Current Pharmaceutical Design, 2018, *24*, 3347-3358

REVIEW ARTICLE

Machine Learning-based Virtual Screening and Its Applications to Alzheimer's Drug Discovery: A Review

Kristy A. Carpenter and Xudong Huang*

Neurochemistry Laboratory, Department of Psychiatry, Massachusetts General Hospital and Harvard Medical School, Charlestown, MA 02129, USA

ARTICLE HISTORY

Received: April 24, 2018
Accepted: June 1, 2018

DOI:
10.2174/1381612824666180607124038

Abstract: ***Background*:** Virtual Screening (VS) has emerged as an important tool in the drug development process, as it conducts efficient *in silico* searches over millions of compounds, ultimately increasing yields of potential drug leads. As a subset of Artificial Intelligence (AI), Machine Learning (ML) is a powerful way of conducting VS for drug leads. ML for VS generally involves assembling a filtered training set of compounds, comprised of known actives and inactives. After training the model, it is validated and, if sufficiently accurate, used on previously unseen databases to screen for novel compounds with desired drug target binding activity.

***Objective*:** The study aims to review ML-based methods used for VS and applications to Alzheimer's Disease (AD) drug discovery.

***Methods*:** To update the current knowledge on ML for VS, we review thorough backgrounds, explanations, and VS applications of the following ML techniques: Naïve Bayes (NB), k-Nearest Neighbors (kNN), Support Vector Machines (SVM), Random Forests (RF), and Artificial Neural Networks (ANN).

***Results*:** All techniques have found success in VS, but the future of VS is likely to lean more largely toward the use of neural networks – and more specifically, Convolutional Neural Networks (CNN), which are a subset of ANN that utilize convolution. We additionally conceptualize a work flow for conducting ML-based VS for potential therapeutics for AD, a complex neurodegenerative disease with no known cure and prevention. This both serves as an example of how to apply the concepts introduced earlier in the review and as a potential workflow for future implementation.

***Conclusion*:** Different ML techniques are powerful tools for VS, and they have advantages and disadvantages albeit. ML-based VS can be applied to AD drug development.

Keywords: Artificial intelligence, machine learning, drug discovery, virtual screening, k-nearest neighbors, support vector machines, naïve bayes, random forests, artificial neural networks, convolutional neural networks, Alzheimer's disease.

1. INTRODUCTION

Today, drug development is a very time-consuming and capital-intensive process. A 2014 study found that the cost of developing a prescription drug is on average $2.87 billion [1], and has likely since increased. Getting a potential drug through development stages and clinical trial phases takes years to complete, and often compounds fail before ever reaching the market. One of the more problematic processes in the drug development pipeline is the early stage of identifying potential drug leads among thousands to millions of candidate compounds. High-throughput Screening (HTS) of compounds is very time- and resource-heavy, especially when considering the low number of hits it produces. In an attempt to remedy this, many researchers choose to supplement the *in vitro* HTS with Virtual Screening (VS). This *in silico* process is a faster and cheaper way of searching for potential leads and can be utilized to reduce the number of compounds put through HTS, thereby greatly increasing HTS's yield. Like all computational biology processes, it is important to note that VS is not a replacement for HTS, because any sort of simulation or computer approximation is never guaranteed to be accurate – rather, it is a tool to aid and be used in conjunction with experiments.

1.1. Virtual Screening

The first step in VS is the assembly of a compound database on which to conduct the screening. This may begin with pulling mass quantities from publicly available chemogenomics libraries such as ChEMBL [2], PubChem [3], or ZINC [4], each include tens of millions of compounds annotated with information about their structure, known targets, and in the case of ZINC, purchasability. It is also common for pharmaceutical companies to use their own, in-house compound databases – which may have come from drugs that did not pass all clinical trials – to conduct VS. Whether the initial dataset is queried from the internet or not, it must go through further filtering in order to discard infeasible compounds and lower the number of false positives. It is common for researchers to exclude compounds that are much larger than the binding site of their target, or ones that are not available for purchase within the desired timeframe. Most datasets also get filtered according to Lipinski's Rule of Five [5] or standard metrics for lead-likeness [6], which remove compounds that are unlikely to be good drugs or leads, respectively. This is a particularly important step because VS is not done in isolation, but rather for the purpose of developing a drug for production. Finding a compound that can bind to the target but cannot be properly physiologically absorbed (or in the case of a lead, modified to be properly physiologically absorbed) does not accomplish this goal. Likewise, it is important to take precautionary measures to reduce the number of false positives, which take up time and resources in the hit-to-lead development and clinical trial phases only to ultimately fail. This can be done by removing compounds deemed to be pan-assay interference compounds (PAINS) [7].

