Martino, and Sperduti 2008). Another famous example is the seminal work that introduced Adaboost (Freund and Schapire 1997), in which GT is applied to on-line learning. Many others ML concepts have been also related to GT, e.g., as showed by Ionnadis et al. (Ioannidis and Loiseau 2013), linear regression problems can be seen as a non-cooperative game, while in (Rezek et al. 2008) authors elucidate the equivalence between inference in ML and GT.

In this paper we present a principled algorithm inspired by GT and preference learning (Fürnkranz and Hüllermeier 2010) for classification. The learning problem is seen as a two-players zero-sum game, in which the considered hypotheses spaces consist in a set of preference prototypes along with (possibly non-linear) features. Moreover, we show how feature selection naturally comes as a side effect of the algorithm. One of the biggest challenges in feature selection is dealing with large scale data in particular with (infinitely) many input features. This is a typical scenario in real-world applications when data instances are of high dimensionality or it is expensive/inconvenient to acquire all attributes. In these contexts, batch approaches are simply not applicable for computational reasons. Thus, there is the need to move towards online feature selection (OFS) approaches (Wang et al. 2014; Wu et al. 2013), which are allowed to work with a small and limited number of features. Following this direction, we discuss how the on-line feature generation plays a key role in the algorithm, especially when it comes to interpret the solution of the model, which is very useful for producing explanations. Differently from post-hoc methods which explain black box models (Guidotti et al. 2018; Polato and Aiolli in press 2019), our model is intrinsically explainable when features are easy to interpret. There are plenty of applications in which explanation plays a key role, such as bioinformatic applications, recommender systems, and support systems for physicians, just to mention a few. The notion of explanation is also one of the most controversial subject of the recently introduced GDPR (General Data Protection Regulation).

To summarize, the main contributions of this work are the following:

- a new large margin preference learning method based on game theoretical concepts that naturally offers feature and rule selection capability. The proposed framework is general enough to deal with different kinds of features and rules that are very useful when interpretability is desired;
- an efficient approximated algorithm for large scale game matrices with potentially infinite number of columns, allowing the application of the method to online scenarios (OFS). Empirical evidence shows that the number of selected columns by the algorithm is typically orders of magnitude less than the total number of available columns;
- an extensive set of experiments is reported to assess both the effectiveness and the quality of the feature/rule selec-

tion of our method. Results demonstrate that the algorithm is able to provide sparse solutions that are suitable when the explanation of the decision is required/desirable.

The reminder of this paper is structured as follows: next section will introduce all the background knowledge needed to fully understand the notions discussed afterwards. Then, the main contributions of this paper will be described. Finally, all the performed experiments are reported and thoroughly discussed. For more details about the experiments and the datasets please refer to the supplementary material.

Background

Preference Learning

Preference learning (PL) is a machine learning subfield in which the goal is to learn a preference model from observed item preferences. The constructed model has to predict preferences for previously unseen items. Label ranking is one of the three main PL tasks (Fürnkranz and Hüllermeier 2010): given a set of input patterns $\mathbf{x}_i \in \mathcal{X}$, $i \in [1, \ldots, n]$, and a finite set of labels $\mathcal{Y} \equiv \{y_1, y_2, \ldots, y_m\}$ the goal is to learn a scoring function $g_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ which assigns a score for each instance-label pair $(\mathbf{x}, y)$. Label ranking represents a generalization of a classification task, since g_θ implicitly defines, for an instance $\mathbf{x}$, a total order over $\mathcal{Y}$. In the context of PL, the training set used to build the model consists of a set of pairwise preferences $y_i \succ_{\mathbf{x}} y_j$, $i \neq j$, i.e., for the pattern $\mathbf{x}$, y_i is preferred to y_j. In the case of classification, in which a pattern $\mathbf{x}$ is associated to a unique label y_i, the set of preferences is $\{y_i \succ_{\mathbf{x}} y_j \mid 1 \leq j \neq i \leq m\}$.

In this work we focus on linear preference models (Tsochantaridis et al. 2004; Aiolli and Sperduti 2004; 2010) of the form $g_\theta(\mathbf{x}, y) = \mathbf{w}^\intercal \psi(\mathbf{x}, y)$, where $\theta \equiv \mathbf{w}$ is the parameters vector and $\psi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^{d \cdot m}$, $\mathcal{X} \equiv \mathbb{R}^d$, $\mathcal{Y} \equiv \{1, \ldots, m\}$ is a joint representation of instance-label pairs. In order to properly rank the labels w.r.t. each item, given a preference $y_i \succ_{\mathbf{x}} y_j$ then $g_\theta(\mathbf{x}, y_i) > g_\theta(\mathbf{x}, y_j)$ should hold, and thus

$$\mathbf{w}^\intercal \psi(\mathbf{x}, y_i) > \mathbf{w}^\intercal \psi(\mathbf{x}, y_j) \Rightarrow \mathbf{w}^\intercal (\psi(\mathbf{x}, y_i) - \psi(\mathbf{x}, y_j)) > 0,$$

which can be interpreted as the margin (a.k.a. *confidence*) of the preference. Intuitively, large margins correspond to good generalization capability of the ranker (Schapire et al. 1997). We assume an instance-label joint representation defined as $\psi(\mathbf{x}, y) = \mathbf{x} \otimes \mathbf{e}_y^m$, where the symbol $\otimes$ indicates the Kronecker product and $\mathbf{e}_y^m$ is the *y-th* vector of the canonical basis of $\mathbb{R}^m$. Thus, given a preference $y_i \succ_{\mathbf{x}} y_j$ we construct its corresponding representation by $\mathbf{z} = \psi(\mathbf{x}, y_i) - \psi(\mathbf{x}, y_j) = \mathbf{x} \otimes (\mathbf{e}_{y_i}^m - \mathbf{e}_{y_j}^m)$, $\mathbf{z} \in \mathbb{R}^{d \cdot m}$. The f-th m-dimensional chunk of a preference $\mathbf{z}$ is indicated by

$$\mathbf{z}[f] = (z_{(f-1)\cdot m}, z_{(f-1)\cdot m+1}, \ldots, z_{f \cdot m-1}) \in \mathbb{R}^m.$$

