Weighted Tanimoto Extreme Learning Machine with Case Study in Drug Discovery

Wojciech Marian Czarnecki

***Abstract*—Machine learning methods are becoming more and more popular in the field of computer-aided drug design. The specific data characteristic, including sparse, binary representation as well as noisy, imbalanced datasets, presents a challenging binary classification problem. Currently, two of the most successful models in such tasks are the Support Vector Machine (SVM) and Random Forest (RF). In this paper, we introduce a Weighted Tanimoto Extreme Learning Machine (T-WELM), an extremely simple and fast method for predicting chemical compound biological activity and possibly other data with discrete, binary representation. We show some theoretical properties of the proposed model including the ability to learn arbitrary sets of examples. Further analysis shows numerous advantages of T-WELM over SVMs, RFs and traditional Extreme Learning Machines (ELM) in this particular task. Experiments performed on 40 large datasets of thousands of chemical compounds show that T-WELMs achieve much better classification results and are at the same time faster in terms of both training time and further classification than both ELM models and other state-of-the-art methods in the field.**

***Index Terms*—Extreme Learning Machine, Drug design, Tanimoto activation function, Fast training, Classification.**

I. Introduction

COMPUTER-AIDED drug design has become a very popular technique for speeding up the process of finding new biologically active compounds. Despite the existence of huge databases of known drugs as well as numerous approaches to predicting the activity of different molecules, this problem is still very challenging and even good generation of training sets [1], data representation [2] and evaluation metrics [3] are open problems. One of the recognized bottlenecks in this field is the size of the datasets required to process. There are hundreds of thousands of compounds in databases and numerous proteins for which we would like to know the activity. Furthermore, the resulting datasets are very noisy and highly imbalanced.

The main aim of this paper is to introduce a new machine learning model, the Tanimoto Extreme Learning Machine (T-ELM), which is well suited for this type of problem, has a strong theoretical background, achieves good classification results, and is extremely fast (in terms of both training and evaluation times). T-ELM is a modification of the Extreme Learning Machine (ELM) [4] customized for the drug discovery problem. We show how the proposed model differs from the classical ELM and why it is better for the specific type of data occurring in the cheminformatics domains. In particular, the T-ELM differs from ELM in two important aspects. First, it selects random weights from the training samples instead of continuous probability distribution. Second, it projects data into a much richer feature space (see subsection B of Section IV). As a result, T-ELM uses only sparse, binary weights in the hidden layer, which increases the speed of calculation significantly (in particular, such an approach can simplify hardware implementation of ELMs [5]) while at the same time requiring careful theoretical analysis. We perform a detailed empirical comparison based on 40 datasets from 8 different protein targets represented by 5 different fingerprints with both traditional ELMs and with state-of-the-art methods in the field.

The paper is structured as follows. First we provide a basic introduction to the computer-aided drug discovery process with a focus on the fingerprints and class imbalance. Then, we recall some basic information regarding ELMs and proceed to the description of the proposed model. We show two ways of looking at T-ELMs and prove some theoretical properties (including the perfect learning ability). A practical section follows, including an efficient training method for Weighted ELMs. Finally we perform a detailed evaluation and conclude with discussion.

II. Computer-aided drug discovery

The increasing amount of information in the fields of bio- and cheminformatics has made tools addressing data analysis more and more popular. The most widely used approaches adopt machine learning models for making predictions regarding chemical compounds. These methods are applied both to evaluate the potential biological activity and to provide predictions about the physicochemical and pharmacokinetic properties of chemical structures.

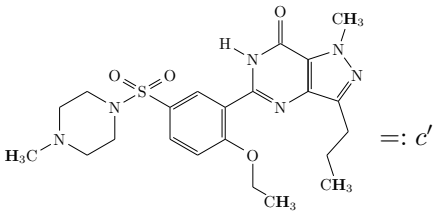
A popular model used in ligand-based virtual screening is the Support Vector Machine (SVM) [6]–[9]. Many of the existing solutions use the Gaussian kernel [10], [11] with rare attempts to exploit the vast diversity of potential kernel functions including graph-based approaches [12], [13]. In this paper we show that one can achieve better classification results using a much simpler and faster model.

A. Fingerprints

Typical machine learning methods require data to be a subset of the real space $\mathbb{R}^d$. In other words we need some fixed-length numeric representation of each object that we want to work with. In case of drug design problems our objects are

WM Czarnecki is with the Faculty of Mathematics and Computer Science, Jagiellonian University, Krakow, Poland, e-mail: wojciech.czarnecki@uj.edu.pl .

Manuscript received April 19, 2005; revised December 27, 2012.



	i	1	2	3	4	…	d
predicates	$\phi_i(c)$	$\subset c$	$\subset c$	$\subset c$	$\lvert\mathrm{N}\rvert_c \geq 3$	…	$\subset c$
fingerprint	$\varphi(c')$	1	0	1	1	…	1

Fig. 1. Sample fingerprint of the chemical molecule c'. $|A|_c$ denotes the number of atoms/substructures A in c, so in particular $A \subset c \iff |A|_c \geq 1$.

chemical compounds, which cannot be directly transformed into such structures. As a result, researchers proposed multiple fingerprints (data representations) across years [14]–[17]. Most of them are a predefined set of d predicates $\phi_i(\cdot)$ (also called descriptors), which result in encoding the compound in the finite space $\{0,1\}^d$. For a given compound $c \in \mathcal{C}$, we can define embedding φ:

$$\varphi : \mathcal{C} \ni c \to [1_{\phi_1(c)}, 1_{\phi_2(c)}, \ldots, 1_{\phi_d(c)}]^T \in \{0,1\}^d \subset \mathbb{R}^d,$$

where $1_{\phi_i(c)}$ equals 1 if $\phi_i(c)$ is true and 0 otherwise. Predicates can be of various types, including those with some particular atoms, substructures or easily verifiable features (such as "the total mass is above some threshold"), see Fig. 1 for an example.

There are two important aspects of such fingerprints representation that affect the whole classification process. The first is the ambiguity of the representation. There might be (and often are) compounds of two different activities (two different labels) with the exact same fingerprint. This makes them completely undistinguishable and as a result it is impossible to create a perfect model. The second is representation sparsity, which varies across different fingerprints; however, it is in general true that very few $\phi_i(c)$ are true for a fixed c. This leads to possible speed up (further investigated in other sections) and also requires some custom methods of similarity measuring, as discussed in Section IV.

B. Class imbalance

The resulting classification problem is highly imbalanced for two main reasons. Statistically speaking, for a given protein, almost none of the compounds (that are possible to synthesize) are active. However, at the same time, most of the published (in both databases and papers) compounds are active (positive samples). This is a result of the "positive results bias" present in every field. Scientists rarely publish negative results (confirmed inactivity of the given compound), so although they are very common, we do not have a reliable source of such information. Moreover, these reported inactive compounds are very similar to the active compounds violating the true class distributions. This causes problems with the generation of valid training/test sets for machine learning techniques. Proposed methods of dealing with such phenomenon vary from working with incorrect class balance (true active and true inactive compounds, which result in small classification problems) to artificial generation of inactive compounds (such as Directory of Useful Decoys – DUD [18]). In this work we follow the second path, to preserve valid class balancing and to be able to work with large parts of chemical compounds databases. Such an approach is accepted in computational chemistry discipline and often applied in real virtual screening based on machine learning methods [19]–[21].