*Address correspondence to this author at the Neurochemistry Laboratory, Department of Psychiatry, Massachusetts General Hospital and Harvard Medical School, Charlestown, MA, USA; Tel: 001-617-724-9778; Fax: 001-617-726-4078; E-mail: Huang.Xudong@mgh.harvard.edu

18'3-4286/18 $58.00+.00

Once the dataset has been assembled, the next step is to perform the actual screening. This can be done in a structure-based or ligand-based manner, or with a combination of the two. Structure-Based Virtual Screening (SBVS) involves examination of the structures of the ligand and target binding site and evaluation of the likelihood that the ligand will bind. This is most often done with docking, which involves "placing" compounds in the binding site of the target and scoring how likely they are to bind given a predetermined metric [8]. This method relies upon knowledge of each compound's structure, as well as the structure of the target. Ligand-Based Virtual Screening (LBVS) does not require structure information, but rather the molecular and chemical properties of known actives and the tested compounds [9]. The idea behind LBVS is that undiscovered actives will share some chemical features with the known ones. While it seems counterintuitive to use a drug discovery method that requires already knowing viable compounds, it is possible that the preexisting compounds cause undesirable side effects, do not treat all stages of the disease, or target something that has developed resistance to them.

There are advantages and disadvantages to both screening techniques. SBVS has the potential to discover actives with novel scaffolds, while LBVS is restricted to finding actives that share a limited number of predetermined chemical descriptors with known ligands. Additionally, if a target has no known actives, it is only possible to conduct SBVS. However, SBVS varies widely in accuracy due to approximations in physics, thermodynamics, and molecular positioning, and is dependent upon the use of a very accurate scoring function [10]. In this way, LBVS is more dependable. It is therefore up to the researchers to decide which screen is more appropriate to their experiment – or how to combine the hits produced by using both screens on the same data.

After obtaining hits from the VS, it is imperative to validate the results *in vitro*. Once one or more compounds have been experimentally verified as being able to bind to the desired target, they can undergo hit-to-lead development and clinical trials in order to hopefully be made into viable drugs.

But how does one actually perform VS? It is not an option to simply perform an *in silico* HTS simulation with molecular dynamics, as this would be extremely computationally intensive and likely take an incredible amount of time to run. Instead, computational chemists are turning to ML in order to efficiently conduct VS.

1.2. Machine Learning

Machine learning (ML) is a subfield of Artificial Intelligence (AI), and the two are the biggest buzzwords in many technological fields today. It has led to incredible breakthroughs in image processing [11] and Natural Language Processing (NLP) [12], and is being utilized in a number of other fields including sentiment analysis [13] and autonomous vehicles [14]. This versatility comes from the fact that ML constitutes generalizable methods of learning that only require large training datasets in order to perform well. The upsurge in chemical data availability makes ML viable for VS.

Before learning about the applications of ML in VS, it is important to understand its general principles. While most computer programs require an input and some functions to produce an output, ML uses training inputs and outputs to generate a function, which it can then use on test inputs to produce corresponding outputs. A good ML implementation must follow the *Structural Risk Minimization* (SRM) principle: it strikes an ideal balance between being both generalizable to unseen testing data and not overfitting the training data. This is done by minimizing the confidence interval (which corresponds to overfitting) and minimizing the empirical risk (which is the average error for the training data) [15]. ML can be *supervised* or *unsupervised*. The former involves giving inputs already labelled with classifications and asking the computer to determine the classification pattern; the latter uses unlabeled inputs and requires the computer to cluster similar data points in order to generate logical classes. Because the purpose of VS is to determine the activity of tested compounds, only supervised learning algorithms are used.

