Support Vector-based Estimation of Multilinear Games for Feature Selection and Explanation

Majid Mohammadi, Ilaria Tiddi, Annette Ten Teije

Department of Computer Science, Vrije Universiteit Amsterdam
De Boelelaan 1111, 1081 HV Amsterdam
majid.mohammadi690@gmail.com, i.tiddi@vu.nl, annette.ten.teije@vu.nl

Abstract

In recent years, employing Shapley values to compute feature importance has gained considerable attention. Calculating these values inherently necessitates managing an exponential number of parameters—a challenge commonly mitigated through an additivity assumption coupled with linear regression. This paper proposes a novel approach by modeling supervised learning as a multilinear game, incorporating both direct and interaction effects to establish the requisite values for Shapley value computation. To efficiently handle the exponentially increasing parameters intrinsic to multilinear games, we introduce a support vector machine (SVM)-based method for parameter estimation, its complexity is predominantly contingent on the number of samples due to the implementation of a dual SVM formulation. Additionally, we unveil an optimized dynamic programming algorithm capable of directly computing the Shapley value and interaction index from the dual SVM. Our proposed methodology is versatile, and we demonstrate that it can be applied to local explanation and feature selection. Experiments underscore the competitive efficacy of our proposed methods in terms of feature selection and explanation.

1 Introduction

Shapley value-based feature importance is a prominent model-agnostic interpretability method, utilizing an additive feature attribution approach to clarify the predictions of any machine learning model (Lundberg and Lee 2017; Covert and Lee 2021). However, the current explainable methods are limited to providing the importance of individual features and fall short of providing information about feature interaction (i.e., the synergistic influence of a set of features on model prediction), or have exponential complexity for such computations (Tsai, Yeh, and Ravikumar 2023; Sundararajan, Dhamdhere, and Agarwal 2020; von Luxburg and Bordt 2023). In addition, while these explainable methods offer significant insights, they are not readily applicable to feature selection due to their reliance on sampling various feature subsets and calculating their contributions based on a trained model. One approach for using the Shapley value for feature selection involves training a separate model for each subset of features and using a performance metric (e.g., R^2 for regression) as the game value for computing the Shapley values (Lipovetsky and Conklin 2001; Tripathi, Hemachandra, and Trivedi 2020). However, such methods are impractical as they require training an exponential number of models to accurately requisite game values for the Shapley value, becoming readily impractical for even a mediocre number of samples.

In this paper, we put forward an approach based on the multilinear extension of cooperative games, allowing supervised learning tasks to be modeled as multilinear games. This extension holds the current explainable methods as a special case and, additionally, provides insights into the interactions that exist among features. Besides, we show that such an extended formulation can be applied to feature selection problems, where we proved that the parameters of the multilinear model can be used to compute the Shapley value as an importance indicator for feature selection. A significant challenge with this approach is the exponential growth in main and interaction effects as the number of features increases. To address this, we employ a support vector machine (SVM)-based estimation whose dual formulation significantly simplifies the problem and derives a data-driven representation of the Shapley value and interaction index based on the dual SVM solution. We also unveil an efficient dynamic programming algorithm that computes the Shapley values based on the data-driven Shapely value representation.

In summary, the paper has the following contributions: (i) We formulate the supervised learning task as multilinear games, employing direct and interaction effects as the characteristic function of a game; such a formulation can provide more insights when used as local explainer, and its extension is guaranteed to compute the Shapley value as the feature importance for feature selection; (ii) We effectively handle the exponential parameters in multilinear games by employing SVM, focusing the complexity of the problem on the number of samples rather than the myriad of features and their interactions; (iii) We put forward a data-driven representation of the Shapley value based on the dual SVM solution and devise a dynamic programming algorithm to efficiently compute the Shapley value or interaction index.

Notation. We show the matrices with upper-case (unbold-faced) letters, the vectors with bold-faced lower-case, scalars with lower-case (unbold-faced) letters, and the sets with curly upper-case letters. The training set containing n samples is denoted by $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$, where $\boldsymbol{x}_i \in R^d$ and $y \in \{-1, 1\}$. The set of all data points $\boldsymbol{x}_i$'s is denoted by $X \in R^{n\times d}$, and

$X_{i\bullet}$ and $X_{\bullet j}$ refer to the i^{th} row and j^{th} column of matrix X, respectively. The data matrix with all interactions is shown by $\hat{X} \in R^{n\times(2^d-1)}$. We also show the set of features by $\mathcal{F}$ and the set of features with interactions by $\hat{\mathcal{F}}$. Obviously, $\mathcal{F}$ and $\hat{\mathcal{F}}$ contain d and 2^d-1 elements, respectively. We also use the one-to-one mapping function $u : \hat{\mathcal{F}} \rightarrow Ind$, where $Ind \in [1, 2^d-1]$ is an integer mapping a set in $\hat{\mathcal{F}}$ to the corresponding column in $\hat{X}$. So, for instance, if we have three features $\mathcal{F} = \{1, 2, 3\}$ and want to have the interaction term $\{1, 2\}$ in the fourth column of $\hat{X}$, then it follows that $u(\{1, 2\}) = 4$. We also use the inverse of u for mapping the index to an interaction set, e.g., $u^{-1}(4) = \{1, 2\}$. The element-wise multiplication is also shown by $\otimes$.

2 Supervised Learning and Cooperative Game Theory

Shapley Value and Möbius Transformation. A cooperative game is characterized by specifying a function for each coalition. For a set of player $\mathcal{F}$, the characteristic function $\mu : 2^{\mathcal{F}} \rightarrow R$ assigns a value to each subset of players. The function represents the collective payoff of a set of players when forming the coalition. From a supervised learning perspective, the features and the label (or predicted variable) serve as the players and the payoff, respectively.