At classification time, given a new pattern $\mathbf{x}$, the predicted label $\hat{y}$ is computed by selecting the label that maximizes the score $g_\theta(\mathbf{x}, y)$, that is, $\hat{y} = \arg\max_{y \in \mathcal{Y}} g_\theta(\mathbf{x}, y)$.

Game Theory

Game theory is a branch of mathematics that studies the strategic interaction between rational decision-makers. For the purposes of this paper, we focus on two-players zero-sum games, which are by definition non-cooperative games. The strategic form of a two-players zero-sum game is defined by a matrix $\mathbf{M}$, dubbed payoff matrix or game matrix. The game takes place in rounds in which the two-players, the row player P and the column player Q, play simultaneously: the row player picks a row and the column player picks a column of $\mathbf{M} \in \mathbb{R}^{P \times Q}$, where P and Q are the number of available strategies for P and Q respectively. Each matrix entry $\mathbf{M}_{i,j}$ represents the loss of P, or equivalently the payoff of Q, when the strategies i and j are played by the two-players. The goal of the player P is to define a strategy that minimizes its expected loss V. Conversely, the player Q aims at finding a strategy that maximizes its payoff. Typically, the players strategies are randomized over the rows/columns of the game matrix: player P selects a row according to a probability distribution $\mathbf{p}$ over the rows, and, similarly, player Q selects a column according to a probability distribution $\mathbf{q}$ over the columns. These strategies are usually referred to as mixed strategies. For presentation purposes we refer to the vectors $\mathbf{p}$ and $\mathbf{q}$ as stochastic vectors, that is $\mathbf{p} \in \mathscr{S}_P$ and $\mathbf{q} \in \mathscr{S}_Q$, where $\mathscr{S}_P = \{\mathbf{p} \in \mathbb{R}_+^P \mid \|\mathbf{p}\|_1 = 1\}$ and $\mathscr{S}_Q = \{\mathbf{q} \in \mathbb{R}_+^Q \mid \|\mathbf{q}\|_1 = 1\}$. The optimal pair of strategies $(\mathbf{p}^*, \mathbf{q}^*)$, i.e., the saddle-point (a.k.a. Nash equilibrium) of $\mathbf{M}$, has a well-know formulation (von Neumann 1928), that is

$$V^* = \mathbf{p}^{*\intercal} \mathbf{M} \mathbf{q}^* = \min_{\mathbf{p}} \max_{\mathbf{q}} \mathbf{p}^\intercal \mathbf{M} \mathbf{q} = \max_{\mathbf{q}} \min_{\mathbf{p}} \mathbf{p}^\intercal \mathbf{M} \mathbf{q},$$

where V^* is the value of the game. It is well known that the saddle-point solution of the equation above can be found using linear programming with a number of variables proportional to the number of (pure) strategies. It is evident that for high dimensional game matrices the computational complexity can become prohibitive. A way to overcome this computational issue is to rely on approximated solutions. An adaptive approach to compute an approximate saddle-point strategy using multiplicative weights has been proposed by Freund et al. (Freund and Schapire 1999; 1996). This algorithm, called *Adaptive multiplicative weights* (AMW), is guaranteed to come close to the minimum loss achievable by any fixed strategy. More recently, an incremental version of AMW (i-AMW) has also been proposed in (Bopardikar and Langbort 2014). A randomized approach is presented in (Bopardikar et al. 2010), where each player chooses its best mixed strategy on a sampled set of rows/columns, that is, the game matrix is a submatrix of the whole payoff matrix. Authors prove that when the submatrix is sufficiently large the achieved approximation is good with high probability.

The *fictitious play* algorithm (Brown 1951) (a.k.a. Brown-Robinson learning process) is one of the first methods for computing approximate saddle-point strategies. Fictitious play starts with a random initial pure strategy for the player P. Then, in turn, each player picks its next pure strategy as the best response, assuming the opponent picks uniformly at random one among its previous choices. In other words, at each round both players try to infer the opponent mixed strategy from its previous picks. The pseudo-code of Fict-Play adapted to our purposes is reported in Alg. 1.