Two important processes that are common to all forms of ML – and particularly to ML in VS – are dataset preparation and model validation. A good ML model learns from a thoughtfully curated training dataset and is applied to a distinct testing set. When used for VS, both sets must consist of compounds with labelled binding activity – the training set requires this in order to establish patterns that the model can learn, and the testing set requires this for evaluating the model's accuracy. Active compounds must be taken from experimental results. Inactives may also be selected this way, but it is not always the case that a chosen chemogenomics library will contain enough nonbinding compounds that have been tested against the target for which the VS is being conducted. For this reason, many researchers opt to use decoy compounds, which are structurally similar to actives but have very different chemical features. The rationale behind this is that it is important to provide inactives that physically resemble actives to prevent the VS model from erroneously equating common structural features with activity, and that the decoys' chemical differences are sufficient to assume a high unlikeliness to bind. The most common method of obtaining decoys is through the use of directories such as DUDE [16]. Dissimilarity between known actives and assumed inactives can be further enforced by also calculating the Tanimoto coefficient [17] and excluding presumed inactives that are too similar to actives.

The Tanimoto coefficient serves another key purpose; it can provide a measure of the dataset's diversity. Diversity is critical for the creation of good training and testing sets in order to make the resulting ML model as general as possible. For this reason, it is typical to calculate the average Tanimoto coefficient between all compounds in the dataset to ensure that it is sufficiently diverse.

Once the overall dataset has been assembled, it is very likely that there will be an imbalance between the number of active and inactive compounds. This can be problematic for some ML methods [18]. Potential ways to resolve this problem are negative-undersampling of inactives and/or positive-oversampling of actives [19].

After all this preparation, the labelled dataset can be split into training and testing data. Most often, about 70% of the data goes to the training set and the remainder to testing [20-27], although an 80/20 split can also be used [28-30]. These splits are usually done randomly – however, it has been shown that a temporal-based split generally increases classifier accuracy [31]. An alternative to splitting is *k*-fold cross-validation, in which the dataset is randomly split into *k* partitions of equal size. *k-1* partitions are used as training data, and the final partition is the testing data. This process is repeated a total of *k* times, with each partition serving as the testing set exactly once. The final model is chosen based on the split that produced the lowest error. The value of *k* is usually chosen to be 5 or 10. The special case when *k* is equal to the number of samples is called leave-one-out cross-validation.

Regardless of how the testing set is separated from the training set, it is used in a process called *internal validation* in order to judge an ML model. This is often done by first calculating the confusion matrix of the model, which consists of the intersections between predicted actives/inactives and actual actives/inactives. The confusion matrix yields values that can help quantify the performance of any ML model: sensitivity (Eq. **1**) [32], specificity (Eq. **2**) [32], accuracy (Eq. **3**) [32], and the Matthews' Correlation Coefficient (MCC) (Eq. **4**) [33]. In each of these equations, TP, FP, TN, and FN represent the number of true positives, false positives, true negatives, and false negatives, respectively.

$$SE = \frac{TP}{TP + FN} \#(1)$$

$$SP = \frac{TN}{TN + FP} \#(2)$$

$$Q = \frac{TP + TN}{TP + TN + FP + FN} \#(3)$$

$$MCC = \frac{TP \times TN - FN \times FP}{\sqrt{(TP + FN)(TP + FP)(TN + FN)(TN + FP)}} \#(4)$$

The MCC is usually used to compare the performance of different models, with a perfect model having a score of 1. It is also very common to measure accuracy by way of the area under the receiver operating characteristic curve (AUC), which plots SP against 1 - SE. Again, the closer this value is to 1, the better. An AUC of 0.5 indicates performance equivalent to random classification. ML for VS also often uses the Boltzmann-enhanced discrimination of receiver operating characteristic (BEDROC) [34], an accuracy metric specifically designed to compare VS ranking methods.

Now that we have reviewed the general workflows of both VS and ML, we can dive into specific ML techniques. There are many types of ML, and we will describe them below in turn, along with how to specifically use them for VS and an overview of how recent VS research has achieved so far.

2. MACHINE LEARNING METHODS FOR VIRTUAL SCREENING

2.1. Naïve Bayes

One of the simpler ML techniques is the Naïve Bayesian (NB) classifier. This classifier is based on Bayes' Theorem [35], which relates conditional probabilities (Eq. **5**).

$$\Pr[A|B] = \frac{\Pr[B|A] * \Pr[A]}{\Pr[B]} \#(5)$$

In general, A is some class, and B is a feature of the data. The equation above shows that it is possible to find the probability that an input with some feature belongs to a certain class given three quantities that can be obtained from the training set: the probability that a member of the class has a certain attribute, the probability that an arbitrary data point belongs to that class (called the "prior probability"), and the probability that an arbitrary data point has that feature (the "marginal probability") [30].