III. Extreme Learning Machines

ELMs are relatively new models introduced by Huang et al. [4] that are based on the idea that single layer feed forward neural networks (SLFN) can be trained without the iterative process by performing linear regression on data mapped through random, nonlinear projection (random hidden neurons). Strictly speaking, basic ELM architecture consists of d input neurons connected with each input space dimension, which are fully connected with h hidden neurons by the set of weights $\mathbf{w}_j$ (selected randomly from some arbitrary distribution) and set of biases b_j (also randomly selected). Given some generalized nonlinear activation function $\mathbf{G}$, one can express the hidden neurons activation matrix $\mathbf{H}$ for the whole training set $\mathbf{x}_i \in \mathbb{R}^d, \mathbf{t}_i \in \{-1,+1\}$ (we consider a binary classification but in general $\mathbf{t}_i \in \mathbb{R}^p$) as

$$\mathbf{H}_{ij} = \mathbf{G}(\mathbf{x}_i, \mathbf{w}_j, b_j). \tag{1}$$

If we denote the weights between the hidden layer and output neurons by β it is easy to show [22], that putting

$$\beta = \mathbf{H}^\dagger \mathbf{T},$$

gives the best solution in terms of the mean squared error of the regression:

$$\|\mathbf{H}\beta - \mathbf{T}\|_2^2 = \left\|\mathbf{H}(\mathbf{H}^\dagger\mathbf{T}) - \mathbf{T}\right\|_2^2 = \min_{\alpha \in \mathbb{R}^{h\times p}} \|\mathbf{H}\alpha - \mathbf{T}\|_2^2,$$

where $\mathbf{H}^\dagger$ denotes the Moore-Penrose pseudoinverse of matrix $\mathbf{H}$.

Final classification of the new point $\mathbf{x}$ can now be performed analogously by classifying according to

$$cl(\mathbf{x}) = \textsc{sign}(\begin{bmatrix}\mathbf{G}(\mathbf{x},\mathbf{w}_1,b_1) & \ldots & \mathbf{G}(\mathbf{x},\mathbf{w}_d,b_d)\end{bmatrix}\beta).$$

Such an approach has numerous advantages over similar methods including SVMs and traditional Neural Networks, such as:

- rapid training (even orders of magnitude faster than competitive methods),
- simple implementation, consisting of basic operations (no complex optimization routines),
- generalization capabilities comparable to state-of-the-art methods in numerous applications [23]–[27].

IV. TANIMOTO ELM

In this section we introduce the Tanimoto ELM, a classifier constructed especially to work with sets classification, based on the ELM concepts. We first show how the proposed method is derived from the classical set similarity measures and then show the alternative derivation as the special case of generalized SLFN.

A. From the set similarity network perspective

Most of the machine learning models require real valued data ($\mathbf{X} \in \mathbb{R}^{d\times N}$), although many real life datasets are not of such form. A common alternative data representation is sets of some features/attributes. For instance, in cheminformatics data (compounds) are often represented as sets of some predicates constituting binary fingerprints. Consequently it is natural to define an embedding ϕ:

$$\phi : \mathcal{C} \ni c \to \{i : \phi_i(c)\} \subset \{1, 2, \ldots, d\} \subset \mathbb{N}.$$

Although such data can be easily embedded in $\mathbb{R}^d$ (which is often performed in practice [6]–[9]) we show that exploiting this characteristic leads to significant improvement in the classification process.

First let us recall a well known coefficient for measuring the similarity of the two sets. For any two sets A and A', such that at least one of them is not empty, the Tanimoto coefficient (also known as the Jaccard coefficient) is defined as:

$$J(A, A') = \frac{|A \cap A'|}{|A \cup A'|}, \tag{2}$$

and $J(\emptyset, \emptyset) = 0$. This measures the ratio of features that two data-points have in common and the total amount of distinct features reported for at least one of them.

For easier further manipulation, we use the fact that

$$|A \cup A'| = |A| + |A'| - |A \cap A'|,$$

to rewrite (2) in the form

$$J(A, A') = \frac{|A \cap A'|}{|A| + |A'| - |A \cap A'|}.$$

Let us now assume that we have h predefined compounds c_i which are characteristic in terms of some interesting property (for example, biological activity), and corresponding sets $W_i = \phi(c_i)$, so we will be able to predict a new point label according to its similarity to those W_i.

$$\begin{aligned} cl(c) &= \text{SIGN}\left(\sum_{i=1}^{h} \beta_i J(\phi(c_i), \phi(c))\right) \\ &= \text{SIGN}\left(\sum_{i=1}^{h} \beta_i J(W_i, \phi(c))\right), \end{aligned}$$

where $\beta_i \in \mathbb{R}$ are some (possibly negative) weights associated with a given c_i.

More generally, for given sets X_i and our predefined sets W_i, we build a pairwise set similarity matrix

$$\mathbf{H}_{ij} = J(X_i, W_j) = \frac{|X_i \cap W_j|}{|X_i| + |W_j| - |X_i \cap W_j|}, \tag{3}$$

and (in the binary case) classify X_i according to the sign of

$$\text{T-ELM}_{W,\beta}(X) = \mathbf{H}\beta.$$

If we are given our sets X_i, W_i in forms of binary matrices $\mathbf{X}, \mathbf{W}$, we may put

$$\text{T}(\mathbf{X}_i, \mathbf{W}_j) = \frac{\mathbf{X}_i^T \mathbf{W}_j}{\mathbb{1}\mathbf{X}_i + \mathbb{1}\mathbf{W}_j - \mathbf{X}_i^T \mathbf{W}_j}, \tag{4}$$

where $\mathbb{1}$ is a matrix of ones of appropriate dimension and A_i denotes ith column of matrix A. From (3) and (4), we obtain

$$\mathbf{H}_{ij} = J(X_i, W_j) = \text{T}(\mathbf{X}_i, \mathbf{W}_j),$$

and consequently

$$\mathbf{H} = \frac{\mathbf{X}^T \mathbf{W}}{\mathbb{1}\mathbf{X} + \mathbb{1}\mathbf{W} - \mathbf{X}^T \mathbf{W}}. \tag{5}$$

Now let us think about SLFN consisting of hidden neurons computing the Tanimoto coefficient between the input signal and the set (encoded as a binary vector) in a particular neuron. We will refer to such units as *Tanimoto neurons*. We can now define a Tanimoto ELM as an ELM with Tanimoto neurons in the hidden layer (see Fig. 2).

$$\text{T-ELM}_{\mathbf{W},\beta}(\mathbf{X}) = \left(\frac{\mathbf{X}^T \mathbf{W}}{\mathbb{1}\mathbf{X} + \mathbb{1}\mathbf{W} - \mathbf{X}^T \mathbf{W}}\right)\beta,$$

where $\mathbf{W} \in \{0, 1\}^{d\times h}$ and $\mathbf{X} \in \{0, 1\}^{d\times N}$. The very same equation can be used for $\mathbf{W} \in [0, 1]^{d\times h}$ but as we show in Section VI and in Theorem 1, it is preferable to use $\mathbf{W} \subset \mathbf{X}$[1]. Thus, as opposed to the classical ELM, we are selecting hidden neurons randomly from the training set instead of some arbitrary continuous distribution. This way, we are ensuring, on one hand, that the weights used are binary and sparse and, on the other, that they correctly span the (possibly much degenerated) sample space. This is conceptually similar to taking orthogonal features [28] which ensures better spanning of the whole $\mathbb{R}^d$ but is focused on a very small subset of the input space, which is actually filled with data points.

B. From the generalized SLFN perspective

One can see the proposed method as a generalized SLFN with a non-neuron like unbiased activation function

$$\begin{aligned} \mathbf{G}(\mathbf{x}, \mathbf{w}, b) &= \mathbf{G}(\mathbf{x}, \mathbf{w}) \\ &= \frac{\langle \mathbf{w}, \mathbf{x} \rangle}{\mathbb{1}\mathbf{x} + \mathbb{1}\mathbf{w} - \langle \mathbf{w}, \mathbf{x} \rangle} \\ &= \frac{\langle \mathbf{w}, \mathbf{x} \rangle}{\|\mathbf{w}\|_1 + \|\mathbf{x}\|_1 - \langle \mathbf{w}, \mathbf{x} \rangle}. \end{aligned} \tag{6}$$

We will refer to this function as the *Tanimoto activation function*. First, let us show some of its basic properties, which are of interest to ELM theory [22], [29].