Algorithm 1: FictPlay: Fictitious Play algorithm

Input: $\mathbf{M} \in \mathbb{R}^{P \times Q}$: matrix game, T_e: number of iterations
Output: $\mathbf{p}, \mathbf{q}$: row/column player strategy, V: the value of the game

$r \leftarrow randint[1, P]$
$\mathbf{s}_p, \mathbf{v}_p \leftarrow \mathbf{0}, \mathbf{0}$
$\mathbf{s}_q, \mathbf{v}_q \leftarrow \mathbf{M}_{r,:}, \mathbf{e}_r^P$
for $t \leftarrow 1$ **to** T_e **do**
  $\hat{q} \leftarrow \arg\max \mathbf{s}_q, \quad \mathbf{s}_p \leftarrow \mathbf{s}_p + \mathbf{M}_{:,\hat{q}}$
  $\hat{p} \leftarrow \arg\min \mathbf{s}_p, \quad \mathbf{s}_q \leftarrow \mathbf{s}_q + \mathbf{M}_{\hat{p},:}$
  $\mathbf{v}_q \leftarrow \mathbf{v}_q + \mathbf{e}_{\hat{q}}^Q, \quad \mathbf{v}_p \leftarrow \mathbf{v}_p + \mathbf{e}_{\hat{p}}^P$
end
$\mathbf{p} \leftarrow \mathbf{v}_p / \|\mathbf{v}_p\|_1, \mathbf{q} \leftarrow \mathbf{v}_q / \|\mathbf{v}_q\|_1, V \leftarrow \mathbf{p}^\intercal \mathbf{M} \mathbf{q}$
return $\mathbf{p}, \mathbf{q}, V$

In the algorithm, $\mathbf{M}_{r,:}$ and $\mathbf{M}_{:,c}$ indicate the r-th row and the c-th column of the matrix $\mathbf{M}$, respectively.

A game theoretic perspective of PL

In this section we introduce the main theoretical contribution of the paper. Specifically, we propose a new learning approach for label ranking based on game theory. We assume to have a set of training preferences of the form $(y_+ \succ_{\mathbf{x}} y_-)$ which can be easily transformed to their corresponding vectorial representation as previously described. We consider an hypothesis space $\mathcal{H}$ composed by linear functions, i.e., $\mathcal{H} \equiv \{\mathbf{z} \mapsto \mathbf{w}^\intercal \mathbf{z} \mid \mathbf{w}, \mathbf{z} \in \mathbb{R}^{d \cdot m}\}$.We say that a preference $\mathbf{z}$ is satisfied w.r.t. an hypothesis $\mathbf{w}$ iff $\mathbf{w}^\intercal \mathbf{z} > 0$, that is, whether the margin of the preference $\rho(\mathbf{z}) = \mathbf{w}^\intercal \mathbf{z}$ is strictly positive. Such margin represents the confidence of the hypothesis $\mathbf{w}$ over the preference $\mathbf{z}$. According to the maximum margin principle, we aim to select $\mathbf{w}$ such that it maximizes the minimum margin over the training preferences. Thanks to the Representer Theorem (Kimeldorf and Wahba 1971; Hofmann, Scholkopf, and Smola 2008) we know that the optimal $\mathbf{w}$ can be defined as a convex combination of the training preferences, that is $\mathbf{w} \propto \sum_j \alpha_j \mathbf{z}_j, \boldsymbol{\alpha} \in \mathscr{S}_P$. Thus, we can rewrite the margin of a preference $\mathbf{z}$ as

$$\rho(\mathbf{z}) = \sum_j \alpha_j \mathbf{z}_j^\intercal \mathbf{z} \doteq \sum_j \alpha_j \sum_f \mu_f \mathbf{z}_j[f]^\intercal \mathbf{z}[f]$$
$$= \sum_j \sum_f \alpha_j \mu_f \mathbf{z}_j[f]^\intercal \mathbf{z}[f] = \sum_{(j,f)} q_{(j,f)} \mathbf{z}_j[f]^\intercal \mathbf{z}[f],$$

where the dot-product $\mathbf{z}_j^\intercal \mathbf{z}$ is generalized by giving different weights to the features according to a distribution $\boldsymbol{\mu}$ over the features, and $\mathbf{q}$ such that $q_{(j,f)} = \alpha_j \mu_f$ is a new distribution over all the possible preference-feature pairs.

Now, let us call $\mathbf{p}$ the distribution over the training preferences, then the expected preference margin w.r.t. the distribution $\mathbf{p}$ is defined by

$$\bar{\rho}(\mathbf{p}, \mathbf{q}) = \sum_i p_i \sum_{(j,f)} q_{(j,f)} \mathbf{z}_j[f]^\intercal \mathbf{z}_i[f] = \mathbf{p}^\intercal \mathbf{M} \mathbf{q}, \quad (1)$$

where $\mathbf{M}_{i,(j,f)} = \mathbf{z}_i[f]^\intercal \mathbf{z}_j[f]$.

If we take a closer look at Eq. (1), we can easily grasp the strong relation between such a preference learning problem and the game theory setting. In particular, let us consider a two-players zero-sum game where the row player P (the nature) picks a distribution over the whole set of training preferences (i.e., rows) aiming at minimizing the expected margin. Simultaneously, the opponent player Q (the learner) picks a distribution over the set of preference-feature pairs (i.e., columns) aiming at maximizing the expected margin (payoff). So, the value of the game, that is the maximal minimum margin solution is given by

$$V = \bar{\rho}(\mathbf{p}^*, \mathbf{q}^*) = \min_{\mathbf{p}} \max_{\mathbf{q}} \mathbf{p}^\intercal \mathbf{M} \mathbf{q}.$$

Thus, it turns out that the distribution $\mathbf{q}$ maximizing the minimum margin in the training set can be found as the saddle-point solution of the game matrix $\mathbf{M}$.

Approximating the optimal strategies

Dimensionally speaking the game matrix $\mathbf{M}$ in Eq. (1) can be huge, in particular, this is the case for its number of columns as it is equal to the number of all possible preference-feature pairs. Thus, solving the game using standard off-the-shelf algorithms from game theory is impractical. In this section we propose a new method to overcome this issue by solving the game incrementally.