Given a characteristic function, the Shapley value is a solution concept in cooperative game theory that concerns the attribution of a payout among the involved players (Shapley 1953). A salient property of the Shapley value is that it is the unique value that fulfills the four axioms of efficiency, symmetry, dummy, and additivity (Shapley 1953). The Shapley value computation is based on the marginal contributions of each feature to all feature subsets. Let $\mu(\mathcal{S})$ be the value for a feature subset $\mathcal{S}$, the marginal contributions of feature i to $\mathcal{S}$ is $\mu(\mathcal{S}\cup\{i\})-\mu(\mathcal{S}), \forall \mathcal{S} \subseteq \mathcal{F}\setminus\{i\}$. In particular, the Shapley value for feature i, shown by ν_i is the weighted average of all marginal contributions and is defined as (Shapley 1953):

$$\nu_i = \sum_{\mathcal{S}\subseteq\mathcal{F}\setminus\{i\}} \frac{|\mathcal{S}|!(|\mathcal{F}|-|\mathcal{S}|-1)!}{|\mathcal{F}|!}\Big[\mu(\mathcal{S}\cup\{i\})-\mu(\mathcal{S})\Big], \quad (1)$$

For feature importance, the $\mu(\mathcal{S})$ values can be computed by retaining a model for each subset of features (Lipovetsky and Conklin 2001); for explanation, however, recent approaches use a sampling approximation of equation (1) with no need of having a model for all feature subsets (Lundberg and Lee 2017; Mitchell et al. 2022; Datta, Sen, and Zick 2016; Štrumbelj and Kononenko 2014). A useful representation of the Shapley value is computed by using the Möbius transformation. The set function μ can be represented as (Shapley 1953; Grabisch 1996):

$$\mu(B) = \sum_{\mathcal{A}\subseteq\mathcal{B}} m_\mu(\mathcal{A}), \quad \forall \mathcal{B} \subseteq \mathcal{F}, \quad (2)$$

and the Möbius transformation m_μ can be written as follows:

$$m_\mu(\mathcal{A}) = \sum_{\mathcal{B}\subseteq\mathcal{A}} (-1)^{|\mathcal{A}\setminus\mathcal{B}|}\mu(\mathcal{B}).$$

The Shapley value as in equation (1) can be represented by using the Möbius transformation as (Grabisch 1996):

$$\nu_i = \sum_{\mathcal{B}\subseteq\mathcal{F}|i\in\mathcal{B}} \frac{1}{|\mathcal{B}|} m_\mu(\mathcal{B}). \quad (3)$$

The idea of averaging over the marginal contributions for computing the feature importance can be extended to compute the Shapley interaction index. In particular, for any $\mathcal{T} \subseteq \mathcal{F}$, we need to compute the marginal contributions $\mu(\mathcal{S}\cup\mathcal{T}) - \mu(\mathcal{S}), \forall \mathcal{S} \subseteq \mathcal{F}\setminus\mathcal{T}$. The Shapley interaction index for feature subset $\mathcal{T}$ is then defined as (Grabisch and Roubens 1999):

$$I_\nu(\mathcal{T}) = \sum_{\mathcal{B}\subseteq\mathcal{F}|\mathcal{T}\subseteq\mathcal{B}} \frac{1}{|\mathcal{B}|-|\mathcal{T}|+1} m_\mu(\mathcal{B}). \quad (4)$$

Multilinear Extension of Games. The multilinear extension of a cooperative game provides a way to extend a discrete game into a continuous setting (Owen 1972, 1988). The multilinear extension can used to find the parameters of the characteristic function of a game when there are multiple games available from the players, each game with a different participation level (Owen 1988). For a traditional cooperative game with a player (or here feature) set $\mathcal{F}$ and a characteristic function $\mu : 2^{\mathcal{F}} \rightarrow \mathbb{R}$, the multilinear extension $G : [0, 1]^d \rightarrow \mathbb{R}$ is defined as follows:

$$G(\boldsymbol{x}) = \sum_{\mathcal{S}\subseteq\mathcal{F}} \mu(\mathcal{S}) \prod_{i\in\mathcal{S}} x_i \prod_{j\in\mathcal{F}\setminus\mathcal{S}} (1-x_j) \quad (5)$$

Here, x_i is a continuous variable between 0 and 1 that represents the extent to which player i is part of a coalition. This formula allows for fractional coalition memberships and thereby converts the game into a continuous form. Also, it is shown that the integration of the first derivative of equation (5) results in the Shapley value (Owen 1972). It is also shown that G can be represented by the Möbius transformation of μ as:

$$G(\boldsymbol{x}) = \sum_{\mathcal{S}\subseteq\mathcal{F}} m_\mu(\mathcal{S}) \prod_{i\in\mathcal{S}} x_i. \quad (6)$$

The extension to this formulation for multichoice games is also provided, which extends the continuous variable x_i beyond the hypercube (Jones and Wilson 2010; Borkotokey, Hazarika, and Mesiar 2015).

Supervised Learning as Multilinear Game. The multilinear extension of games as in equation (6) provides a compact formulation that could be used in the supervised learning approaches instead of conventional linear models. We first define multilinear game-theoretic learning.

Definition 1 *A multilinear model for supervised learning is:*

$$y = h\Big(b + \sum_{\mathcal{S}\subseteq\mathcal{F}|j=u(\mathcal{S})} m_j \prod_{i\in\mathcal{S}} x_i\Big), \quad \boldsymbol{m} \in R^{2^d-1}, \boldsymbol{x} \in R^d, \quad (7)$$

where h is a link function, b is a bias term, and $m_j = m_\mu\big(u^{-1}(j)\big)$.