Remark 1. *The Tanimoto activation function is nonconstant, bounded, continuous and infinitely differentiable, for* $\boldsymbol{x}, \boldsymbol{w} \in [0, 1]^d$*, such that* $\boldsymbol{x} + \boldsymbol{w} \neq 0$.

[1] $A \subset A'$ means that $\forall_i \exists_j \forall_k A_{ki} = A'_{kj}$.

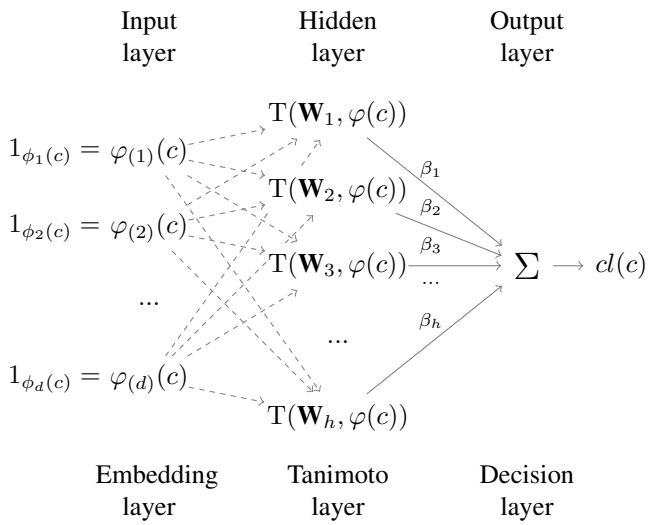


Fig. 2. T-ELM as a neural network with h hidden Tanimoto neurons for the classification of the compound c, where $\mathbf{W}_i = \varphi(c_i)$. All weights are either randomly selected (dashed) or are the result of closed-form optimization (solid).

Proof. The continuity is a direct consequence of the continuity of the scalar product and division. The only point of discontinuity is when the denominator is zero, which can only occur if both $\mathbf{w}$ and $\mathbf{x}$ are zero vectors, which is beyond functions domain[2]. Similarly, the Tanimoto activation function is C^∞ as it is a composition of C^∞ functions. □

Now we proceed to the main theoretical result of our paper. We will show, that under simple restrictions, a Tanimoto ELM with N hidden neurons can perfectly learn any set of N binary vectors using only sparse, binary weights.

Theorem 1. *Given arbitrary N distinct samples $\boldsymbol{x}_i \in \{0,1\}^d$ with fixed sparseness $\|\boldsymbol{x}_i\|_1 = m$, some labeling $\mathbf{t}_i \in \mathbb{R}^p$ and hidden layer of N distinct Tanimoto neurons $\boldsymbol{w}_i$ selected from the training set, Tanimoto ELM hidden layer activation matrix $\mathbf{H}$ is invertible and $\|\mathbf{H}\beta - \mathbf{T}\| = 0$.*

Proof. As $\mathbf{w}_i$ are selected from $\mathbf{x}_i$, without loss of generality we can assume that $\mathbf{w}_i = \mathbf{x}_i$ for all i. Thus, from (1) and (6), activation matrix $\mathbf{H}$ is of form

$$\mathbf{H}_{ij} = \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{\|\mathbf{x}_i\|_1 + \|\mathbf{x}_j\|_1 - \langle \mathbf{x}_i, \mathbf{x}_j \rangle}. \tag{7}$$

Let us notice that for binary vectors $\mathbf{x}_i, \mathbf{x}_j$, we have

$$\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 = \|\mathbf{x}_i - \mathbf{x}_j\|_1 = \|\mathbf{x}_i\|_1 + \|\mathbf{x}_j\|_1 - 2\langle \mathbf{x}_i, \mathbf{x}_j \rangle,$$

thus, using the assumption $\|\mathbf{x}\|_1 = m$ in (7), we obtain

$$\begin{aligned}\mathbf{H}_{ij} &= \frac{2m - \|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2m + \|\mathbf{x}_i - \mathbf{x}_j\|_2^2} \\ &= \mathbf{g}(\|\mathbf{x}_i - \mathbf{x}_j\|_2^2),\end{aligned}$$

[2]One could also redefine the function to be continuous in zero by simply adding small constant $\varepsilon > 0$ to both numerator and denominator $\mathbf{G}(\mathbf{x}, \mathbf{w}, b) = (\langle \mathbf{w}, \mathbf{x} \rangle + \varepsilon)/(\|\mathbf{w}\|_1 + \|\mathbf{x}\|_1 - \langle \mathbf{w}, \mathbf{x} \rangle + \varepsilon)$.

for $\mathbf{g} : \mathbb{R} \ni x \to \frac{a-x}{a+x} \in \mathbb{R}$ and $a = 2m > x \geq 0$. We obtain that $\mathbf{g}$ is a function of class $C^\infty(0, \infty)$ being positive in the interval $[0, 2m)$ (while the arguments are always in $[0, m]$ due to the sparseness restriction). Furthermore, it is easy to show that it is completely monotonic on $(0, \infty)$, meaning that

$$(-1)^l \frac{\partial^l \mathbf{g}}{\partial x^l}(x) \geq 0, \forall x \in (0, \infty), \forall l \in \mathbb{N},$$

which means that, according to Schoenberg's theorem [30], [31], $\mathbf{H}$ is strictly positive definite and thus invertible. As a result, for $\beta = \mathbf{H}^\dagger \mathbf{T} = \mathbf{H}^{-1}\mathbf{T}$, we have $\|\mathbf{H}\beta - \mathbf{T}\| = 0$.

□

We showed that, under some restrictions even sparse, binary weights can be used for a model construction with perfect learning guarantees. However, in general, this type of model is much less flexible than its continuous counterparts. It seems natural to pose the question as to whether it is possible to present a general discrete ELM with high probability of perfect learning.

Open problem. *Let us assume that we are given an ELM with discrete weights and the number of hidden neurons equal to the number of training samples selected randomly. Is it possible to prove that under reasonable conditions for activation function $\mathbf{G}$ and data dimension/sparsity, this ELM has an invertible hidden layer activation matrix with high[3] probability?*

Even for fixed sparsity $m = 2$, we can easily find a matrix

$$\mathbf{X} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}^T,$$

for which there is a significant probability of generating $\mathbf{W}$ for which the $\mathbf{H}$ matrix is not invertible. In fact, above $\mathbf{X}$, which is linearly separable, can be perfectly learned even by an unbiased perceptron.

C. Comparison with standard activation functions

The Tanimoto activation function has a nice property of scaling well with data sparsity. If we again assume fixed sparsity of both data and hidden neurons m, the Tanimoto activation function viewed as a function of the Euclidean distance $\|\mathbf{w} - \mathbf{x}\|_2^2$ is convex, monotonically decreasing from 1 (for $\|\mathbf{w} - \mathbf{x}\|_2^2 = 0$) to $1/3$ (for $\|\mathbf{w} - \mathbf{x}\|_2^2 = m$). On the other hand, the sigmoid function (expressed as a function of the same parameter) becomes a concave function nearly constantly equal to 1 with $m \to \infty$. Symmetrically, the Gaussian activation function with fixed variance becomes convex, nearly constantly equal to 0 under the same conditions. Fig. 3 shows plots of these functions for $m = 4$ and $m = 100$.

[3]It remains also open what would exactly "high" mean.