The main idea is to iteratively consider only a subset of columns of the whole game matrix, in such a way that, at each iteration, the suboptimal computed solution becomes closer and closer to the optimal one. Formally, let us suppose to have a game matrix $\mathbf{M}$ and let denote with $(\mathbf{p}^*, \mathbf{q}^*, V^*)$ its corresponding optimal solution. At each iteration we consider a subset of columns of $\mathbf{M}$, that is $\mathbf{M}_t = \mathbf{M}\boldsymbol{\Pi}_t$ where $\boldsymbol{\Pi}_t \in \{0,1\}^{Q \times B}$ are left-stochastic (0,1)-matrices, i.e. matrices whose entries belong to the set $\{0,1\}$ and whose columns add up to one. B is the number of columns considered at each iteration, $B \ll P$. Let $(\mathbf{p}_t^*, \mathbf{q}_t^*, V_t^*)$ be the solution for the matrix $\mathbf{M}_t$ computed at iteration t. At the end of each iteration, the columns of $\mathbf{M}_t$ corresponding to null entries in $\mathbf{q}_t^*$ are replaced by new columns randomly drawn from the whole set of columns. We can show that the value of the game at each iteration increases monotonically and it is upper bounded by the optimal margin, that is the value of the game when considering the full matrix $\mathbf{M}$. Specifically, let us assume to be at iteration $t+1$: a new left-stochastic (0,1)-matrix $\boldsymbol{\Pi}_{t+1}$ is considered, which is $\boldsymbol{\Pi}_t$ where every column corresponding to null entries in $\mathbf{q}_t^*$ are substituted with a new random stochastic vector $\mathbf{e}_h^Q$ for a random column h. Thus, it can be shown that

$$V_t^* = \mathbf{p}_t^{*\intercal} \mathbf{M}_t \mathbf{q}_t^* = \mathbf{p}_t^{*\intercal} \mathbf{M} \boldsymbol{\Pi}_t \mathbf{q}_t^* \quad (2)$$
$$\leq \mathbf{p}_{t+1}^{*\intercal} \mathbf{M} \boldsymbol{\Pi}_t \mathbf{q}_t^* \quad (3)$$
$$= \mathbf{p}_{t+1}^{*\intercal} \mathbf{M} \boldsymbol{\Pi}_{t+1} \mathbf{q}_t^* \quad (4)$$
$$\leq \mathbf{p}_{t+1}^{*\intercal} \mathbf{M}_{t+1} \mathbf{q}_{t+1}^* = V_{t+1}^* \quad (5)$$

and

$$\forall t, V_t^* = \mathbf{p}_t^{*\intercal} \mathbf{M} \boldsymbol{\Pi}_t \mathbf{q}_t^* \leq \mathbf{p}^{*\intercal} \mathbf{M} \underbrace{\boldsymbol{\Pi}_t \mathbf{q}_t^*}_{\hat{\mathbf{q}}_t} \leq \mathbf{p}^{*\intercal} \mathbf{M} \mathbf{q}^* = V^*.$$

Equivalence (2) is trivial since $\mathbf{M}_t = \mathbf{M}\mathbf{\Pi}_t$ by definition. Inequality (3) holds because the strategy $\mathbf{p}^*_{t+1}$ is suboptimal for $\mathbf{M}_t$. In (4) we simply replace columns of the game matrix corresponding to null entries of $\mathbf{q}^*_t$ which does not affect the value. Finally, inequality (5) is true because $\mathbf{q}^*_t$ is suboptimal for $\mathbf{M}_{t+1}$, and similar considerations can be done for the last series of inequalities. The pseudo-code of the full algorithm (PRL: Preference and Rule Learning algorithm) is given in Algorithm 2.

Algorithm 2: PRL: Preference and Rule Learning

Input: $\mathcal{P}$: set of training preferences
F_{gen} : random feature generator
B: size of the working set
T: number of epochs
T_e: number of iterations of *FictPlay*
Output: $\mathcal{Q}$: working set of hypothesis
$\mathbf{q}$: mixed strategy in $\mathcal{Q}$

random initialization of the set $\mathcal{Q}$ such that $|\mathcal{Q}| = B$
compute the matrix game $\mathbf{M}$ on the basis of $\mathcal{P}$ (rows) and $\mathcal{Q}$ (cols)
for $t \leftarrow 1$ **to** T **do**
  $\mathbf{p}, \mathbf{q}, v \leftarrow$ *FictPlay*$(\mathbf{M}, T_e)$
  if $t < T$ **then**
    foreach $(j, f) \mid \mathbf{q}_{(j,f)} = 0$ **do**
      $(j', f') \leftarrow$ *pick*$(\mathcal{P}), F_{gen}()$
      update $\mathcal{Q}$: replace (j, f) with (j', f')
      update columns of $\mathbf{M}$ w.r.t. $\mathcal{Q}$:
        let k the position of (j', f') in $\mathcal{Q}$,
        for all $i \in \mathcal{P}$, $\mathbf{M}_{i,k} = \mathbf{z}_i[f]^\intercal \mathbf{z}_{j'}[f]$
    end
  end
end
return $\mathbf{q}$, $\mathcal{Q}$

It is worth to notice that the algorithm does not require any prior knowledge about $\mathbf{M}$, and the number of columns can be also infinite. Thus, a natural approach for dealing with potentially infinite game matrices is to use an online column generation approach.

Online feature/rule generation

One of the most important steps of the PRL is the generation of new columns. Since a game matrix column is defined by a preference-feature pair we need to define how such features are generated. In this work we focus on two features generation schemes: polynomial feature generation, and rule generation. In the algorithm we referred to a generic feature generator scheme with the function F_{gen}.