NB classifiers are best applied for LBVS. The classes must be active and inactive compounds. The attributes can be molecular descriptors, which are properties of the molecule such as molecular weight or AlogP that are usually calculated with software such as MOE [36], PaDel [37], or Discovery Studio [38]. Molecular fingerprints [39] also can serve as attributes. These are binary strings that represent structural information and can be used to measure molecular similarity in non-ML contexts.

The direct application of Bayes' Theorem may give probabilities that are erroneously large – or zero – if some attribute is undersampled in the training set and becomes incorrectly associated with exclusively one class. In order to combat this, multiple studies [9, 30] utilized a Laplace estimator (Eq. **6**).

$$\Pr_{[lap]}[A|B] = \frac{(\Pr[B|A] * \Pr[A]) + 1}{\Pr[B] + (\Pr[A])^{-1}} \#(6)$$

While some studies have found that NB performs poorly in comparison to other ML methods [20, 26], it does have the advantage of not being susceptible to the "curse of dimensionality" [40]. This phenomenon causes ML methods based on clustering (such as k-Nearest Neighbors and Support Vector Machines, both described below) to lose accuracy when the number of samples does not increase exponentially with the number of dimensions. Because effective LBVS will occur in a high-dimensional space, non-clustering methods like NB become more manageable due to their reasonably-sized datasets. NB is also able to extract important attributes from the data, instead of acting like a black box. Knowing which specific features highly correlate to membership of either the active or inactive class allows for discovery of "privileged fragments" or "unprivileged fragments," respectively [9, 21, 23, 26, 41]. These fragments can greatly aid in scaffold design for compounds that are likely to bind to the desired target.

Several VS experiments have found success with NB classifiers. Yu *et al.* used one in conjunction with 3D Quantitative Structure-Activity Relationship (QSAR) pharmacophore hypothesis modeling to find prospective inhibitors of PI3Kα, a key target protein for many cancers; *in vitro* assays confirmed the discovery of some novel inhibitors [22]. Wang *et al.* chose an NB classifier from over 800 ML models that utilized four distinct techniques because it performed the best on external testing and gave favorable fragments. When using this classifier on a novel compound database, 56 hits were found after filtering, and *in vitro* assays deemed 12 of them as significantly active [42]. Jang *et al* used an NB classifier as a crucial step in their drug discovery workflow, which produced new and structurally diverse hits for mGlu1 receptor inhibitors [21]. Lian *et al*. combined NB models together with Support Vector Machines (described below) to create an enhanced ensemble model that produced 9 potent Influenza A neuraminidase inhibitors [43].

2.2. k-Nearest Neighbors

The k-Nearest Neighbors (kNN) classification method [44] is another simple ML method. The idea behind kNN is even more intuitive than that of NB: when projecting data into a feature space, the class of a given point is most likely going to be the same as its nearest neighbors. In its most basic implementation, kNN performs classification by assigning a point to the class that is most prevalent out of the *k* points closest to it. The *k* parameter can either be predetermined or chosen from a given range (usually between 1 and 5, though can go as far as the number of compounds screened [45]) based on internal validation scores. Usually, an odd *k* is used in order to prevent ties. The most common way of measuring distance between points is Euclidean distance, though other metrics such as Manhattan distance can also be used. If some features have much greater ranges than others, often a normalization process will occur before measuring distance in order to avoid erroneously ignoring variability in small-scale features.

Since the feature vectors used in VS are often based on QSAR [46] information, there are too many descriptors for all to be projected into a feature space in a computationally efficient manner. In order to allow for high-volume data analysis, a subset of features must be chosen, but there is no way of initially knowing which are important for binding activity to the desired target. This can be discovered through variable selection [45], in which randomness and statistical mechanics help refine a QSAR topography that is suitable for use in the kNN model.

Variable selection optimizes *k*, *n* (the number of features chosen), and the features used in the topography. The workflow begins by choosing *k* and *n* values, each within a given range. Then *n* random features are chosen. The predictive power (q^2) of the current model is calculated through leave-one-out cross-validation over all the compounds in the dataset. In the q^2 equation (Eq. **7**), *m* represents the number of compounds, y_i represents activity of the i^{th} compound, $\hat{y}_i$ represents the predicted activity of the i^{th} compound, and $\bar{y}$ represents the average activity of all compounds.

$$q^2 = 1 - \frac{\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{m}(y_i - \bar{y})^2} \#(7)$$

Because trying all possible combinations of features would not be accomplishable in a reasonable amount of time, a process called *simulated annealing* [45, 47] – modeled after statistical mechanics – tweaks the model by random perturbation and adoption of a new set of features if the predictive power improves. After completing the variable selection process, a kNN-QSAR model using a subset of all introduced QSAR features is obtained that can be used for binding activity predictions.

