ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6.
Available from http://www.i6doc.com/en/.

# Boolean kernels for interpretable kernel machines

Mirko Polato and Fabio Aiolli

University of Padova - Department of Mathematics
Via Trieste, 63, 35121 Padova - Italy

**Abstract**. Most of the machine learning (ML) community's efforts in the last decades have been devoted to improving the power and the prediction quality of ML models at the expense of their interpretability. However, nowadays, ML is becoming more and more ubiquitous and it is increasingly demanded the need for models that can be interpreted. To this end, in this work we propose a method for extracting explanation rules from a kernel machine. The core idea is based on using kernels with feature spaces composed by logical propositions. On top of that, a searching algorithm tries to retrieve the most relevant features/rules that can be used to explain the trained model. Experiments on several benchmarks and artificial datasets show the effectiveness of the proposed approach.

## 1 Introduction

The lack of interpretability of many machine learning methods, e.g., kernel machines and (deep) neural networks, makes hard their application in scenarios in which explanations are as important as the prediction quality, for example, support systems for physicians and recommender systems. Also the European Parliament in one of the Articles of its "General Data Protection Regulation" [1] underlines the need of explanations when a decision regarding a user is taken automatically by a machine. In the past, some efforts have been devoted in order to alleviate this black-box nature of ML models [2, 3]. In this work we focus on interpreting kernel machines, in particular SVM. In the literature [2], most of the proposed methods for extracting explanation rules from SVM are based on the definition of regions in the input space that are then converted into *if-then-else* rules. Here, we propose a different approach which works directly in the feature space. Specifically, by means of Boolean kernels, which have shown state-of-the-art performance in binary classification tasks [4], the data are mapped onto an easy-to-interpret feature space, and in such space an SVM is trained (BK-SVM). Since the feature space of a BK-SVM is composed of Boolean rules, it is possible to give a human-readable interpretation of the solution of the SVM by extracting the most influential rules in the decision. So, the main contribution of this work is two-fold: $(i)$ first we present a new Boolean kernel which creates a feature space made of (potentially) all possible monotone DNF formulas over the input variables; $(ii)$ then, we propose an algorithm for extracting from the BK-SVM the most relevant rule which can be used to interpret the solution. Throughout the paper we consider binary valued datasets for binary classification tasks. Formally, $\mathcal{T} \equiv \{\mathbf{x}_i, y_i\}_{i=1}^{L}$ is a training set where $\forall i, \mathbf{x}_i \in \{0,1\}^n$ and $y_i \in \{+1, -1\}$.

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6.
Available from http://www.i6doc.com/en/.

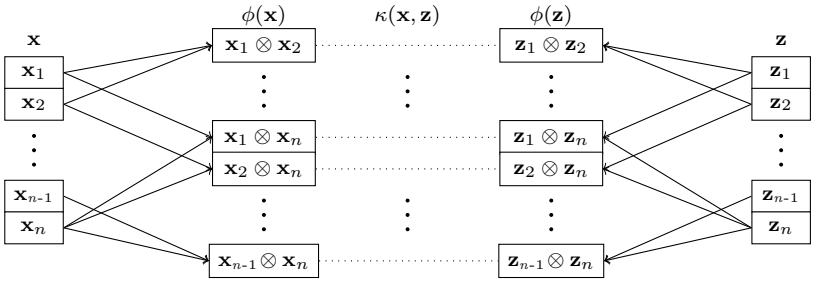


**Fig. 1:** Depiction of a generic Boolean (operator $\otimes$) kernel of arity 2: firstly the input vectors are mapped into the feature space which is formed by all formulas with arity 2 (without repetition). Then, the kernel is computed by "matching" (dotted lines) the corresponding features. Arrows from the input vectors to the feature vectors indicate when a variable influences the formula.

We refer to generic $n$-dimensional Boolean vectors with $\mathbf{x}$ and $\mathbf{z}$, and with the notation $\mathbf{x}^{\mathbf{b}}$ we indicate the component-wise exponentiation, i.e., $x_1^{b_1} x_2^{b_2} \cdots x_n^{b_n}$. Finally, the notation $[\![\cdot]\!]$ represents the indicator function.

## 2 Boolean kernels

In the literature, the term Boolean kernel has been used as a synonym of the so-called DNF kernel [5]. However, in a recent work [4] a broader definition has been provided: Boolean kernels are kernel functions which take binary vectors as input and apply the dot-product in a feature space where each dimension represents a logical proposition over the input variables. The general idea behind Boolean kernels is depicted in Figure 1.