Polynomial features generation This scheme generates features that are taken from the space of features of a polynomial kernel. In particular, we focus on homogeneous polynomial features of a given degree. For example, given an n-dimensional instance $\mathbf{x}$ some possible polynomial features of degree 3 are: $x_1x_2x_n$, $x_1^2x_3$ and x_n^3. Note that, when the input variables are binary-valued such monomials correspond to logical conjunctions.

Rules generation A rule is a logical condition over an input variable. The introduction of rules can be useful when it comes to interpret the model. Let us make some examples. In the case of binary valued input variables a rule is simply its truth value. When dealing with continuous variables, a rule is a relation involving the values of the variables. For instance, by defining a threshold like $x \geq 5$, or equalities such as $x = 3.2$. Such thresholds can be chosen on the basis of a heuristic or randomly from the set of unique values of the variable. In order to create more complex rules, it is also possible to combine them. In particular, given two rules, their product represents, from a logical stand point, the conjunction of the two conditions. In the remainder we will refer to the arity of this combination as the degree of the rule.

Evaluation

In this section we describe and discuss the set of experiments performed to assess the effectiveness of PRL. Specifically, we performed two different sets of experiments: the first set aims to evaluate the degree of interpretability of PRL, while the second set focuses on the assessment of the performance and the quality of the feature selection. In all the experiments the number of iterations T_e of FictPlay has been set to 10^6, while the number T of epochs of PRL has been set to 10^3. The complete set of experiments as well as a thorough analysis of the obtained results is reported in the supplementary material. The PRL implementation is open source and freely available at https://github.com/makgyver/PRL. In the following sections we present the most relevant and interesting results.

Model interpretation

In the first set of experiments, we employed PRL to select the most relevant features for interpreting the model. The aim is to use these features to explain the decision. We run PRL on four benchmark datasets:

`tic-tac-toe` is a dataset containing 958 ending positions of the game tic-tac-toe, and the task is to classify whether the $\times$ is the winner;

`breast-cancer` is the well known Breast Cancer Wisconsin Diagnostic Dataset, where the task is to classify a tumor as malignant or benign. For more details about the dataset please refer to (Hayashi and Nakano 2015);

`poker` dataset contains 25010 poker hands and the task is to classify the value of the hand, e.g., pair, full house and so on (10 classes). In our experiments, three binary classification tasks have been derived from the original multiclass dataset as described in the following section;

`mnist` is a (well known) dataset of handwritten digits. The task consists in classifying the digits (10 classes).

The datasets have been pre-processed as in the following: `tic-tac-toe` has been converted into a binary-valued dataset through one-hot encoding, obtaining 27 binary input variables for each instance. Both `breast-cancer`

and mnist did not require any pre-processing. For every datasets, instances with missing values have been removed. For the poker dataset we defined a hierarchy of features. The goal was showing that, increasing the expressiveness of the features, PRL is still able to identify the smallest set of features/rules able to explain the classification. The hierarchy can be summarized as follows:

Level 1 The features are a simple enumeration of the cards. An hand is represented with a binary vector of dimension 52 in which an entry is equal to 1 iff the corresponding card is in the hand;

Level 2 Features which are a counting aggregation of the suits and the ranks of the cards in the hand are added. Suits are represented as a four dimensional vector, while ranks as a 13 dimensional vector. Note that the Ace is assumed of rank 1, while J=11, Q=12, and K=13;

Level 3 This level contains 5 additional features that are a further aggregation of the features in the first and second level, namely: number of different ranks, number of different suits, number of cards of the most popular suit, number of cards of the most popular rank, and maximum difference between ranks.

Note that, at each level of the hierarchy the features of the previous level are kept in the representation. For extensive details about the construction of the hierarchy please refer to the supplementary material.

The tic-tac-toe dataset The tic-tac-toe dataset has been used as a toy testbed in which the positive class (i.e., win for $\times$) can be expressed with a single DNF rule. This experiment aims to show that PRL is able to identify such explaining rules. Since we have a-priori knowledge about the game, we trained PRL using polynomial features of degree 3. In the case of binary valued data, polynomial features correspond to conjunctions, and hence they are suited for our purposes. Experiments have been performed using a 70-30% training and test split division. After the training the 10 features with the highest weights were: $x_8x_{17}x_{26}$, $x_2x_{11}x_{20}$, $x_2x_{14}x_{26}$, $x_8x_{14}x_{20}$, $x_{20}x_{23}x_{26}$, $x_{11}x_{14}x_{17}$, $x_2x_5x_8$, $x_5x_{14}x_{23}$, $\tilde{x}^3_{13}$ and $\tilde{x}^3_{25}$. Features with a $\sim$ on top are the ones which characterize a negative preference (no win for $\times$). The main observation about these features is that the first 8 are in fact the available three-in-a-row for the crosses, while the last two features represent a naught in the central and in the bottom right cell. The former is reasonable since it is more likely to win by occupying the central square, the latter is instead not very significant. Overall, the algorithm has been able to identify all the conditions that determine a win for the cross.

The poker dataset Akin to the tic-tac-toe, in the poker dataset the classes can be described by means of logical rules. However, in this case the rules are generally much more complex. As described earlier, a three levels hierarchy of features has been defined, with the aim of testing whether the algorithm is able to recognize the smallest set of useful features to explain the classification. The main idea is that the algorithm should be able to retrieve the easiest rules which explain the classification when more expressive features are also available.