The multilinear model, delineated in equation (7), expands upon the traditional linear model by incorporating feature interactions, which is already known in the fields of machine learning and statistics. What sets this model apart is how interaction effects are interpreted through the lens of the multilinear extension. In particular, the main and interaction effects could be used to compute game-theoretic indicators, such as Shapley value as an importance indicator, which enjoys a rigorous theoretical foundation. In the following, we discuss how the multilinear model can be used for the local explanation and feature selection.

Local Explanation: Given a trained model f and an input $\boldsymbol{x}$, local explanation creates subsets of these features and computes the model's prediction for each subset. For a subset $\mathcal{S} \in \mathcal{F}$, the prediction $f(\mathcal{S})$ is calculated by replacing the features not in $\mathcal{S}$ with values from a background dataset, which imitates the presence and absence of a feature (Lundberg and Lee 2017; Covert, Lundberg, and Lee 2020). As a result, $f(\mathcal{S})$ is an approximation of the characteristic function for the features in $\mathcal{S}$, i.e., $f(\mathcal{S}) \approx \mu(\mathcal{S})$. On the other hand, the multilinear extension of games boils down to the game when $\boldsymbol{x}$ is binary. So, $G(\boldsymbol{x}_{\mathcal{S}}) = \mu(S)$ where $\boldsymbol{x}_{\mathcal{S}} \in \{0,1\}^d$ and its i^{th} element is one only if $i \in \mathcal{S}$ (and otherwise it is zero) (Owen 1972). Thus, the multilinear model is adapted as the following problem to identify the values of the cooperative games for a local explanation:

$$\min_{\boldsymbol{m},b} \sum_{\mathcal{S}\subset\mathcal{F}} \left(f(\mathcal{S}) - b - \sum_{\mathcal{T}\subseteq\mathcal{F}|j=u(\mathcal{T})} m_j \prod_{i\in\mathcal{T}} x_i \right)^2. \quad (8)$$

In contrast to the additive explanation methods like Kernel SHAP that compute the Shapley value of features, the above problem estimates the characteristic functions of the cooperative games between features, according to which not only the Shapley value but the interaction indices could also be computed. In particular, identifying the parameters $\boldsymbol{m}$ from the above minimization is through a finite number of samples, and then one can readily compute the Shapley value and interaction index from equations (3) and (4).

Feature Selection. For feature selection, we take the training instance as the repeated games among features with the payoff being the predicted label, and the goal is to find an aggregated set of parameters of the game from such repeated games among features. From this perspective, this is in line with the use of multilinear games (Owen 1988), but the domain of x_i's for feature selection is the real space $\mathcal{R}$, and not $[0,1]$. We now show through the following theorem that computing m_j's as in equation (7), we can compute the Shapley value based on its Möbius transformation as in equation (3). All proofs are presented in Appendix B.

Theorem 1 *Let $X_{\bullet j} \sim N(1,\sigma), \forall j \in \{1,...,d\}$ are independent features, and assume that a multilinear model as in equation* (7) *is fitted to the data. Then, the Shapley value of each feature is computed by equation* (3)*.*

We also discuss in Appendix C that the mean of features has just a scaling effect on our estimation. The challenge is that the multilinear model contains an exponential number of parameters, which is computationally burdensome. The next section presents efficient algorithms for such computations.

3 Support Vector-based Shapley Value Learning

This section presents a support vector-based method for Shapley value estimation. We focus on the SVM for binary classification, but the results could be simply generalized to other types of problems, such as conventional and ordinal regression, as well as arguably any kernel learning algorithm.

3.1 Kernel Support Vector Machine for Multilinear Extension

Multilinear Feature Mapping and SVM. Given an instance $\boldsymbol{x}$, we define a multilinear mapping $\phi_{ML} : R^d \to R^{2^d-1}$ that contains all the feature interactions and is defined as,

$$\phi_{ML}(\boldsymbol{x}) = (x_1, ..., x_d, x_1x_2, ..., x_1x_2...x_d). \quad (9)$$

Then, for a training set for binary classification, the SVM seeks to find a hyperplane $\boldsymbol{m}^T\phi_{ML}(\boldsymbol{x}) + b$ by solving the following minimization (Cortes and Vapnik 1995):

$$\min_{\boldsymbol{m},b} \frac{1}{2}\|\boldsymbol{m}\|^2 + C\sum_i \max\Big(0, 1 - y_i(\boldsymbol{m}^T\phi_{ML}(\boldsymbol{x}_i) + b)\Big), \quad (10)$$

where $C > 0$ is a trade-off parameter between the loss function and regularization. Since the dimension of feature space is potentially big (exponential here), the SVM provides a dual formulation for minimization (10) as:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \phi_{ML}(\boldsymbol{x}_i)^T\phi_{ML}(\boldsymbol{x}_j) - \sum_i \alpha_i$$
$$s.t. \quad \sum_i \alpha_i y_i = 0, \quad 0 \le \boldsymbol{\alpha} \le C. \quad (11)$$

An interesting property of problem (11) is that it depends only on the number of samples and not on features (or its mapping to a higher dimensional space). Also, if we map $\boldsymbol{x}_i$'s to a higher dimensional space $\phi_{ML}(.)$, we only need to know the inner product of the points in that space (i.e., $\phi_{ML}(\boldsymbol{x}_i)^T\phi_{ML}(\boldsymbol{x}_j)$) and use minimization (11) for classification. Realizing such an inner product is known as the kernel trick and the corresponding inner product is known as the kernel function.

Full and q-additive Kernel functions. To compute the kernel function in ϕ_{ML}, we consider the case that we account for all interactions, as well as when we restrict the order of interactions.