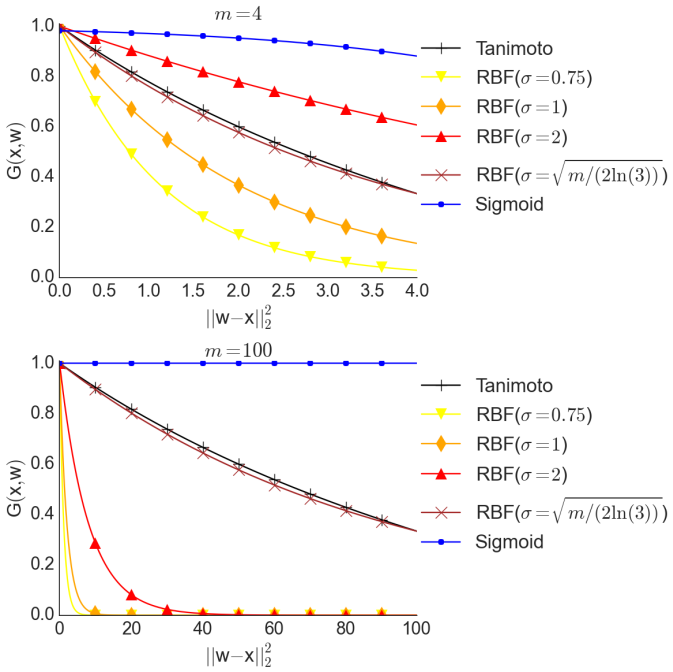


Fig. 3. Comparison of the activation functions as functions of a squared Euclidean distance on fixed sparsity m.

For further intuition we also show that the Tanimoto activation function is very similar (slightly higher) to a Gaussian activation function with standard deviation σ equal to $\sqrt{m/(2\ln(3))}$, which ensures that $\exp(-\|\mathbf{w}-\mathbf{x}\|_2^2/(2\sigma^2)) = 1/3$. This is, however, only achievable for fixed sparsity m, which in practice is a data dependent value; therefore, construction of the analogous RBF like activation function would require introduction of the $\mathbf{w}$ and $\mathbf{x}$ 1-norms to the equation.

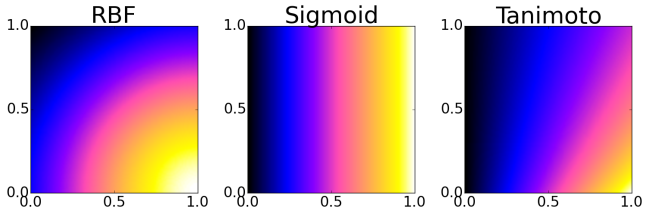


Fig. 4. Visualization of the RBF, Sigmoid and Tanimoto activation functions for fixed $\mathbf{w} = [1,0]^T$ for the $\mathbf{x} \in [0,1]^2$.

It is worth noting that, as opposed to the sigmoid, the Tanimoto activation function is not constant in the subspace spanned by such base vectors $\mathbf{e}_i$ such that $\mathbf{w}_i = 0$. In the set theory language, if we have a set A_1 and some element $a \notin A_1$, using the sigmoid activation function would result in the same "similarity" assigned to other set A_2 regardless of $a \in A_2$ or $a \notin A_2$. This holds for all functions of form $\mathbf{g}(\langle \mathbf{w}, \mathbf{x} \rangle)$.

The Tanimoto activation function is also asymmetric in regard to comparing "how different" two sets are. For given sets A_1, A_2, A_3 such that $A_1 \neq A_2 \neq A_3$, $A_1 \cup \{a_1\} = A_2$ and $A_1 \setminus \{a_2\} = A_3$, A_2 is considered more similar to A_1 than A_3 to A_1. RBF, on the other hand, is fully symmetric in this manner and considers A_3 and A_1 to be as similar as A_2 and A_1. This holds for all activation functions of form $\mathbf{g}(\|\mathbf{w}-\mathbf{x}\|_2^2)$.

Consequently, the Tanimoto activation function is able to capture much more complex relations (similarities) between samples. We argue that for binary representation it is important to be able to distinguish between different types of similarity, so the classifier trained on top of the extreme hidden layer has more freedom in defining a hyperplane (the space of possible hypotheses is richer).

Binary samples (points in discrete space) behave significantly differently from points in continuous spaces. In particular, there are only 2^d distinct point representations in d-dimensional binary space, while there are infinitely many of them in any dimension of continuous space (even in a 1-dimensional real line). We can think about points in $\{0,1\}^d$ space as points in $\mathbb{R}^d$, simply placed in the vertices of the hypercube; however, such an approach can lead to unexpected consequences. Although most of the activation functions can be used for such data (such as sigmoids and RBFs), once applied to the very small, finite subset of their domain, they can exhibit degenerative behavior. For example, some function $\mathbf{G}(\mathbf{x}, \mathbf{w}, b)$ might always return a constant value when applied to $\mathbf{x}$ from the hypercube's vertices while resulting in a valid similarity measure in the rest of the $\mathbb{R}^d$. When dataset X is for the most part in $\mathbb{R}^d \setminus \{0,1\}^d$, this degenerative behavior has negligible importance, and thus in most cases, $\mathbf{G}$ works well. However, in our binary space, this function embeds data in a single point; thus, no learning algorithm or amount of hidden neurons may lead to the construction of a good model.

The following remarks and considerations are focused on showing that the degeneration of activation functions as a result of considering their application on binary data alone is much stronger in the case of sigmoid and RBF than in the Tanimoto activation function. In particular, we will investigate the richness of the feature space measured as the size of the image of the $\mathbb{R}^d$ subset, which is truly filled with our data points. In other words, we will show, that the Tanimoto activation function better captures relations (similarities) between objects inside the space of possible chemical compound representation[4] $\{0,1\}^d$.

More importantly, this variety of values is practically realized. We recall from information theory that Shannon's entropy is the measure of the random variable's uncertainty. In our case, if we measure the entropy of achieving particular values of the activation function for the hidden neuron, we will obtain a comparable measure of how diverse the activations are. Alternatively, one can think about it as a minimal number of bits required to encode data produced with the given probability distribution. Thus, we may view our neurons as coding methods, and we are interested in how predictable their outputs are when they are fed with our data. Consequently, the greater the unpredictability is, the richer the output space.

Given a discrete random variable X with values $x_1, ..., x_k$,

[4]Using binary fingerprints.

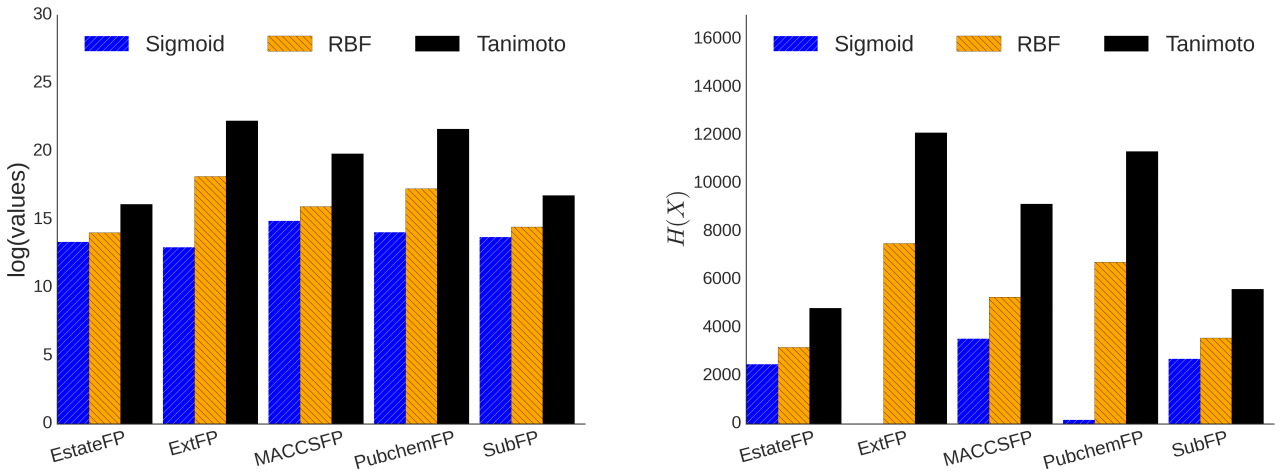


Fig. 5. Comparison of the number of achievable values (in log scale) of sigmoid, RBF and Tanimoto activation functions (on the left) and the entropy $H(\cdot)$ (on the right) using different fingerprints.

the Shannon's entropy of X is defined as

$$H(X) = -\sum_{i=1}^{k} \Pr(X = x_i) \log_2 \Pr(X = x_i).$$

One can easily observe in Fig. 5 that the number of achievable activation function values is orders of magnitude greater in the case of the Tanimoto function (notice that the y-axis is in the log scale). Although most of them are highly unlikely to appear, the entropy of these values is also significantly greater for a Tanimoto function (approximately 1.5 to 2 times larger than the RBF entropy), showing that Tanimoto truly captures more complex similarity patterns in binary, cheminformatics data.