***Monotone Conjunctive kernel*** One of the simplest Boolean kernel is the monotone Conjunctive kernel (mC-kernel) [4]. As the name suggests, its feature space is composed by all conjunctions of exactly $c$ different input variables, where $c$ is an hyper-parameter. Hence, the mC-kernel of arity $c$ between $\mathbf{x}$ and $\mathbf{z}$ computes the number of *true* conjunctions of $c$ literals in common between $\mathbf{x}$ and $\mathbf{z}$. Formally, the embedding of the mC-kernel of arity $c$ is given by $\boldsymbol{\phi}_\wedge^c : \mathbf{x} \mapsto (\boldsymbol{\phi}_\wedge^{(\mathbf{b})}(\mathbf{x}))_{\mathbf{b} \in \mathbb{B}_c}$, where $\mathbb{B}_c = \{\mathbf{b} \in \{0,1\}^n \mid \|\mathbf{b}\|_1 = c\}$, and $\boldsymbol{\phi}_\wedge^{(\mathbf{b})}(\mathbf{x}) = \mathbf{x}^{\mathbf{b}}$. The dimension of the resulting feature space is $\binom{n}{c}$. Thus, the mC-kernel of arity $c$ is computed by $\kappa_\wedge^c(\mathbf{x},\mathbf{z}) = \binom{\langle \mathbf{x},\mathbf{z} \rangle}{c}$.

***Monotone Disjunctive kernel*** Similarly to the mC-kernel, the monotone Disjunctive kernel (mD-kernel) [4] of arity $d$ between $\mathbf{x}$ and $\mathbf{z}$ computes the number of *true* disjunctions of $d$ literals in common between $\mathbf{x}$ and $\mathbf{z}$. Thus, the embedding of the mD-kernel is the same as the mC-kernel, however the logical interpretation is different since the combinations of variables represent disjunctions. Formally, the embedding of the mD-kernel of arity $d$ is given by

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6. Available from http://www.i6doc.com/en/.

$\boldsymbol{\phi}^d_\vee : \mathbf{x} \mapsto (\boldsymbol{\phi}^{(\mathbf{b})}_\vee(\mathbf{x}))_{\mathbf{b}\in\mathbb{B}_d}$, with $\boldsymbol{\phi}^{(\mathbf{b})}_\vee(\mathbf{x}) = [\![\langle \mathbf{x}, \mathbf{b}\rangle > 0]\!]$. The corresponding mD-kernel is computed by

$$\kappa^d_\vee(\mathbf{x}, \mathbf{z}) = \binom{n}{d} - \binom{n - \langle \mathbf{x}, \mathbf{x}\rangle}{d} - \binom{n - \langle \mathbf{z}, \mathbf{z}\rangle}{d} + \binom{n - \langle \mathbf{x}, \mathbf{x}\rangle - \langle \mathbf{z}, \mathbf{z}\rangle + \langle \mathbf{x}, \mathbf{z}\rangle}{d}. \quad (1)$$

The full derivation of Equation (1) is presented in [4].

## 3 The feature space of monotone DNFs

As described in [4], by composing the mC-kernel and the mD-kernel it is possible to compute the mDNF-kernel (monotone Disjunctive Normal Form) in which the feature space is composed by monotone DNF formulas of the input variables. A shortcoming of the mDNF-kernel defined in such way is that the mDNF formulas have a fixed form, that is, they are composed by disjunctions of $d$ conjunctive clauses made of $c$ literals. In order to overcome this limitation, we first create a feature space composed by all conjunctions up to a certain arity $C$, by concatenating the feature spaces of $\kappa^c_\wedge$ for $c \in [1, C]$, that is $\phi^C_\Sigma(\mathbf{x}) = (\phi^1_\wedge(\mathbf{x}), \phi^2_\wedge(\mathbf{x}), \ldots, \phi^C_\wedge(\mathbf{x}))$. The corresponding kernel can be implicitly computed [6] by $\kappa^C_\Sigma(\mathbf{x}, \mathbf{z}) = \sum_{c=1}^{C} \kappa^c_\wedge(\mathbf{x}, \mathbf{z})$. Now, by composing $\phi^C_\Sigma$ with $\phi^d_\vee$ (for some $d$) we obtain a feature space constituted of all possible mDNFs made of $d$ conjunctive clauses of at most $C$ literals. This kernel can be calculated by replacing $\langle \mathbf{x}, \mathbf{z}\rangle$ (i.e., the linear kernel) with $\kappa^C_\Sigma(\mathbf{x}, \mathbf{z})$ and $n$ with $\sum_{c=1}^{C}\binom{n}{c}$ in Eq. (1). Finally, by summing up all these kernels with $d \in [1, D]$ we obtain a kernel with a feature space composed of all possible mDNF formulas with at most $D$ conjunctive clauses of at most $C$ literals. Formally, $\kappa^{D,C}_*(\mathbf{x}, \mathbf{z}) = \sum_{d=1}^{C} \kappa^d_\vee(\phi^C_\Sigma(\mathbf{x}), \phi^C_\Sigma(\mathbf{z}))$, which is a valid positive semi-definite kernel because it is a result of closure properties [6].