For all interactions, we show in the following lemma that the inner product of points in the $\phi_{ML}()$ space can be realized very efficiently.

Lemma 1 *The kernel function for points $\boldsymbol{x}$ and $\boldsymbol{z}$ with the multilinear extension is computed as:*

$$k_{ML}(\boldsymbol{x},\boldsymbol{z}) = \phi_{ML}(\boldsymbol{x})^T\phi_{ML}(\boldsymbol{z}) = -1 + \prod_{i=1}^{d}(1 + x_i z_i). \quad (12)$$

Lemma 1 provides a linear-time formulation for computing the kernel function for the multilinear extension. However, for some problems, there is some prior knowledge that restricts the order of interactions among features. In addition, some interaction indices fulfill a set of axioms when the interaction order is restricted (Sundararajan, Dhamdhere, and Agarwal 2020; Tsai, Yeh, and Ravikumar 2023). As such, we define the q-order Shapley mapping as follows.

Definition 2 *The multilinear mapping is said to be q-order additive, or simply q-additive, if maximum q features can interact. The corresponding kernel is called the q-additive.*

The q-additive multilinear mapping $\phi^q_{ML}(\boldsymbol{x})$ has the following form:

$$\phi^q_{ML}(\boldsymbol{x}) = (x_1, ...x_d, x_1x_2, ..., x_1...x_q, ..., x_{d-q}...x_d).$$

Given the q-additive multilinear mapping, the q-additive multilinear kernel, shown by k^q_{ML}, cannot be computed by equation (12). Instead, we formulate the q-additive kernel as a dynamic programming problem, whose recursive formula is as follows:

$$k^{\tilde{q},\tilde{d}}_{ML}(\boldsymbol{x},\boldsymbol{z}) = \begin{cases} \sum_{i=1}^{\tilde{d}} x_i z_i & \text{if } \tilde{q} = 1 \\ k^{\tilde{d},\tilde{d}}_{ML}(\boldsymbol{x},\boldsymbol{z}) & \text{if } \tilde{q} > \tilde{d} \\ x_{\tilde{d}} z_{\tilde{d}} \big(k^{\tilde{q}-1,\tilde{d}-1}_{ML}(\boldsymbol{x},\boldsymbol{z})\big) + k^{\tilde{q},\tilde{d}-1}_{ML}(\boldsymbol{x},\boldsymbol{z}) & \text{otherwise.} \end{cases} \tag{13}$$

In equation (13), $k^{\tilde{q},\tilde{d}}_{ML}$ is the $\tilde{q}$-additive kernel for the first $\tilde{d}$ elements, and $\boldsymbol{x}, \boldsymbol{z} \in R^d$, $\tilde{d} \leq d$. We initialize $\tilde{q}$ and $\tilde{d}$ with q (i.e., maximum order of interaction) and d (number of features), and the output of equation (13) is the q-additive multilinear kernel for $\boldsymbol{x}$ and $\boldsymbol{z}$. The first two cases in equation (13) give the solution for cases the interaction is one (i.e., no interaction) and when the interaction order is greater than $\tilde{d}$, respectively, and the last case captures the recursion computation and is based on the first $\tilde{d} - 1$ elements. Appendix D gives the iterative implementation of the dynamic programming approach presented in equation (13).

3.2 Shapley Value Calculation: A Dual SVM Representation and DP Algorithm

Using the multilinear mapping and the corresponding kernel in the dual SVM simplifies the computations; instead of solving a minimization with exponentially numerous parameters as in minimization (10), we solve a minimization with n parameters as in problem (11). To compute the Shapley value and interaction, however, we need to compute the primal SVM solution $\boldsymbol{m}$. The vector $\boldsymbol{m}$ in equation (14) has exponentially many elements in the number of features, and its computation is thus time- and memory-consuming. For only 30 features, for instance, $\boldsymbol{m}$ contains more than one billion elements. We now present some formalization and algorithms to compute the Shapley value and interaction index based on the dual SVM solution, thereby circumventing the computational burdens of computing the primal solution $\boldsymbol{m}$.

Given the dual SVM solution $\boldsymbol{\alpha}$, the primal solution is computed as (Cortes and Vapnik 1995):

$$\boldsymbol{m} = \sum_i \alpha_i y_i \phi_{ML}(\boldsymbol{x}_i). \tag{14}$$

Contingent on this relation between $\boldsymbol{m}$ and $\boldsymbol{\alpha}$, the following theorem provides a data-driven representation for Shapley value and interaction index (see Appendix B for the proof).

Theorem 2 *The interaction index in equation* (4) *can be represented based on the dual SVM solution $\boldsymbol{\alpha}$ as:*

$$I_\nu(\mathcal{T}) = \hat{\boldsymbol{\alpha}}^T \Big(\big(\otimes_{i\in\mathcal{T}} X_{\bullet i} \big) \otimes \Big(\mathbf{1} + \sum_{\substack{\mathcal{B}\subseteq\mathcal{F}\setminus\mathcal{T} \\ |\mathcal{B}|\leq q-|\mathcal{T}|, j=u(\mathcal{B})}} \frac{1}{|\mathcal{B}|+1} \hat{X}_{\bullet j} \Big) \Big), \tag{15}$$

where $\mathbf{1}$ *is a vector of one and* $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha} \otimes \boldsymbol{y}$.