In order to have more control on the tasks, and to highlight the different difficulty in recognizing hand values such as a straight, we defined the following three binary classification tasks: *TOK* (Three Of a Kind) versus rest, *Flush* versus rest, and *Straight* versus rest. Besides the rule extraction test, we also compared the balanced accuracy (BACC) of PRL w.r.t. a standard SVM with polynomial kernel. We employed the BACC because the dataset is highly imbalanced, i.e., positive class $\leq 2\%$. The balanced accuracy is defined as:

$$\text{BACC} = \frac{1}{2}\left(\frac{TP}{P} + \frac{TN}{N}\right) \times 100,$$

where TP stands for true positives (P = positives), and TN for true negatives (N = negatives).

The hyper-parameter C of SVM has been validated in the set $\{10^{-3}, \ldots, 10^4\}$, and the degree in the range [1,3] via 3-fold cross validation. PRL has been trained using rules of degree 1 on the set of relations $\{=\}$. Experiments have been performed using a 80-20% training and test split division.

The achieved results are reported in Table 1.

Method	# Level	TOK	Flush	Straight
SVM	1	**50.00**	50.00	**50.00**
PRL	1	48.08	**59.03**	49.97
SVM	2	50.00	50.00	50.00
PRL	2	**100.00**	**77.75**	**51.29**
SVM	3	99.99	96.43	50.00
PRL	3	**100.00**	**100.00**	**100.00**

Table 1: Balanced accuracy (%) on the poker dataset. The highest accuracies in all classification tasks, and in all levels, are highlighted in **bold**.

The first observation worth to be mentioned is that PRL using the third level of features has been able to identify, for all the tasks, the rules which explain the classification, achieving perfect accuracy. A remarkable difference w.r.t. the SVM can be noticed in the *Straight* classification, which is the hardest task. SVM simply classifies any instance as negative ($TP = 0$), with the same behaviour in all the tasks both at the second and at the first level of the hierarchy. Concerning the third level, SVM had one false negative in the *TOK* task and one false positive in the *Flush*, achieving a BACC of 99.99% and 96.43%, respectively.

Let us now analyze the rules extracted by PRL in each task. We can observe that none of the tasks can be explained with simple rules in the first level of the hierarchy. In fact, at level 1, the algorithm struggles in all the tasks achieving a BACC around 50% like SVM, with the exception of the *Flush* in which it achieves a 59%. In the second level, instead, there are features expressive enough to explain both the *TOK* and the *Flush*, while the *Straight* remains a quite hard task. Specifically, an hand contains a TOK anytime one of the rank has a cardinality $= 3$. However, this rule also includes the full house as a false positive. Similarly, a flush

can be described with a suit of cardinality 5, but this also includes the straight flush and the royal flush as false positives. In both these tasks, PRL found the just mentioned rules. It is interesting to note that the method has been able to avoid the full house mistake by adding rules for the negative class (i.e., not a flush). In particular, these rules state that there is a rank with cardinality $= 2$, which is enough to say that there is no TOK in the hand. With the features contained in the third and final level of the hierarchy it is possible to define all the considered classes:

TOK # of ranks = 3, # of cards of the most popular rank = 3;

Straight # of ranks = 5, max difference between ranks = 4, # of suits $\neq 1$;

Flush # of suits = 1, max difference between ranks $\neq 4$.

At this level, PRL was able to identify all the correct rules achieving a BACC of 100% in all the tasks.

Remark: whenever each class can be defined by simple rules, we showed that PRL is able to identify them all. However, when there are classes which cannot be easily characterized, PRL fails in finding reasonable rules, as for example in the `poker`. In this case, the positive class (e.g., *Flush*) is defined by simple rules, but there are no easy ways to express its opposite. Thus, PRL will return, with the highest weights, the rules that explain the positive class. All the remaining rules (with small weights) will be associated with negative preferences, but with very small coverage and hence with small generalization capabilities. To address this issue, we allow the feature generation mechanism to pick new pairs consisting of rules which are always true. When such feature is selected and associated with a negative preference it will give a bias to the negative class. In this way, when the positive rules are not satisfied the decision of the classifier will be the negative class. This additional feature has been used in all experiments concerning the `poker` dataset.

The `breast-cancer` dataset The Wisconsin Breast Cancer dataset (`breast-cancer`) is a standard UCI (Lichman 2013) dataset that contains 682 hospital patients values captured via a Fine-needle aspiration test. Each patient is described by 9 attributes concerning breast tumoral cells. The task consists in classifying a tumor between benign or malignant. The classes distribution is 35% benign, and 65% malignant.

Unlike the previous datasets, `breast-cancer` is a real-world dataset which cannot be explained with simple rules, hence it is not possible to compare the retrieved rules w.r.t. a given ground truth. For this reason we compared the rules extracted by PRL with the ones extracted by the rule extraction methods described in (Hayashi and Nakano 2015). We directly applied the rules to the entire dataset and the achieved accuracies have been compared. All the methods have been trained using a 90-10% training and test split. This evaluation procedure has been chosen because we only had at our disposal, for each model, the set of extracted rules after a 10-fold cross validation procedure. We trained PRL using rules of degree 2 on the set of relations $\{\geq, \leq\}$. In Table 2 the achieved results are summarized.

Method	# Rules	Accuracy (%)
SSV	3	86.36
GASVM	2	90.03
C-MLP2LN	5	96.92
QSVM-G	12	96.48
ReRXJ48	4	94.28
PRL only 4th	1	92.67
PRL@5	5	96.12
PRL@10	10	**97.95**

Table 2: Accuracy of the rules extracted by the different algorithms. The highest accuracy is highlighted in **bold**.