This supports our claim that Tanimoto neurons are better suited for processing set-like data (sparse, binary vectors) than traditional neurons as they capture more complex features of such objects.

D. Projection speed

All Sigmoid, RBF and Tanimoto activation functions can be easily vectorized. As a result, they require simple operations:

- Sigmoid: dot product, division, addition and exponentiation;
- RBF: dot product, division, 2-norm, addition and exponentiation;
- Tanimoto: dot product, division, 1-norm and addition.

Although exponentiation is a highly optimized operation in modern hardware, it is still expensive compared to basic arithmetics. As a result, projection through Tanimoto neurons is approximately 1.5 times faster than through RBF and sigmoid (see Fig. 6) using a modern computer and writing code with NUMPY. There is no significant difference between the other two activation functions, although the sigmoid function requires slightly less computation.

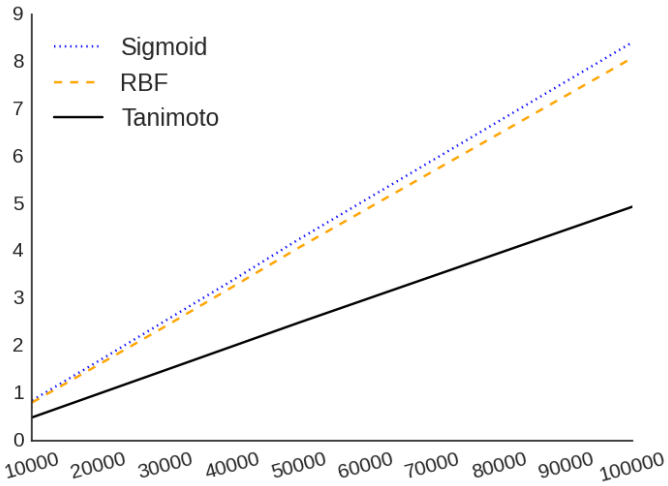


Fig. 6. Projection time for different activation functions. X axis denotes the number of samples in $\mathbb{R}^{100}$, projected onto 1000 hidden neurons. Y axis denotes time in seconds. The results are averaged across 100 random matrices.

In summary, we showed that the Tanimoto activation function significantly differs from the traditional scalar product and RBF-based functions. It projects binary data into much richer space as shown both theoretically and empirically. Furthermore, these values are truly achievable as shown using the entropy approach. Finally, we showed why such an approach is significantly faster from the computational point of view.

Let us now investigate the imbalanced datasets problem.

E. Class balancing

ELMs, in their most basic form, are not well-suited for imbalanced classification [32]. This is a direct result of the fact that their main optimization routine is based on the ordinary least squares (OLS) method. OLS assumes that each observation has the same variance; as a result, it is aimed at the maximization of the accuracy rather than any balanced metric. However, there are many approaches that can address

imbalanced datasets (for both binary classification [33] and multi-class classification [34]). In particular it is possible to use the weighted least squares method for ELMs, which has already been applied in the past [32]. In general, one constructs the diagonal weight matrix Q such that Q_{ii} equals the inverse of the ith sample importance. The training routine becomes

$$\beta = \left(\tfrac{I}{C} + \mathbf{H}^T Q \mathbf{H}\right)^{-1} \mathbf{H}^T Q \mathbf{T}. \tag{8}$$

It is easy to show that such an approach solves the L^2-regularized weighted least squares problem. The C parameter controls the regularization strength; higher values of C lead to non-regularized linear regression (possibly overfitting), while small values lead to the creation of the trivial model (possibly underfitting). A valid choice of this parameter is required for best generalization results, and it is often performed using some additional metaoptimization (such as a cross-validation-based grid search, as used in our evaluation).

A similar approach can be adapted for SVM, in which the C parameter defines the tradeoff between regularization and correct fitting to data. If we use distinct C_i for each sample, we can control per-sample weighting and thus analogous prior assumptions about parameter variance. During evaluation, we will use these techniques for both ELMs and SVMs to balance classes priors.

We could use (8) for Tanimoto ELMs; however, we propose a slightly different approach. Instead of diagonal matrix Q, we construct vector B, such that B_i is equal to the square root of the inverse of the ith sample importance. For simplicity, we will use notation $B \cdot A$ to denote element-wise multiplication of A by the weight vector B. It is easy to show that the proposed modification is mathematically equivalent, but (as shown in Section V) has important practical consequences.

We can now formulate the optimization problem being solved for training set $\mathbf{X}, \mathbf{T} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^N$, where $\mathbf{X} \in \mathbb{R}^{d \times N}$ and $\mathbf{T} \in \{-1, +1\}^N$ using the above observations (8) and (5):

Optimization problem: Weighted Tanimoto ELM

$$\begin{aligned} \underset{\beta}{\text{minimize}} \quad & \left\| \left(B \cdot \tfrac{\mathbf{X}^T\mathbf{W}}{\mathbb{1}\mathbf{X} + \mathbb{1}\mathbf{W} - \mathbf{X}^T\mathbf{W}} \right) \beta - B \cdot \mathbf{T} \right\|_2^2 + \tfrac{1}{C} \|\beta\|_2^2 \\ \text{where} \quad & \mathbf{W} \text{ is randomly chosen from } \mathbf{X} \\ & B_i = \sqrt{\text{MAX}(N_{-1}, N_{+1})/N_{\mathbf{t}_i}},\ i = 1, \ldots, N \\ & N_{\mathbf{t}} = |\{i : \mathbf{t}_i = \mathbf{t}\}|,\ \mathbf{t} = -1, +1. \end{aligned}$$

V. Efficient training

It is worth noting, that the slight difference in a weighting approach, in which instead of computing $\mathbf{X}^T Q \mathbf{X}$, we actually used $(B \cdot \mathbf{X})^T (B \cdot \mathbf{X})$, where $B_i = \sqrt{Q_{ii}}$, makes training twice as fast. This is caused by the fact that computation of the $A^T A$ for matrix $A \in \mathbb{R}^{N \times h}$ for $N > 2h$ can be easily sped up twice due to the resulting matrix symmetry and the fact that computation of this element is the efficiency bottleneck of the whole process. Such optimizations are available in current BLAS and LAPACK distributions. A simple change of order in the operations in any code using these basic libraries (such as python's NUMPY or SCIPY) considerably speeds up computation.

We are actually solving our optimization problem using standard L^2-regularized weighted least squares (compare with (8)) by putting

$$\beta = \left(\tfrac{I}{C} + \hat{\mathbf{H}}^T \hat{\mathbf{H}}\right)^{-1} \hat{\mathbf{H}}^T B \cdot \mathbf{T},$$

where

$$\hat{\mathbf{H}} = B \cdot \mathbf{H} = B \cdot \frac{\mathbf{X}^T\mathbf{W}}{\mathbb{1}\mathbf{X} + \mathbb{1}\mathbf{W} - \mathbf{X}^T\mathbf{W}}.$$

As both $\mathbf{X}$ and $\mathbf{W}$ are sparse binary matrices, computation of all four scalar products required for $\mathbf{H}$ is very fast. Assuming that $\mathbf{X}$ has average m nonzero elements in each column, we can compute $\mathbf{X}^T\mathbf{W}$ scalar product with a naive algorithm in just $\mathcal{O}(Nhm)$ operations and summation, divisions and scalar products with $\mathbb{1}$ take, i.e., $\mathcal{O}(Nh)$ operations. As a result, the entire $\mathbf{H}$ computation is only $\mathcal{O}(Nhm)$ basic operations long (only additions and multiplications, no exponentiation or logarithms).

The proposed method leads to extremely simple algorithms for both training and prediction (see Algorithm 1).