Many recent VS studies further enhance kNN-QSAR models by also implementing Multi-Task Learning (MTL) [48, 49]. MTL involves modeling multiple related tasks in parallel, rather than trying to do each in isolation. Its driving idea is that it is easier to learn several related complex tasks together than it is to do so separately because of *inductive bias* – the preference given to hypotheses/methods that aid in accomplishing more than one task. For example, in VS it is helpful to model binding activity to many different targets concurrently because although no target has the same binding site, the laws of chemistry are uniform. If two targets are closely related, it is very likely that they will interact in similar manners with a given compound. Therefore, it is useful to consider interactions with both targets when constructing an ML model for their binding in general. In the kNN approach, this means parallelizing variable selection so that the optimal kNN-QSAR topographic model (and related parameters, k and n) is built by taking many targets into account. While there is a general consensus that MTL on related tasks outperforms Single-Task Learning (STL), it is important to note that some improvements are simply due to the fact that kNN-QSAR performance is closely related to the size of the dataset used [26, 50].

Luo *et al*. found that kNN-QSAR with variable selection outperformed LBVS approaches that do not use ML in searching for ligands of G-Protein Coupled Receptors [47]. However, when comparing kNN with other ML methods, its performance often falls somewhere in the middle [26, 41, 42]. Because of this, kNN is not used quite as often as the more popular ML methods described below. Nevertheless, kNN has still been successfully employed for VS: for example, in the screening for Estrogen Receptor-mediated endocrine disruptors [50].

2.3. Support Vector Machines

Support Vector Machines (SVMs) were first introduced by Vapnik *et al*. [15, 51]. They function by representing input data as feature vectors and plotting them in a space with the same dimensionality. The SVM will then construct an optimal hyperplane that divides the data points into two categories. Because SVM is most often used in supervised learning, these categories are usually predetermined. It has been shown that SVM *can* be used for unsupervised learning [52]; however, for the purposes of VS, supervised learning is more desirable because this guarantees that the classifier will assign a compound as either an active or an inactive.

An ideal dataset would be perfectly linearly separable – that is, it would be possible to draw a hyperplane in the feature space that has all the points of one class on one side, and all the points of another class on the other. In this case, there are many hyperplanes that will separate the classes of the training data with zero empirical risk. The optimal hyperplane is the one which minimizes the confidence interval by maximizing the margin of separation (the minimum distances between the hyperplane and the points closest to it) (Fig. **1**). This means that the selection of the optimal hyperplane is dependent upon the positions of the points closest to it. If and only if these points were to move, the optimal hyperplane would change. These important points are called *support vectors*. It can be shown that, regardless of the dimensionality of the hyperspace, if there are fewer support vectors, there will be a tighter bound on the expected error of the classifier (Equation **8**) [15].

$$Ex[\Pr[error]] \leq \frac{Ex[number\ of\ support\ vectors]}{(number\ of\ training\ vectors) - 1} \#(8)$$

The minimization of margin of separation can be calculated with Lagrange multipliers, and is described in detail by Vapnik [15].

Obviously not all datasets are ideal. If the points in a given dataset are not perfectly linearly separable in their initial hyperspace, one can use a kernel function to transform them into a hyperspace in which they are. Linear, polynomial, and radial basis function (RBF) kernels are often the kernels of choice. If there does not exist a kernel which can transform the datapoints into a hyperspace in which they are perfectly linearly separable, then a kernel and optimal hyperplane must be chosen that minimize the number of misclassified training points. In this case, the SVM is said to have a "soft margin" and requires a cost parameter (C) that dictates how much cushion there is for training set misclassification; a small C gives large empirical risk for the sake of generality, whereas a large C risks overfitting in order to have a "harder" boundary [51].

Most implementations of VS that use SVM do so with LIBSVM [53] and an RBF kernel [20, 21, 26, 28, 42, 43]. In order to set the metaparameters C and gamma (which is an input to the RBF), LIBSVM has a "grid" function which uses five-fold cross-validation to search for the optimal configuration.