As evident from the table PRL achieves the best accuracy using the 10 most relevant rules. Nevertheless, even using fewer rules (i.e., 5) it is able to achieve very good results even though methods such as C-MLP2LN and QSVM-G had slightly higher accuracies. Another interesting observation is that by using the 4th most relevant rule alone the proposed method achieves more than 92% of accuracy. However, it is also worth to notice that the first three rules are not enough to get good results, as highlighted in Figure 1.

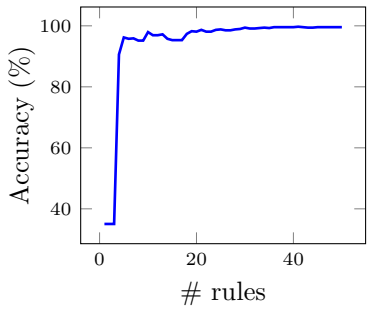
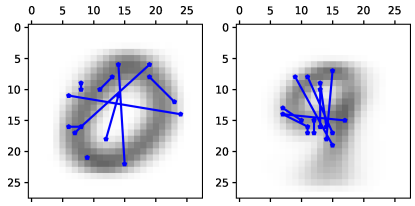


Figure 1: Plot of the accuracy w.r.t. the number of considered rules during the classification.

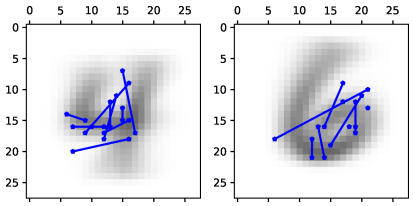
The figure shows the accuracy achieved by considering a limited set of rules (1 up to 50). The algorithm is able to overcome 95% of accuracy quite rapidly (first four rules) and then it increases almost monotonically until reaching 99.56% using 50 rules. However, 50 rules are not a reasonable number when it comes to interpret a classification, and thus in Table 2 we only mentioned the accuracies up to 10 rules. More details about the extracted rules are reported in the supplementary material.

The `mnist` dataset The `mnist` dataset is one of the most widely used dataset for the hand-written digit classification task. The digits are stored in a grey scale 28 by 28 pixel matrix, where each pixel can assume a value between 0 and 255 (0-1 normalized). The task is to recognize the digit represented by an instance.

As for `breast-cancer`, the classification of the `mnist` digits cannot be explained with simple rules. The



(a) 0 versus 9



(b) 4 versus 6

Figure 2: Visualization of the most relevant polynomial features of degree 2. The polynomial features are visualized as segments limited by the involved input variables. The left hand side plot shows the features relevant to discriminate the (a) 0 from the 9 and (b) 4 from the 6. Viceversa in the right hand side plots.

goal here is to use the most relevant features for interpreting, in a human fashion, which are the visual characteristics that are leveraged by the model to discriminate each class w.r.t. another. Experiments have been performed using polynomial features of degree 2. Figure 2 illustrates two examples of the most relevant features used by the model to distinguish: (a) 0 from a 9 (left) and viceversa (right); (b) 4 from a 6 (left) and viceversa (right).

The features are represented as segments between the two involved variables (i.e., pixels) in each monomial (i.e., rule). In the background the average digit of that class is depicted.

From the first plots (Figure 2a) it is clear that in order to discriminate a 0 from a 9 the most important characteristics for the algorithm are the "big" curvature for the 0, and the smaller one for the 9. Similarly, in Figure 2b the same kind of features are important to characterize the 6 w.r.t. the 4, while the 4 is mainly recognized by its horizontal dash.

The complete set of features for every pair of classes have been omitted for space reasons, and can be found in the supplementary material.

Feature Selection

This set of experiments aims to assess the effectiveness of PRL on datasets with many noisy and redundant features. The chosen testbeds have been the datasets of the NIPS 2003 Feature selection challenge (Guyon et al. 2005). All the datasets are freely available at the NIPS 2003 Feature selection challenge site, http://clopinet.com/isabelle/Projects/NIPS2003/. Further details about the datasets are reported in the supplementary material and in (Guyon et al. 2005; Johnson 2009). A common characteristic of these datasets is the huge number of features they have compared to the number of training instances. All the datasets consist of binary classification tasks.

We compared PRL with a standard soft-margin SVM. Given the huge number of features of the target datasets, the linear kernel turned out to be a good kernel for these tasks, with the exception of `madelon` in which the degree 2 polynomial was the best performing kernel for SVM. The C hyper-parameter of the SVM has been validated in the set of values $\{10^{-4}, \ldots, 10^{5}\}$ using a 3-fold cross validation procedure. Experiments have been performed using a 70-30% training and test split. In Table 3 the results achieved by both methods as well as the number of relevant features according to PRL are summarized. In these experiments, the size B of the working set has been set to 2000. As evident from the table, the proposed method is able to achieve better performance than SVM. It is worth to mention that, generally the number of features used by PRL is orders of magnitude less than the number of original features.

Dataset	SVM	PRL	# Relevant/Total feat.
`dorothea`	91.88	**92.69**	476/100k
`gisette`	96.71	**97.19**	900/5k
`madelon`	60.10	**62.75**	1225/250k

Table 3: Accuracy results achieved by SVM and PRL. The last column indicates the number of support preference-feature pairs used by PRL. The best results are highlighted in **bold**.