Algorithm 1 Weighted Tanimoto ELM.

TRAIN($\mathbf{X}, \mathbf{T}$)
  select randomly $W \subset \mathbf{X}$
  $N_{\mathbf{t}} \leftarrow |\{i : \mathbf{t}_i = \mathbf{t}\}|$ **for** $\mathbf{t} = -1, +1$
  $B_i \leftarrow \sqrt{\text{MAX}(N_{-1}, N_{+1})/N_{\mathbf{t}_i}}$ **for** $i = 1, ..., N$
  $\hat{\mathbf{H}} \leftarrow B \cdot (\mathbf{X}^T\mathbf{W})/(\mathbb{1}\mathbf{X} + \mathbb{1}\mathbf{W} - \mathbf{X}^T\mathbf{W})$
  $\beta \leftarrow \left(I/C + \hat{\mathbf{H}}^T \hat{\mathbf{H}}\right)^{-1} (\hat{\mathbf{H}}^T B \cdot \mathbf{T})$
  return $\mathbf{W}, \beta$

PREDICT($\mathbf{X}$)
  $\mathbf{H} \leftarrow (\mathbf{X}^T\mathbf{W})/(\mathbb{1}\mathbf{X} + \mathbb{1}\mathbf{W} - \mathbf{X}^T\mathbf{W})$
  return SIGN($\mathbf{H}\beta$)

VI. Evaluation

For evaluation purposes we compare seven ELM-based methods and six state-of-the-art approaches, namely the following: classical ELM with a sigmoid activation function (ELM [22]), Weighted ELM with sigmoid (WELM [32]), Weighted ELM with RBF (WELM_{RBF}), Weighted Tanimoto ELM (T-WELM), SVM with class balancing[5] and RBF (SVM_{RBF}), SVM with class balancing and a Tanimoto kernel (SVM_{TAN}), Random Forest (RF [36]), Weighted Random Forest (wRF[6]), AdaBoost [37] and Weighted AdaBoost (wAdaBoost). Additionally, each method except ELM and SVMs has an alternative version denoted by *, which means that in this case, hidden neurons are selected randomly from the training set instead of the continuous probability distribution.

All experiments were performed using code written in Python with use of SCIKIT-LEARN, NUMPY [38] and SCIPY [39] packages included in ANACONDA[7]. All ELM-based approaches were implemented in the uniform way,

[5]We use automatic class balancing from SCIKIT-LEARN library [35], where C parameter is changed into C_+ and C_- cost parameters, one for each class.

[6]We use analogous samples weighting scheme as in case of weighted ELMs and SVMs.

[7]https://store.continuum.io/cshop/anaconda/

ensuring comparable results in terms of computation time. For SVM we used the SCIKIT-LEARN SVC module, which implements a well-known LIBSVM [40] library, a low-level C implementation of the SVM. Unfortunately, using a full Tanimoto kernel with SVM failed to converge in a reasonable time in our experimentations. To overcome this issue and show comparable results, we instead used a random projection through the Tanimoto activation function (the same as in T-ELM*) followed by the highly efficient large scale linear SVM implementation (LIBLINEAR [41]), which is denoted as SVM_{TAN} in our evaluation. Both Random Forest and AdaBoost are implemented through the SCIKIT-LEARN ensemble module.

We used machines with Intel Xeon 2.8GHz processors and sufficient RAM to perform all needed computations in memory. We did not use any GPU-based acceleration to provide fair comparison with SVM implementation that was not accelerable, although it is very easy to run both an ELM [42] and a T-ELM on a graphic card.

We investigated eight different biological targets in our experiments, i.e., the following serotonin receptors: $5\text{HT}_{2\text{A}}$, $5\text{HT}_{2\text{C}}$, 5HT_6, 5HT_7, M_1, H_1, HIV INTEGRASE and HIV PROTEASE. Unfortunately, due to positive results bias (i.e., mainly positive results being published), there is a very small amount of actual, experimentally confirmed inactive compounds (true negative samples). As a result, using a dataset consisting of only experimentally confirmed samples has the great flaw of disrupting the samples class ratios (as in real life there are much more inactive compounds than active ones). To overcome this issue, for each protein, we consider a dataset consisting of ligands (compounds) of experimentally confirmed biological activity (positive samples) and negative samples generated using the DUD methodology [18] to obtain a more realistic positive-negative ratio. This phase of the experiment was carried out under supervision of experts in the field of computational chemistry from the Institute of Pharmacology of the Polish Academy of Sciences to ensure correctness from the perspective of the drug design process. The exact number of particular samples generated for each protein is summarized in Table I. For each such dataset we investigated 5 popular fingerprints (φ embeddings) generated using PaDEL descriptors software [17]: ESTATEFP, EXTFP, MACCSFP, PUBCHEMFP and SUBFP. Such a procedure results in 40 distinct datasets with binary labeling.

TABLE I
COMPARISON OF DATASETS USED DURING EVALUATION. n_k DENOTES SIZE OF THE k CLASS, d IS THE REPRESENTATION DIMENSION ($\varphi : \mathcal{C} \to \mathbb{R}^d$) AND $\bar{m} = \frac{1}{|\mathbf{C}|}\sum_{c\in\mathbf{C}} \|\varphi(c)\|_1$ IS MEAN SPARSITY OF PARTICULAR FINGERPRINT AMONG ALL USED COMPOUNDS $\mathbf{C}$.

protein	N_{+1}	N_{-1}
$5\text{HT}_{2\text{A}}$	1835	9019
$5\text{HT}_{2\text{C}}$	1210	97459
5HT_6	1490	135723
5HT_7	704	56653
M_1	759	27604
H_1	635	48608
HIV INTEGRASE	101	6395
HIV PROTEASE	3155	21163

fingerprint	d	$\bar{m}$
ESTATEFP	79	10
EXTFP	1024	282
MACCSFP	166	52
PUBCHEMFP	881	151
SUBFP	307	15

We use a GMEAN[8] (geometric mean of accuracy over positive and negative samples) as an evaluation metric. Due to its balanced nature and usage in previous works regarding Weighted ELM [32]. We also include results obtained with Balanced Accuracy[9] used in previous works regarding compounds activity prediction [43]. One could obviously analyze many other classification quality metrics; however, this would require a change in the weighting schemes in all the balanced methods (WELM, T-WELM, SVM with class weighting, wRF, and wAdaBoost), which is beyond the scope of this work.

Experiments were performed in the repeated 10-fold cross-validation manner with fixed maximum evaluation time at 10 minutes per training. Each model was tuned in terms of hyperparameters; thus, for ELMs, we tuned the number of hidden neurons ($h = 250, 500, 1000, 1500, 2000$) and regularization parameter $C = 10^{-1}, ..., 10^{9}$; for SVM with the RBF kernel, we tuned $\gamma = 10^{-10}, ..., 10^{0}$ and the regularization parameter $C = 10^{-1}, ..., 10^{9}$; and for Tanimoto SVM we tuned kernel sampling size ($h = 250, 500, 1000, 1500, 2000$) and regularization parameter $C = 10^{-4}, ..., 10^{3}$. Biases and weights of the hidden ELM neurons (in both the sigmoid and RBF cases) are drawn from the uniform distribution on $[0, 1]$. Random Forest and AdaBoost use decision trees as base estimators (with tuned number of trees/estimators $t = 10, 50, 100, 200, 500$), and RF used the Gini index for node splitting.

Table II summarizes the results of all considered models for GMEAN, BAC scores and time used for training. First, it is quite natural that the imbalanced ELM completely failed for data with highly skewed positive/negative distributions. It is quite interesting that the ELM using sigmoid activation functions also achieved very weak generalization scores in the experiments. Only the use of SubFP or EstateFP resulted in a GMEAN above 90%.