Given that $I_\nu(\{i\}) = \nu_i$, the Shapley value is a special case of the interaction index and the above formula. Theorem 2 facilitates the computation of the Shapley interaction index based on the dual SVM solution. However, the computation is still intense since the summation on the right-hand side of equation (15) makes the calculation complex. We now present a dynamic programming approach for computing equation (15). First, we define $\Omega_\mathcal{T}$ as:

$$\Omega_\mathcal{T} = \sum_{\substack{\mathcal{B}\subseteq\mathcal{F}\setminus\mathcal{T} \\ |\mathcal{B}|\leq q-|\mathcal{T}|, j=u(\mathcal{B})}} \frac{1}{|\mathcal{B}|+1} \hat{X}_{\bullet j}. \tag{16}$$

Computing $\Omega_\mathcal{T}$ efficiently leads to an efficient computation for the Shapley value and interaction index as in equation (27). To that end, we proposed a dynamic programming approach whose recursive formula being given as:

$$\hat{\Omega}^{\tilde{q},\tilde{d},o}_\mathcal{T} = \begin{cases} o^{-1}\mathbf{1} + \sum_{\substack{0<j<\tilde{d} \\ j\notin\mathcal{T}}} X_{\bullet j} & \text{if } \tilde{q} = 1 \\ \hat{\Omega}^{\tilde{d},\tilde{d},o}_\mathcal{T} & \text{if } \tilde{q} > \tilde{d} \\ \hat{\Omega}^{\tilde{q},\tilde{d}-1,o}_\mathcal{T} & \text{if } i = \tilde{d} \\ X_{\bullet\tilde{d}} \otimes \big(\hat{\Omega}^{\tilde{q}-1,\tilde{d}-1,o+1}_\mathcal{T}\big) + \hat{\Omega}^{\tilde{q},\tilde{d}-1,o}_\mathcal{T} & \text{otherwise.} \end{cases} \tag{17}$$

In equation (17), $\Omega^{\hat{q},\hat{d},o}_\mathcal{T}$ is the $\hat{q}$-additive summation of the first $\hat{d}$ features, and o is a positive integer responsible for generating the fractions in the Shapley value formula. The first case in equation (17) gives the solution when the interaction order is one, the second case is when the interaction order is bigger than the size of features at that step, the third case is a jump over the feature under explanation, and the last case is the main recursion and is based on the first $\hat{d} - 1$ features. Then, Ω could be computed by setting $\Omega = \hat{\Omega}^{q-|\mathcal{T}|,d,1}_\mathcal{T}$. The iterative dynamic programming algorithm to compute $\hat{\Omega}$ for the Shapley value is presented in Appendix E.

3.3 Overall Algorithm and Time Complexity

Given a training set, Algorithm 1 summarizes the steps for computing the Shapley values of features in the training set.

The proposed approach is very efficient for high-dimensional data. Training an SVM with the multilinear kernel is computationally as expensive as other kernel functions and its order is of $O(n)$. The number of operations required is d summations and $2d$ multiplication. For the $q-$additive kernel, we require $2qd$ summations and qd multiplications, which is still very efficient given that q is typically a small

Algorithm 1 Support Vector-based Shapley Value Learning (SVSVL)

Input $X \in R^{n\times d}, y, q \in N$
Computing the kernel matrix by Algorithm 2
Solving dual SVM and get the solution $\boldsymbol{\alpha}$
$SV \leftarrow zeros(d)$ # Shapley values array
for i=1:d **do**
  $SV[i]$ = Shapley value for feature i by Algorithm 3
end for
Output: SV

number. Having constructed the kernel matrix, solving the SVM has a complexity between $O(n^2)$ and $O(n^3)$. For some particular cases (i.e., least-square SVM), the solution is even obtained by solving a linear equation system.

For computing the $\hat{\Omega}$ of a continuous-valued feature, the complexity of Algorithm 3 is $O(qd)$, and in each iteration, we need $2qdn$ summations and multiplications. This number of operations could be simplified if we only consider the data points with the corresponding $\boldsymbol{\alpha}$ in the dual SVM nonzero (i.e., the support vectors). Let n_α be the number of support vectors, then the number of operations is reduced to $2qdn_\alpha$ summations and multiplications. Then, for computing the Shapley value given $\hat{\Omega}$, we need n_α summations and multiplications. In total, we need $2qdn_\alpha + n_\alpha$ summations and multiplications for each feature, given the dual SVM solution.

4 Experiments

This section presents some experiments comparing the proposed method for use in feature selection and explanation. All experiments were conducted on a MacBook Pro equipped with a 2.3 GHz 8-Core Intel Core i9 processor and 16GB of RAM. We compared our proposed method, SVSVL, against established benchmarks such as Lasso (Tibshirani 1996), REF (Guyon et al. 2002), mutual information (Vergara and Estévez 2014), and the ANOVA F-test (F-ANOVA) for feature selection, and Kernel SHAP, LIME (Ribeiro, Singh, and Guestrin 2016), Bivariate SHAP (Masoomi et al. 2021), and L2X (Chen et al. 2018a) for explanation (see Appendix A for further explanations of methods and related works).

4.1 Feature Selection

Synthesized Experiments We began by evaluating various feature selectors in a series of synthesized datasets, each designed with known influential features to ensure a precise evaluation. Each dataset consisted of 1000 samples and the objective was to identify the most influential features using different feature selection techniques. The effectiveness of each method was quantified by the average rank of the known influential features–the lower the average rank, the more effective the method. For all datasets, predictors X were sampled from a standard normal distribution, and the response variable is drawn as follows:

- The response variable Y was defined such that the conditional probability $P(Y = 1|X)$ was proportional to $e^{X_1X_2X_3} + e^{X_4X_5}$. This setup meant that only the first five features influenced the outcomes, with an ideal expected mean rank being three.
- For the second dataset, the response variable Y was modeled such that $P(Y = 1|X)$ depended on $e^{-100\sin(0.2X_1)+|X_2+X_3|+e^{-X_4}}$, highlighting the critical role of the first four features with an expected mean rank of 2.5 for the most important features.
- For the third dataset, the response variable Y expressed as $P(Y = 1|X) = e^{\sum_{i=1}^{4} X_i^2}$, placing emphasis on the first four features. The expected mean rank for the most influential features was again set at 2.5.