Conclusions and future work

This paper has proposed a new preference learning framework for classification based on game theoretic concepts. The learning problem is defined as a two-players zero-sum game for which we have given an incremental solution w.r.t. the columns of the game matrix. We provided theoretical guarantees about the convergence of the algorithm as well as an extensive set of experiments demonstrating its effectiveness. We also showed the capabilities of PRL in identifying explanation rules for interpreting the classification.

In this work we only use PRL in classification contexts. In the future we would like to test our method on other PL settings, including instance and label ranking tasks. From a computational point of view, the algorithm can be further improved starting from the column generation policy. In the future we aim to explore new approaches to select in a smart way the columns which have good chances of being included in the strategy. Moreover, we also intend to relax the formulation in order to get a soft margin version of the algorithm.

Finally, another point of improvement is represented by the random feature generation. A possible future path in this direction can be to explore feature generation methods such as the one proposed in (Rahimi and Recht 2007). The same algorithm can also be applied to other applications which are based on a large number of explicit features such as classification of structured data (e.g., graphs) or relational learning.

References

Aiolli, F., and Sperduti, A. 2004. Learning preferences for multiclass problems. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, 17–24. Cambridge, MA, USA: MIT Press.

Aiolli, F., and Sperduti, A. 2010. A preference optimization based unifying framework for supervised learning problems. In *Preference Learning*.

Aiolli, F.; Da San Martino, G.; and Sperduti, A. 2008. A kernel method for the optimization of the margin distribution. In *Artificial Neural Networks - ICANN 2008*, 305–314.

Bopardikar, S. D., and Langbort, C. 2014. Incremental approximate saddle-point computation in zero-sum matrix games. In *53rd IEEE Conference on Decision and Control*, 1936–1941.

Bopardikar, S. D.; Borri, A.; Hespanha, J. P.; Prandini, M.; and Benedetto, M. D. D. 2010. Randomized sampling for large zero-sum games. In *49th IEEE Conference on Decision and Control (CDC)*, 7675–7680.

Brown, G. W. 1951. Iterative solutions of games by fictitious play. *In: Activity Analysis of Production and Allocation* 374–376.

Freund, Y., and Schapire, R. E. 1996. Game theory, on-line prediction and boosting. In *COLT*, 325–332.

Freund, Y., and Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* 55(1):119–139.

Freund, Y., and Schapire, R. E. 1999. Adaptive game playing using multiplicative weights. *Games and Economic Behavior* 29(1-2):79–103.

Fürnkranz, J., and Hüllermeier, E. 2010. *Preference Learning*. Springer, 1st edition.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*. 2672–2680.

Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; and Pedreschi, D. 2018. A survey of methods for explaining black box models. *ACM Comput. Surv.* 51(5):93:1–93:42.

Guyon, I.; Gunn, S.; Ben-Hur, A.; and Dror, G. 2005. Result analysis of the nips 2003 feature selection challenge. In Saul, L. K.; Weiss, Y.; and Bottou, L., eds., *Advances in Neural Information Processing Systems 17*. MIT Press. 545–552.

Hayashi, Y., and Nakano, S. 2015. Use of a recursive-rule extraction algorithm with j48graft to achieve highly accurate and concise rule extraction from a large breast cancer dataset. *Informatics in Medicine Unlocked* 1:9 – 16.

Hofmann, T.; Scholkopf, B.; and Smola, A. J. 2008. Kernel methods in machine learning. *The Annals of Statistics* 36(3):1171–1220.

Ioannidis, S., and Loiseau, P. 2013. Linear regression as a non-cooperative game. In *Proceedings of the 9th International Conference on Web and Internet Economics - Volume 8289*, WINE 2013, 277–290.

Johnson, N. 2009. A study of the nips feature selection challenge.

Kimeldorf, G. S., and Wahba, G. 1971. Some results on tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications* 33(1):82–95.

Lichman, M. 2013. UCI machine learning repository.

Polato, M., and Aiolli, F. in press 2019. Boolean kernels for rule based interpretation of support vector machines. *Neurocomputing*.

Rahimi, A., and Recht, B. 2007. Random features for large-scale kernel machines. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, 1177–1184.

Rezek, I.; Leslie, D. S.; Reece, S.; Roberts, S. J.; Rogers, A.; Dash, R. K.; and Jennings, N. R. 2008. On similarities between inference in game theory and machine learning. *J. Artif. Intell. Res.* 33:259–283.

Schapire, R. E.; Freund, Y.; Barlett, P.; and Lee, W. S. 1997. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, 322–330.

Tsochantaridis, I.; Hofmann, T.; Joachims, T.; and Altun, Y. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, 104–. New York, NY, USA: ACM.

von Neumann, J. 1928. Zur theorie der gesellschaftsspiele. 100:295–320.

Wang, J.; Zhao, P.; Hoi, S. C. H.; and Jin, R. 2014. Online feature selection and its applications. *IEEE Transactions on Knowledge and Data Engineering* 26(3):698–710.

Wu, X.; Yu, K.; Ding, W.; Wang, H.; and Zhu, X. 2013. Online feature selection with streaming features. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(5):1178–1192.

Xu, H.; Ford, B. J.; Fang, F.; Dilkina, B. N.; Plumptre, A. J.; Tambe, M.; Driciru, M.; Wanyama, F.; Rwetsiba, A.; Nsubaga, M.; and Mabonga, J. 2017. Optimal patrol planning for green security games with black-box attackers. In *GameSec*.

Yufei Liu, D. P. 2017. A novel kernel SVM algorithm with game theory for network intrusion detection. *KSII Transactions on Internet and Information Systems* 11(8).