## 4 Interpreting BK-SVM

One of the biggest advantages of using Boolean kernels is that the features in the embedding space are easy to interpret, and this characteristic can be leveraged to explain the solution of a kernel machine, e.g., SVM. In particular, the most influential features (i.e., logical rules) in the solution can be extracted in order to provide a human-readable interpretation of the decision. From the Representer Theorem [6] we know that the solution of an SVM can be written as $\mathbf{w} = \sum_{i\in\mathcal{S}} y_i\alpha_i\phi(\mathbf{x}_i)$, where $\mathcal{S}$ is the set of support vector indexes, and $\alpha_i \geq 0$ are the contributions of the support vectors to the solution. Hence, the weight associated to a feature $f$, i.e., a Boolean rule, inside the feature space induced by $\phi$ can be calculated by:

$$w_f = \sum_{i\in\mathcal{S}} y_i\alpha_i\phi_f(\mathbf{x}_i) = \sum_{i\in\mathcal{S}} y_i\alpha_i[\![f(\mathbf{x}_i)]\!] = \sum_{i\in\mathcal{S}^+} \alpha_i[\![f(\mathbf{x}_i)]\!] - \sum_{i\in\mathcal{S}^-} \alpha_i[\![f(\mathbf{x}_i)]\!], \quad (2)$$

where $\mathcal{S}^+$ (resp. $\mathcal{S}^-$) is the set of positive (resp. negative) support vector indexes, and $\sum_{i\in\mathcal{S}^+}\alpha_i = \sum_{i\in\mathcal{S}^-}\alpha_i$. Our goal is to find the formula $f$ such

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6.
Available from http://www.i6doc.com/en/.

to maximize the value $w_f$. It is easy to show that, if $(i)$ the set is linearly separable, $(ii)$ the target concept is defined by a formula $g$, and $(iii)$ the feature space contains $g$, then $g = \text{argmax}_f \; w_f = \sum_{i \in \mathcal{S}^+} \alpha_i$. It is noteworthy that $w_f$ is maximized for every $f$ consistent with the support vectors, and hence the best formula could not be unique. In the case of non-separability, finding the rule that maximizes the value of $w_f$ is still a good heuristic since it is a way to minimize the loss with respect to the decision function.

### 4.1 Rule extraction via Genetic Algorithm

In order to find the best rule, we adopted (as a proof of concept) a genetic algorithm (GA) based optimization. The design choices for the GA are described in the following:

**population** it is formed by 500 randomly initialized individuals, i.e., mDNF formulas with at most $D$ conjunctions made of at most $C$ literals;

**fitness** given a formula $f$, its fitness is equals to the weight $w_f$ as in Eq. (2);

**crossover** given two mDNF formulas $f$ and $g$, the crossover operator creates a new individual by randomly selecting a subset of the conjunctive clauses from the union of $f$ and $g$ while keeping the number of clauses $\leq D$.

**mutation** given a mDNF formula, the mutation operator randomly performs one out of the following three actions: $(i)$ removing one of the conjunctive clauses (when applicable); $(ii)$ adding a new random conjunctive clause; $(iii)$ replacing a literal in one of the conjunctions;

**selection** we adopted the *elitist selection* (20%) strategy to guarantee that the solution quality will not decrease.