To mitigate random effects, we replicated the sample generation and feature detection process 100 times for each dataset, reporting both the mean and the standard deviation of the average rank of the most influential feature. Figure 1 displays the box plot of these ranks in the 100 datasets. The results indicated that while Lasso and REF, using a linear SVR approach, struggled to identify influential features due to their linear nature and inability to handle feature interactions, both SVSVL and mutual information successfully detected influential features, aligning more closely with the ground truth. This showcases the potential of SVSVL and mutual information in handling complex feature interactions within datasets. However, SVSVL has shown more promising performance across all synthesized datasets in detecting the most influential features of the synthesized datasets.

Real Datasets We also evaluate various feature selectors across six real-world datasets: *housing*, *abalone*, *query*, *miles per gallon*, *breast cancer*, and *diabetes*. Detailed descriptions of these datasets are provided in Appendix F. In our analysis, we apply each feature selector to the datasets, ranking the features based on their importance. Following this, features are incrementally added to train a random forest, allowing us to observe the impact of each feature on the model's performance. We anticipate a marked improvement in performance (or a decrease in error) initially when the most influential features are added, with diminishing returns as less important features are included towards to the end of adding features.

Figure 2 illustrates the performance of the random forest against the number of features (added according to their importance for each method) from the six datasets. The results highlight the competitive performance of SVSVL, which excels particularly in the *querry*, *miles per gallon* and *diabetes* datasets, and remains competitive in *housing*, *breast cancer*, and *abalon* to the best-performing feature selectors.

4.2 Experiments on Explainability

We also scrutinize the effectiveness of SVSVL in real-world scenarios, particularly focusing on its comparative performance in local fidelity and execution time against established explainable methods. Our comparative study involves two models: a bidirectional LSTM model designed for sentiment analysis in IMDB reviews and a Random Forest model comprising 50 trees, honed on the Boston housing dataset. SVSVL is compared against other methods.

We use two metrics for comparison: the execution time and the fidelity score, which is the difference between the pre-

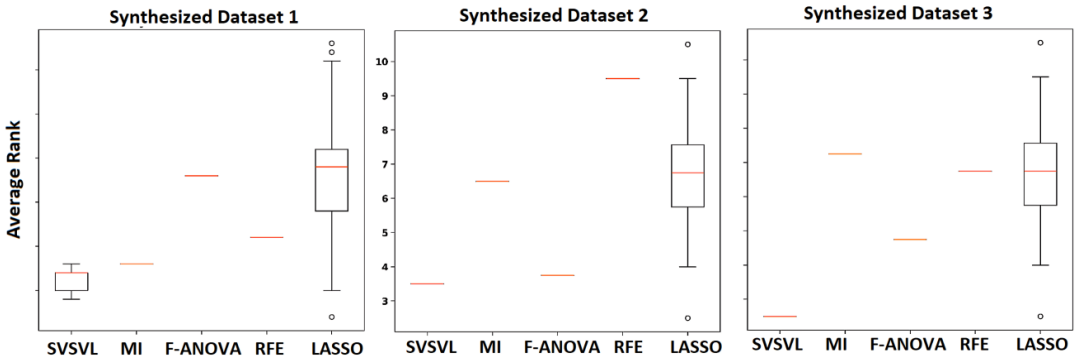

Figure 1: The average of the average rank of influential features on three synthesized experiments.

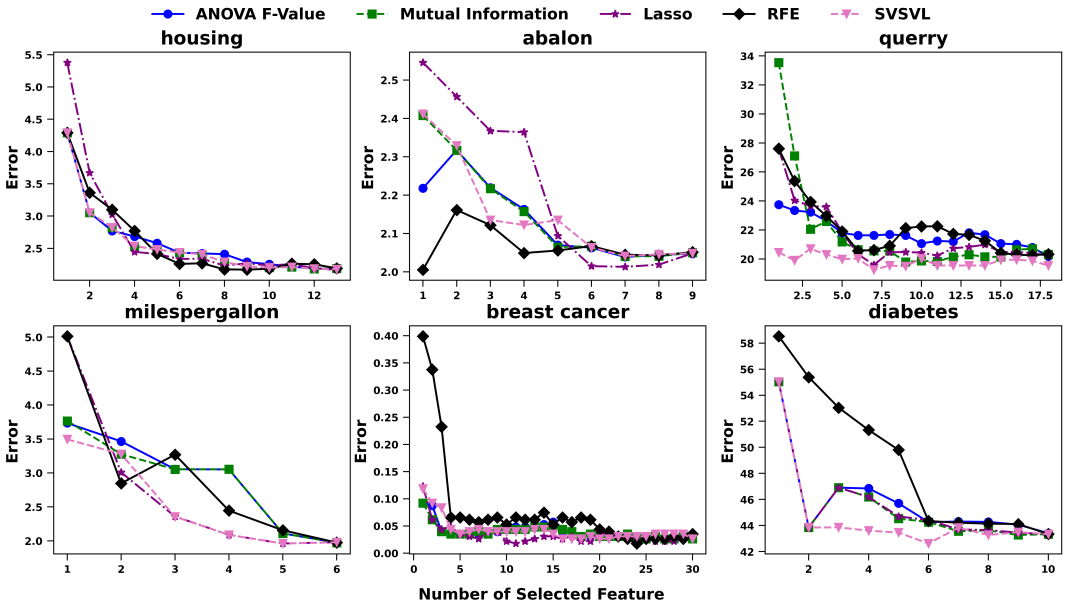

Figure 2: The performance of feature selection methods on six real datasets.