TABLE II
SUMMARY OF THE EXPERIMENTS. EACH VALUE REPRESENTS MEAN OVER ALL CONDUCTED EXPERIMENTS.

model	mean BAC	mean GMEAN	mean time
ELM	55.13%	54.92%	20.3s
WELM	69.38%	69.16%	20.4s
WELM*	73.98%	73.76%	17.6s
WELM_{RBF}	95.65%	95.43%	21.6s
$\text{WELM}^*_{\text{RBF}}$	94.18%	93.95%	20.7s
T-WELM	97.91%	97.69%	18.1s
T-WELM*	**98.03%**	**97.80%**	**14.6s**
SVM_{RBF}	97.06%	97.32%	198.7s
SVM_{TAN}	97.66%	97.44%	54.2s
RF	92.70%	92.47%	69.8s
wRF	93.38%	93.14%	64.5s
AdaBoost	84.91%	84.68%	127.0s
wAdaBoost	95.65%	95.42%	129.7s

Weak results for the ELM were probably caused by the high dimension and sparsity of the remaining fingerprints, for which scalar product based-methods are not well suited. It is further supported by the fact that once the hidden neurons

[8] $\text{GMEAN(TP,FP,TN,FN)} = \sqrt{\frac{\text{TP}}{\text{TP+FN}} \cdot \frac{\text{TN}}{\text{TN+FP}}}$.
[9] $\text{BAC(TP,FP,TN,FN)} = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP+FN}} + \frac{\text{TN}}{\text{TN+FP}}\right)$.

are selected from the training set instead of being completely random, scores increase significantly (although they remain much worse than those for other methods). The RBF-based ELM behaved very well, with surprisingly bad results using the ExtFP representation; this was possibly also due to the high dimension of such data, which would require a more careful choice of the distributions from which biases are drawn (e.g., methods based on Random Projections [44]). There are some existing heuristics for the choice of such values; in the case of SVM, it is common practice to choose a γ parameter such as $1/d$ (the heuristic used in LIBSVM) or to use some statistics of the data distribution [45]. Possibly, something similar could be used for seeding the distribution for RBF-based ELMs.

The proposed method achieved best results in most of the datasets. The results are comparable to those obtained by SVM with an RBF kernel (however, T-WELM is stil slightly better), while they are significantly better than all of the other ELM-based models as well as RF and AdaBoost. In the case of our method, selecting $\mathbf{W}$ from the training set led to a slight increase in overall scores, but more importantly, such an approach has the important advantage of removing the possibly tunable parameter of the distribution from which we draw parameters, which simplifies working with a model. The results are also more stable in the sense of the standard deviation of the GMEAN (and BAC) between folds than the results of the SVM models, which makes it a more reliable classification tool. Based on the scores, it is quite clear that the core element here is the connection between Tanimoto similarity and the ELM, as both the ELM without Tanimoto and SVM with Tanimoto behave significantly worse than T-WELM.

Now we proceed to the analysis of the training time of the evaluated models. We measure empirical time from the moment of feeding the model with data until it is fully trained, i.e., it comprises both the data projection (if needed) and actual training algorithm time. All weighted ELMs use the proposed efficient training scheme, to compare the speed differences in the activation function and model. When we used the standard weighting technique [32] the training times were approximately 40% worse than the training time reported in this paper. Table II summarizes the results for each protein-fingerprint pair. One can notice that training times of traditional ELMs (ELM, WELM, WELM_{RBF}) are quite similar, as expected, with a slightly slower projection through the RBF function. The times of the ELMs with hidden units drawn from the training set were slightly smaller, as they exploit the high data sparseness of both the $\mathbf{X}$ and $\mathbf{W}$ matrices (instead of just the $\mathbf{X}$ sparseness).

The T-WELM (with $\mathbf{W}$ selected from the training set) was the fastest machine evaluated. In only one of 40 cases, when dealing with a small set of HIV protease ligands represented by MACCSFP, the WELM_{RBF} was 10% faster. In all other experiments, it is very clear that the combination of highly efficient ELM training with extremely fast Tanimoto projection leads to a speedup of 50% compared to other ELM models. The training speed is an order of magnitude faster than that of RBF SVM (which, we emphasize, was the only model with very similar GMEAN results). Even the SVM with low-dimensional sampling of the Tanimoto kernel followed by a highly efficient linear solver was at least a few times slower than the proposed method.

These differences are even larger once we take into consideration the whole training process. Apart from projecting the data and training a classifier, we need to tune hyperparameters to achieve good classification results. Here, the T-WELM also achieves the best results. To use SVM with an RBF kernel, we have to tune two real-valued parameters of arbitrary scale (γ and C), and with the 10×10 grid search (which is not very precise), we need approximately 100 times more time to train a particular classifier. At the same time, ELM-based techniques require tuning of a number of hidden units and the C parameter. The number of hidden neurons is a natural number, which can be very approximately estimated, as it has been shown [22] to behave very stably across large intervals of values. It is also very important that the ELM training time is fully predictable (it can be nearly perfectly predicted from the number of data points and hidden units). In practice, we check just 5 values of hidden units. The regularization parameter also has very little impact on ELM results. In fact, in our experiments, the only differences were obtained for $C = 1, 10, 100$, while bigger values yielded exactly the same GMEAN as $C = 1000$ (for nearly all cases). As a result, the tuning of C in ELMs is also a very easy task and could be easily reduced to checking just a few values after which the models are no longer changing. Using such heuristics, we had to check merely $5 \cdot 3 = 15$ hyperparameter pairs for the ELM models. This gives differences that are approximately 10 times (one order of magnitude) greater than those of the SVM models reported in the table. One additional element distinguishing the T-WELM from other ELMs is the fact that it uses the unbiased activation function, which removes the need for a search for a good distribution to draw biases from. Although it might not look like a problem, the experiments suggest that the selection of biases from a valid distribution might significantly impact classification accuracy. As a result, although this distribution is not considered a hyperparameter in the literature, we claim that in practical considerations it matters, so development of the model that behaves well without the need of such tuning is important.

VII. CONCLUSION

In this paper, we have introduced the Weighted Tanimoto Extreme Learning Machine, a simple classification method that is well suited for the drug discovery problem, with which one addresses highly imbalanced datasets consisting of sparse, binary vectors. We showed some theoretical aspects of the proposed approach and performed careful evaluation on large amounts of data. The results confirm that the T-WELM is a valuable tool, with important properties:

- high classification quality (better than ELMs, SVMs, SVMs with a Tanimoto kernel, Random Forest and AdaBoost);
- rapid projection to the feature space (approximately 1.5 times faster than the sigmoid and RBF methods);

- fast training (traditional weighting results in a slowdown of approximately 40% compared to the proposed weighting);
- small number of hyperparameters (in fact only number of hidden neurons);
- theoretical justification of most of the observed aspects.

However, there are many aspects that require future work. In particular, we noted an open problem with the ability of a discrete ELM (an ELM with sparse, binary weights) to learn an arbitrary set of points in $\{0,1\}^d$ space. It would also be valuable to further investigate the aspect of the richness of the projected space. We showed some basic properties of sigmoid, RBF and Tanimoto activation functions in this paper, but the problem is much wider. It is also worth noting that despite this work being devoted to the drug design classification problem, the proposed method can have much wider applicability. In particular, problems in natural language processing share some similarities (large datasets of imbalanced datasets of binary vectors), and it would be valuable to further investigate the T-WELM in terms of such data.

ACKNOWLEDGMENT

This work was partially funded by National Science Centere Poland grant no. 2014/13/B/ST6/01792.

First, we would like to thank Jacek Tabor, Sabina Smusz, and Daniel Wilczak for their valuable help. We would also like to thank the editors and anonymous reviewers for their valuable insights and criticism of this work.

REFERENCES

[1] S. Smusz, R. Kurczab, and A. J. Bojarski, "The influence of the inactives subset generation on the performance of machine learning methods," *Journal of Cheminformatics*, vol. 5, no. 17, p. 8, 2013.

[2] L. Xue and J. Bajorath, "Molecular descriptors in chemoinformatics, computational combinatorial chemistry, and virtual screening," *Combinatorial Chemistry & High Throughput Screening*, vol. 3, no. 5, pp. 363–372, 2000.