SVMs are generally among the top performers in ML for VS comparison studies [26, 28, 41], and have been used successfully to

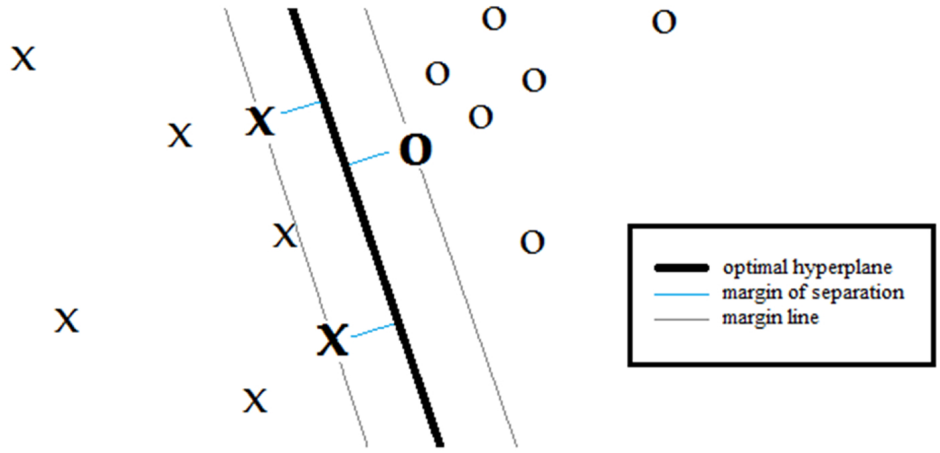

Fig. (1). An illustration of the optimal hyperplane that maximizes the margin between two classes, represented by x's and o's. The support vectors are bolded and lie on the margin lines.

identify novel drugs. Chandra *et al*. constructed several ML models to find PTP1B inhibitors in hopes of finding a potential treatment for Type-2 Diabetes; the best model used SVM and was run on an external database to choose five potential inhibitory compounds. *In vitro* experiments validated two of these as significantly active [41]. Similarly, Deshmukh *et al.* found that their SVM model could both identify nearly half of the known FEN1 inhibitors in a test set and discover previously unknown inhibitors from the Maybridge small molecule database, which were subsequently experimentally verified [28]. Baba *et al.* found that SVM models with regression generally outperformed Random Forests for predicting the ability of a given compound to permeate skin, which is important in the development of cosmetics and topical medicines [24].

2.4. Random Forests

Before one can properly understand Random Forests (RF) or Random Decision Forests, they must first understand their constituent element: the Decision Tree (DT). DTs are tree graphs which are used to partition data into different classes, first used prominently for ML by Quinlan in 1986 [54]. Every node in a DT can be thought of as a question about aspects of the data, and every outgoing edge from a particular node is an answer to its question. There are three types of "questions" that a node can employ (Fig. **2**). The first, and perhaps easiest to conceptualize, is the axis-parallel linear split. This evaluates a Boolean expression, often checking if some feature is above or below a given threshold. If the set of points is visualized in a feature space, the split created by this node can be represented by a line parallel to one of the axes. Alternately, an oblique linear split can be used – this is visualized as a hyperplane in the feature space, much like as if it were an SVM. The oblique linear split could truly act like an SVM by choosing the "optimal" hyperplane: first, it determines which two classes have means that are farthest apart; then, it reclassifies all other points according to which of the two means they are closer to; finally, it uses the methods described in the SVM section to find the hyperplane to produce the split. A second method of oblique linear splitting is through central axis projection. This type of splitting only considers the points belonging to the two farthest-apart classes, and picks the hyperplane that is perpendicular to the line connecting their means which minimizes misclassification between the two classes. The third type of split is the piecewise linear split, which is similar to kNN in that it classifies points according to which cluster they belong to in a Voronoi tessellation.

Regardless of splitting mechanism, each successive node will partition the data until each leaf contains only a single class. Doing this effectively requires each partition to become purer than its ancestors. The separation question at each node must minimize the weighted average of its children's impurities (Eq. **9**). Here, k represents the number of children, $|E_j|$ is the size of the j^{th} child, $|E|$ is the size of the node, and $I(E_j)$ is the impurity of the j^{th} child.

$$\sum_{j=1}^{k} \frac{|E_j|}{|E|} * I(E_j) \#(9)$$

Partition impurity can be calculated with entropy or the Gini index. Quinlan describes how to use entropy to find the split that gives the highest information gain (i.e. lowest impurity) [54]. Generally, the entropy of a partition is given by Eq. **10**, where m is the number of classes and p_i is the fraction of the i^{th} class in the partition). The higher the entropy, the higher the impurity.