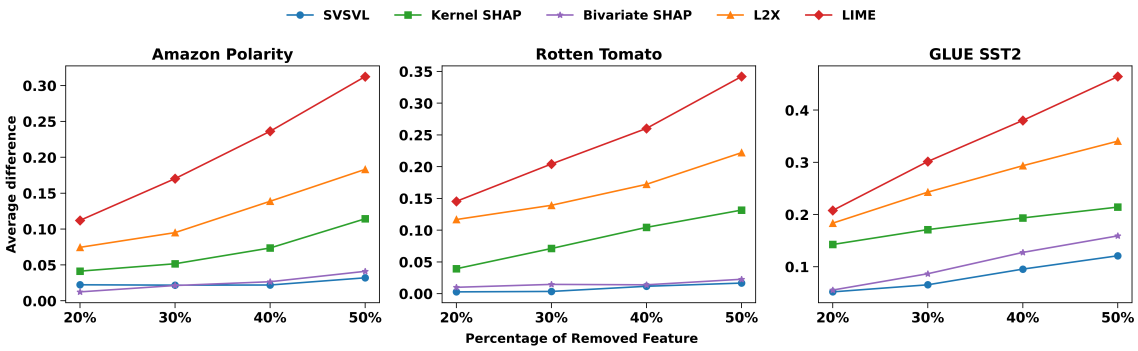

Figure 3: Deviation from original prediction by masking the deemed redundant features.

diction of the original model with that of the local surrogate model. We use the mean square error (MSE) to gauge the fidelity of an explainable model. Table 1 shows the MSE between local explainers and the original predictions of the corresponding models, as well as the average execution time for an explanation in 100 different experiments. The discernible trends from the table underscore a conspicuous superiority of models, like SVSVL, that incorporate interactions among

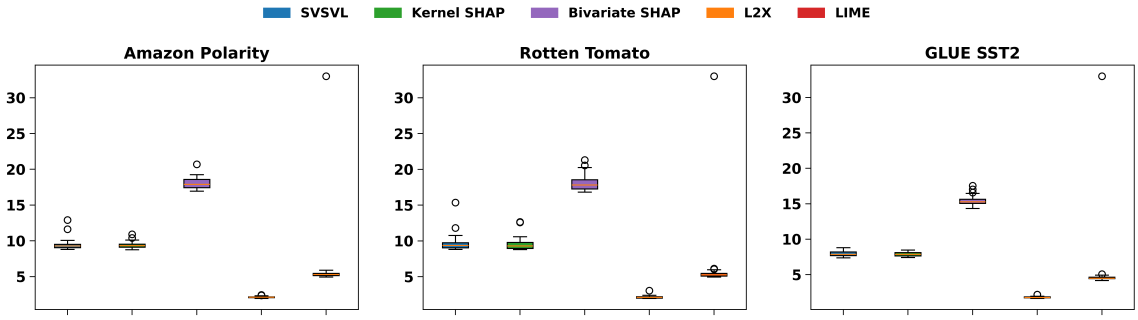

Figure 4: The execution time of methods for explaining the sentiment analysis of 50 reviews.

Method	MSE		Average time (s)	
	LSTM	RF	LSTM	RF
LIME	0.12 ± 0.02	$0.003e \pm 0.0$	123	2
SHAP	0.11 ± 0.03	0.004 ± 0.0	310	29
L2X	0.10 ± 0.01	0.002 ± 0.01	73	2
BiSHAP	0.07 ± 0.02	0.004 ± 0.01	120	12
SVSVL	0.05 ± 0.01	0.004 ± 0.0	129	3

Table 1: The comparison of explainable methods based on fidelity of the local explanation and the average execution time in seconds (rounded to the first integer).

features, exhibiting significantly reduced MSE in surrogate models, especially evident in the LSTM model. This enhancement in local fidelity is attributable to their nonlinear nature, enabling a more nuanced replication of the behaviors of the models under explanation. Interestingly, the differences in explaining the Random Forest model were comparatively marginal, reflecting the inherent simplicity of the model, where linear methods can be nearly as effective as their nonlinear counterparts. In this context, SVSVL stands out by achieving the lowest MSE, highlighting its ability to account for all possible interactions among features. Some interactions detected on IMDB reviews are shown in Appendix H.

Delving into the execution time reveals another layer of competitive advantage for SVSVL. It showcases a time efficiency comparable to LIME, which does not account for interactions, and remarkably outperforms SHAP. The proximity of SVSVL's average execution time to that of LIME further amplifies its competitive stance in the domain of explainable AI. Meanwhile, L2X exhibits impressive speed, attributed to its single training stage for providing explanations across the entire dataset, emphasizing the diverse range of execution efficiencies within the examined explainable methods. In conclusion, the amalgamation of superior local fidelity and competitive execution time positions SVSVL as a robust candidate in the realm of explainable methods, especially when intricate interactions are pivotal.

We extend our analysis of sentiment classification by investigating the impact of masking words deemed least important by various explainability methods on the predictions of the bidirectional LSTM model. Our study focuses on three textual datasets, applying explainable methods to 50 reviews from each. To evaluate these methods, we identify influential words in each review and mask those considered least impactful, hypothesizing that such masking will not significantly alter the model's predictions. Figure 3 illustrates the effect of masking different percentages of words, with the x-axis representing the percentage masked. The results demonstrate that SVSVL excels by accurately identifying both the most and least influential words, resulting in stable predictions even when large portions of reviews are masked. Bivariate SHAP also shows strong performance, closely followed by kernel SHAP, while L2X and LIME prove less effective in identifying critical words. In addition, we evaluate these methods on the basis of the time efficiency of generating explanations. As shown in Figure 4, SVSVL's execution time is comparable to Kernel SHAP and outperforms Bivariate SHAP, despite solving a problem with exponentially many parameters. In contrast, although LIME and L2X are faster, their results are less reliable, as reflected in the word removal analysis.