[3] R. Kurczab, S. Smusz, and A. J. Bojarski, "The influence of training actives/inactives ratio on machine learning performance," *Journal of Cheminformatics*, vol. 5, Supplement 1, p. 1, 2013.

[4] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proc. of 2004 IEEE International Joint Conference on Neural Networks*, vol. 2, 2004, pp. 985–990.

[5] S. Decherchi, P. Gastaldo, A. Leoncini, and R. Zunino, "Efficient digital implementation of extreme learning machines for classification," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 8, pp. 496–500, 2012.

[6] H. Li, C. Y. Ung, C. W. Yap, Y. Xue, Z. R. Li, and Y. Z. Chen, "Prediction of estrogen receptor agonists and characterization of associated molecular descriptors by statistical learning methods," *Journal of Molecular Graphics & Modelling*, vol. 25, no. 3, pp. 313–323, Nov. 2006.

[7] M. Wang, X.-G. Yang, and Y. Xue, "Identifying hERG potassium channel inhibitors by machine learning methods," *QSAR & Combinatorial Science*, vol. 27, no. 8, pp. 1028–1035, Aug. 2008.

[8] F. Hammann, H. Gutmann, U. Baumann, C. Helma, and J. Drewe, "Classification of cytochrome P_{450} activities using machine learning methods," *Molecular Pharmaceutics*, vol. 33, no. 1, pp. 796–801, 2009.

[9] D. C. Weis, D. P. Visco, and J.-L. Faulon, "Data mining PubChem using a support vector machine with the signature molecular descriptor: classification of factor XIa inhibitors," *Journal of Molecular Graphics & Modelling*, vol. 27, no. 4, pp. 466–475, Nov. 2008.

[10] X. Liu, H. Song, J. Zhang, B. Han, X. Wei, X. Ma, W. Cui, and Y. Chen, "Identifying novel type ZBGs and nonhydroxamate HDAC inhibitors through a SVM based virtual screening approach," *Molecular Informatics*, vol. 29, no. 5, pp. 407–420, May 2010.

[11] L. Wang, Z. Wang, A. Yan, and Q. Yuan, "Classification of Aurora-A kinase inhibitors using self-organizing map (SOM) and support vector machine (SVM)," *Molecular Informatics*, vol. 30, no. 1, pp. 35–44, Jan. 2011.

[12] L. Ralaivola, S. J. Swamidass, H. Saigo, and P. Baldi, "Graph kernels for chemical informatics," *Neural Networks*, vol. 18, no. 8, pp. 1093–1110, 2005.

[13] A. Smalter, J. Huan, and G. Lushington, "Graph wavelet alignment kernels for drug virtual screening," *Journal of Bioinformatics and computational biology*, vol. 7, no. 03, pp. 473–497, 2009.

[14] L. H. Hall and L. B. Kier, "Electrotopological state indices for atom types: A novel combination of electronic, topological, and valence state information," *Journal of Chemical Information and Modeling*, vol. 35, no. 6, pp. 1039–1045, 1995.

[15] C. Steinbeck, Y. Han, S. Kuhn, O. Horlacher, E. Luttmann, and E. Willighagen, "The chemistry development kit (CDK): an open-source Java library for chemo- and bioinformatics." *Journal of Chemical Information and Computer Sciences*, vol. 43, no. 2, pp. 493–500, 2003.

[16] J. Klekota and F. P. Roth, "Chemical substructures that enrich for biological activity." *Bioinformatics*, vol. 24, no. 21, pp. 2518–2525, Nov. 2008.

[17] C. W. Yap, "Padel-descriptor: An open source software to calculate molecular descriptors and fingerprints," *Journal of Computational Chemistry*, vol. 32, no. 7, pp. 1466–1474, 2011.

[18] N. Huang, B. K. Shoichet, and J. J. Irwin, "Benchmarking sets for molecular docking," *Journal of Medicinal Chemistry*, vol. 49, no. 23, pp. 6789–6801, 2006.

[19] M. von Korff, J. Freyss, and T. Sander, "Comparison of ligand-and structure-based virtual screening on the dud data set," *Journal of Chemical Information and Modeling*, vol. 49, no. 2, pp. 209–231, 2009.

[20] S. G. Rohrer and K. Baumann, "Maximum unbiased validation (MUV) data sets for virtual screening based on PubChem bioactivity data," *Journal of Chemical Information and Modeling*, vol. 49, no. 2, pp. 169–184, 2009.

[21] T. Scior, A. Bender, G. Tresadern, J. L. Medina-Franco, K. Martínez-Mayorga, T. Langer, K. Cuanalo-Contreras, and D. K. Agrafiotis, "Recognizing pitfalls in virtual screening: A critical review," *Journal of Chemical Information and Modeling*, vol. 52, no. 4, pp. 867–881, 2012.

[22] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.

[23] W. Zong and G.-B. Huang, "Face recognition based on extreme learning machine," *Neurocomputing*, vol. 74, no. 16, pp. 2541–2551, 2011.

[24] N.-Y. Liang, P. Saratchandran, G.-B. Huang, and N. Sundararajan, "Classification of mental tasks from EEG signals using extreme learning machine," *International Journal of Neural Systems*, vol. 16, no. 1, pp. 29–38, 2006.

[25] J. Kim, H. S. Shin, K. Shin, and M. Lee, "Robust algorithm for arrhythmia classification in ECG using extreme learning machine," *Biomed Eng Online*, vol. 8, p. 31, 2009.

[26] B.-S. Oh, J. Jehyoung, K.-A. Toh, A. Beng Jin Teoh, and K. Jaihie, "A system for signature verification based on horizontal and vertical components in hand gestures," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 52–55, 2013.

[27] H. Yu, Y. Chen, J. Liu, and G.-B. Huang, "An adaptive and iterative online sequential ELM-based multi-degree-of-freedom gesture recognition system," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 55–59, 2013.

[28] L. L. C. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong, "Representational learning with ELMs for big data," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 31–34, 2013.

[29] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.

[30] I. J. Schoenberg, "Metric spaces and completely monotone functions," *Annals of Mathematics*, pp. 811–841, 1938.

[31] T. Poggio and F. Girosi, "A theory of networks for approximation and learning," DTIC Document, Tech. Rep., 1989.

[32] W. Zong, G.-B. Huang, and Y. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229–242, 2013.

[33] J. A. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle, "Weighted least squares support vector machines: Robustness and sparse approximation," *Neurocomputing*, vol. 48, no. 1, pp. 85–105, 2002.

[34] I. T. Podolak and A. Roman, "Theoretical foundations and experimental results for a hierarchical classifier with overlapping clusters," *Computational Intelligence*, vol. 29, no. 2, pp. 357–388, 2013.
[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[36] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
[37] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of online learning and an application to boosting," in *Computational Learning Theory*. Springer, 1995, pp. 23–37.
[38] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: A structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011.
[39] E. Jones, T. Oliphant, and P. Peterson, "SciPy: Open source scientific tools for python," *http://www.scipy.org/*, 2001.
[40] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.
[41] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.
[42] M. van Heeswijk, Y. Miche, E. Oja, and A. Lendasse, "GPU-accelerated and parallelized ELM ensembles for large-scale regression," *Neurocomputing*, vol. 74, no. 16, pp. 2430–2437, 2011.
[43] S. Smusz, W. M. Czarnecki, D. Warszycki, and A. J. Bojarski, "Exploiting uncertainty measures in compounds activity prediction using support vector machines," *Bioorganic & Medicinal Chemistry Letters*, vol. 25, no. 1, pp. 100–105, 2015.
[44] P. Gastaldo, R. Zunino, E. Cambria, and S. Decherchi, "Combining ELM with random projections," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 18–20, 2013.
[45] G. Rubio, H. Pomares, I. Rojas, and L. J. Herrera, "A heuristic method for parameter selection in LS-SVM: Application to time series prediction," *International Journal of Forecasting*, vol. 27, no. 3, pp. 725–739, 2011.