## 5 Experiments

The experiments have been performed[1] on 10 binary datasets in which the number of ones is the same for every instance. This is not a limitation since, given a dataset with categorical features, each instance can be converted into a fixed norm binary vector by means of the one-hot encoding [7]. The artificial datasets (indicated by the prefix `art-`) have been created in such a way that the positive class can be described by a mDNF formula over the input variables. The details of the datasets are summarized in Table 1. We evaluated the proposed algorithm in terms of the most used metrics for evaluating explanation rules [2], namely, *comprehensibility*, *accuracy* and *fidelity*. Comprehensibility is the extent to which the extracted representations are humanly comprehensible. In our case we can assume high comprehensibility because the retrieved rules are simple (and short) logical propositions over the input binary variables. The accuracy of a classification function (or rule) $f$ over the test set $\mathcal{T}_{\text{ts}}$ is equal to $|\{(\mathbf{x}, y) \in \mathcal{T}_{\text{ts}} \mid [\![ f(\mathbf{x}) \iff y = +1 ]\!]\}|/|\mathcal{T}_{\text{ts}}|$. The *fidelity* over the test set $\mathcal{T}_{\text{ts}}$ of a

[1] All the experiments have been implemented in python 2.7 using the modules Scikit-Learn, MKLpy and pyros available in the PyPi repository.

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6. Available from http://www.i6doc.com/en/.

| Dataset | #Inst. | #Ft. | Rule |
|---|---|---|---|
| tic-tac-toe* | 958 | 27 | mDNF, $d = 8, c = 3$ |
| monks-1* | 432 | 17 | $(x_0 \wedge x_3) \vee (x_1 \wedge x_4) \vee (x_2 \wedge x_5) \vee x_{11}$ |
| monks-3* | 432 | 17 | mDNF, $d = 7, c = 2$ |
| art-d2-c4 | 1000 | 30 | $(x_{11} \wedge x_9 \wedge x_1 \wedge x_{14}) \vee (x_{27} \wedge x_{17})$ |
| art-d3-c3 | 1000 | 30 | $(x_3 \wedge x_{28}) \vee x_{27} \vee (x_{14} \wedge x_7 \wedge x_{26})$ |
| art-d4-c2 | 1000 | 30 | $x_3 \vee (x_0 \wedge x_7) \vee (x_5 \wedge x_9) \vee x_8$ |
| art-d4-c3 | 1000 | 30 | $(x_{25} \wedge x_{21}) \vee (x_{15} \wedge x_5 \wedge x_{19}) \vee (x_0 \wedge x_{26}) \vee (x_8 \wedge x_{21} \wedge x_{20})$ |
| art-d5-c4 | 1000 | 30 | mDNF, $d = 5, c \leq 4$ |
| art-d5-c5 | 1000 | 30 | mDNF, $d = 5, c \leq 5$ |

**Table 1:** Information of the datasets: number of instances, number of binary features and the rule which describes the positive class. (*) means that the dataset is freely available in the UCI repository.

| | SVM | | Best Rule | | Fidelity | | |
|---|---|---|---|---|---|---|---|
| Dataset | Train | Test | Train | Test | Train | Test | GA #Gen. |
| tic-tac-toe | 100.00 ±0.00 | 98.33 ±0.87 | 100.00 ±0.00 | 100.00 ±0.00 | 100.00 ±0.00 | 98.33 ±0.87 | 358.00 ±156.81 |
| monks-1 | 100.00 ±0.00 | 100.00 ±0.00 | 100.00 ±0.00 | 100.00 ±0.00 | 100.00 ±0.00 | 100.00 ±0.00 | 9.20 ±3.37 |
| monks-3 | 100.00 ±0.00 | 100.00 ±0.00 | 100.00 ±0.00 | 100.00 ±0.00 | 100.00 ±0.00 | 100.00 ±0.00 | 230.40 ±385.79 |
| art-d2-c4 | 100.00 ±0.00 | 98.87 ±0.50 | 99.89 ±0.23 | 99.40 ±0.80 | 99.89 ±0.23 | 99.07 ±0.68 | 10.60 ±3.55 |
| art-d3-c3 | 100.00 ±0.00 | 97.13 ±1.13 | 100.00 ±0.00 | 100.00 ±0.00 | 100.00 ±0.00 | 97.13 ±1.13 | 14.60 ±7.34 |
| art-d4-c2 | 100.00 ±0.00 | 97.87 ±0.75 | 100.00 ±0.00 | 100.00 ±0.00 | 100.00 ±0.00 | 97.87 ±0.75 | 15.0 ±2.28 |
| art-d4-c3 | 100.00 ±0.00 | 95.07 ±1.34 | 100.00 ±0.00 | 100.00 ±0.00 | 100.00 ±0.00 | 95.07 ±1.34 | 35.20 ±19.36 |
| art-d5-c4 | 100.00 ±0.00 | 96.00 ±0.67 | 99.71 ±0.57 | 99.20 ±1.60 | 99.71 ±0.57 | 96.00 ±0.67 | 340.40 ±338.10 |
| art-d5-c5 | 100.00 ±0.00 | 94.27 ±0.85 | 99.97 ±0.06 | 99.40 ±0.33 | 99.97 ±0.06 | 94.20 ±0.85 | 61.20 ±17.68 |