5 Conclusion and Discussion

This paper presents a novel approach to computing the Shapley value and interaction index in supervised learning methods through a multilinear game model, employing support vector machines (SVMs) and dynamic programming techniques. The core innovations include a multilinear kernel for SVMs that captures all feature interactions and is suitable for feature selection and local explanation. Also, a new formulation of the Shapley value using dual SVMs is derived that enhances computational efficiency.

Despite its strengths, the method exhibits certain drawbacks, such as the inability to consider higher-order statistical moments of features in feature selection. In addition, while the method integrates seamlessly with L_2 regularization in SVMs, it does not support L_1 regularization due to limitations in applying the kernel trick directly, which restricts its full potential. Future research will aim to overcome these challenges, possibly by developing alternative interaction functions and more efficient algorithms to pinpoint essential feature interactions within the dual SVM framework.

Acknowledgments

This publication is part of the project PROXCAI, which is financed by the Dutch Research Council (NWO) under the grant *KIVI.2019.003*.

References

Borkotokey, S.; Hazarika, P.; and Mesiar, R. 2015. Fuzzy Bi-cooperative games in multilinear extension form. *Fuzzy Sets and Systems*, 259: 44–55.

Chen, J.; Song, L.; Wainwright, M.; and Jordan, M. 2018a. Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, 883–892. PMLR.

Chen, J.; Song, L.; Wainwright, M. J.; and Jordan, M. I. 2018b. L-shapley and c-shapley: Efficient model interpretation for structured data. *arXiv preprint arXiv:1808.02610*.

Cortes, C.; and Vapnik, V. 1995. Support-vector networks. *Machine learning*, 20(3): 273–297.

Covert, I.; and Lee, S.-I. 2021. Improving kernelshap: Practical shapley value estimation using linear regression. In *International Conference on Artificial Intelligence and Statistics*, 3457–3465. PMLR.

Covert, I.; Lundberg, S. M.; and Lee, S.-I. 2020. Understanding global feature contributions with additive importance measures. *Advances in Neural Information Processing Systems*, 33: 17212–17223.

Datta, A.; Sen, S.; and Zick, Y. 2016. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE symposium on security and privacy (SP)*, 598–617. IEEE.

Gevaert, A.; and Saeys, Y. 2022. PDD-SHAP: fast approximations for Shapley values using functional decomposition. *arXiv preprint arXiv:2208.12595*.

Grabisch, M. 1996. The representation of importance and interaction of features by fuzzy measures. *Pattern Recognition Letters*, 17(6): 567–575.

Grabisch, M.; and Roubens, M. 1999. An axiomatic approach to the concept of interaction among players in cooperative games. *International Journal of game theory*, 28: 547–565.

Guyon, I.; Weston, J.; Barnhill, S.; and Vapnik, V. 2002. Gene selection for cancer classification using support vector machines. *Machine learning*, 46: 389–422.

Hiabu, M.; Meyer, J. T.; and Wright, M. N. 2023. Unifying local and global model explanations by functional decomposition of low dimensional structures. In *International Conference on Artificial Intelligence and Statistics*, 7040–7060. PMLR.

Jones, M. A.; and Wilson, J. M. 2010. Multilinear extensions and values for multichoice games. *Mathematical Methods of Operations Research*, 72: 145–169.

Koller, D.; Sahami, M.; et al. 1996. Toward optimal feature selection. In *ICML*, volume 96, 292.

Lipovetsky, S.; and Conklin, M. 2001. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17(4): 319–330.

Lundberg, S. M.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Masoomi, A.; Hill, D.; Xu, Z.; Hersh, C. P.; Silverman, E. K.; Castaldi, P. J.; Ioannidis, S.; and Dy, J. 2021. Explanations of Black-Box Models based on Directional Feature Interactions. In *International Conference on Learning Representations*.

Mitchell, R.; Cooper, J.; Frank, E.; and Holmes, G. 2022. Sampling permutations for shapley value estimation.

Owen, G. 1972. Multilinear extensions of games. *Management Science*, 18(5-part-2): 64–79.

Owen, G. 1988. Multilinear extensions of games. *The Shapley Value. Essays in Honor of Lloyd S. Shapley*, 139–151.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144.

Shapley, L. S. 1953. A value for n-person games.

Štrumbelj, E.; and Kononenko, I. 2014. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41: 647–665.

Sundararajan, M.; Dhamdhere, K.; and Agarwal, A. 2020. The shapley taylor interaction index. In *International conference on machine learning*, 9259–9268. PMLR.

Thomas, M.; and Joy, A. T. 2006. *Elements of information theory*. Wiley-Interscience.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1): 267–288.

Tripathi, S.; Hemachandra, N.; and Trivedi, P. 2020. Interpretable feature subset selection: A Shapley value based approach. In *2020 IEEE International Conference on Big Data (Big Data)*, 5463–5472. IEEE.

Tsai, C.-P.; Yeh, C.-K.; and Ravikumar, P. 2023. Faith-shap: The faithful shapley interaction index. *Journal of Machine Learning Research*, 24(94): 1–42.

Vergara, J. R.; and Estévez, P. A. 2014. A review of feature selection methods based on mutual information. *Neural computing and applications*, 24: 175–186.

von Luxburg, U.; and Bordt, S. 2023. From Shapley Values to Generalized Additive Models and back.