**Table 2:** Experimental results averaged over 5 runs: for each dataset the accuracy (%) in both training and test is reported for SVM and for the extracted rule. It is also reported the fidelity of the rule w.r.t the SVM as well as the average number of generations required to the GA to find the best rule.

rule $f$ w.r.t. a decision function $h$ learnt by a learning algorithm is computed by $|\{(\mathbf{x}, y) \in \mathcal{T}_{\text{ts}} \mid [\![f(\mathbf{x}) \iff h(\mathbf{x}) = +1]\!]\}|/|\mathcal{T}_{\text{ts}}|$. For each dataset the experiments have been repeated 5 times by using different 70%-30% training-test splits. In each experiment an hard-SVM with the kernel $\kappa_*^{5,10}$ has been trained over the training set and then the most relevant formula has been extracted using the GA (described in Section 4.1) with $C = 5$, $D = 10$, the mutation probability set to 0.6 and the maximum number of generations set to $10^3$. It is worth to notice that the computational time for calculating $\kappa_*$ is in the order of milliseconds for each dataset.

### 5.1 Results

The achieved results are summarized in Table 2. As evident from the table, in every dataset the best rule extracted by the GA is indeed the one which (almost always) explains the label and the decision of the SVM (the fidelity is very high).

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6.
Available from http://www.i6doc.com/en/.

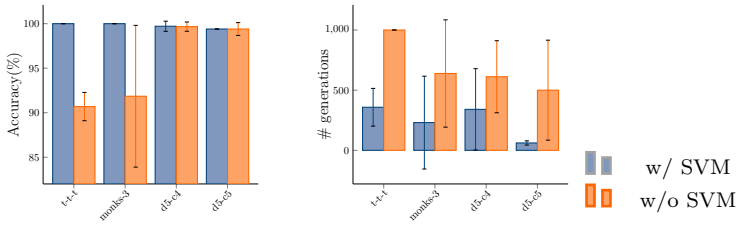


**Fig. 2:** Comparison between the GA guided by the SVM (w/) and w/o the SVM. The plot on the left shows the average accuracy on the test set, while the plot on the right shows the average number of generations required by the GA to find the best rule.

Moreover, despite the huge search space (on average $10^{45}$ formulas), the number of generations required to find the best rule is very low. To highlight how the weights learned by the SVM are indeed useful to guide the research of the GA (through the fitness), we also tried to retrieve the best formula by using the same GA with $\alpha_i = 1/L, \forall\, i \in [1, L]$. In this case the fitness corresponds to the training accuracy. Figure 2 shows the comparison between the GA w/ and w/o SVM. From the figure, it is evident that using the GA guided by the SVM ensures that a better rule will be found with fewer generations. It is also worth to mention that computing the fitness over all the training set is significantly less efficient than calculating it for the support vectors only.

## 6 Future work

In the future we aim to test this approach on real-world datasets (both categorical and real-valued) which are generally noisy and in which we cannot assume of explaining the solution with just one rule. Moreover, a more in-depth theoretical analysis need to be conducted in order to further support the empirical results. Finally, other strategies for extracting the best rules should be tested.

## References

[1] Bryce Goodman and Seth Flaxman. Eu regulations on algorithmic decision-making and a "right to explanation", 2016. presented at 2016 ICML WHI 2016, New York, NY.

[2] Nahla Barakat and Andrew P. Bradley. Rule extraction from support vector machines: A review. *Neurocomputing*, 74(1-3):178–190, 2010.

[3] João Guerreiro and Duarte Trigueiros. A unified approach to the extraction of rules from artificial neural networks and support vector machines. In *ADMA*, pages 34–42, 2010.

[4] Mirko Polato, Ivano Lauriola, and Fabio Aiolli. Classification of categorical data in the feature space of monotone dnfs. In *ICANN*, pages 279–286, 2017.

[5] Ken Sadohara. Learning of boolean functions using support vector machines. In *ALT*, pages 106–118, 2001.

[6] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[7] David Money Harris and Sarah L. Harris. *Digital Design and Computer Architecture*. Morgan Kaufmann, Boston, 2013.