$$-\sum_{i=1}^{m} p_i \log p_i \ \#(10)$$

The Gini index was first described in 1912 [55] and is often used to calculate wealth distribution (Eq. **11**). Models based on Breiman's CART algorithm [56] generally prefer the Gini index as the splitting criterion.

$$1 - \sum_{i=1}^{m} {p_i}^2 \#(11)$$

A major drawback of using DTs is their tendency to overfit to the training data. "Early stopping" can help prevent this – if the information gain (*i.e*. the difference between the entropy of the parent's data and the weighted average of the entropies of the children's data) does not exceed a predetermined threshold, splitting will be terminated. However, this can lead to a loss of accuracy on the training data, which also does not bode well for future classifications to be accurate.

The proposed solution to this is to use an ensemble of DTs, which is called a Random Forest [57]. An RF is made up of a variety of slightly different DTs, and suggests a classification for an input based on the most common output among all its constituent DTs. There are multiple ways to build an RF, but all involve growing each random tree using all the training data and some random vector. An example of this is the random subspace method [58, 59]. This method is based upon the idea of creating DTs using a subset of the full feature space. Each feature is randomly chosen to be included or excluded, and then a DT is built using one of the splitting techniques described above. Because the entire training set is used, there is no empirical risk, but each tree will generalize differently based on which features are excluded.

Another RF construction method is bagging [60]. The term "bagging" is a portmanteau of "bootstrap" and "aggregating," and the process it refers to involves growing an ensemble of DTs from bootstrap samples of the training set. Bootstrapping improves accuracy because, unlike methods such as kNN, DTs are unstable – meaning that small changes to the training set can result in large changes in their structures.

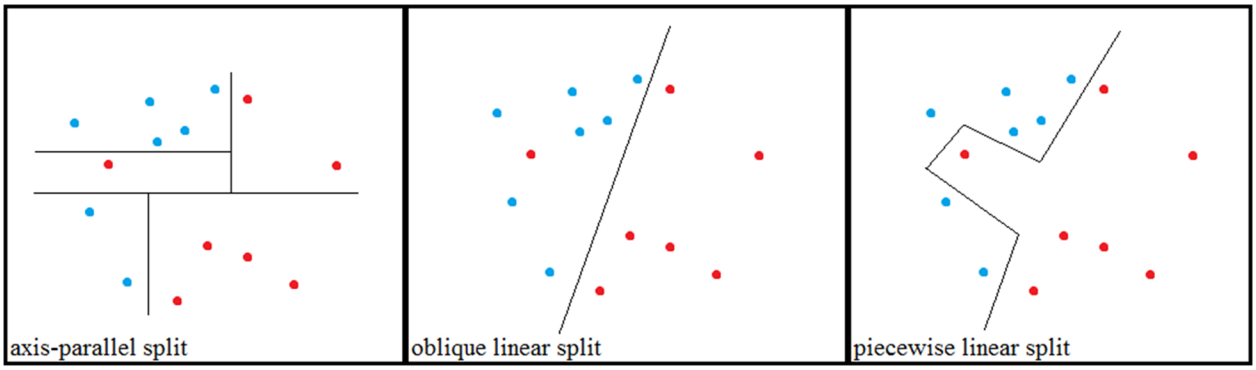

Fig. (2). The three types of node splits for a Decision Tree.

Thirdly, RFs can grow with boosting [61]. Boosting's underlying principle is that the repeated use of weak decision rules can eventually lead to an aggregated strong decision rule that will accurately classify the data. New decision rules are created by weighting previously misclassified training data more heavily. Perhaps the best boosting algorithm is Freund and Schapire's AdaBoost [62], which becomes more accurate as any weak decision rule improves (rather than just the least accurate decision rule, as in other boosting algorithms), making its decrease in error exponential.

RFs classify an input by running it through each of its DTs and assigning it to the most commonly outputted class. As the number of trees in the RF increases, so does its classification accuracy [59]. And not only are RFs more accurate than single DTs, but they also do not overfit the training data. This can be proven using the Strong Law of Large Numbers [57].

RFs can be easily trained in parallel, with different DTs running on different GPUs. This leads to a shorter training time at the cost of requiring more GPUs and the ability to parallelize the code.

As in NB classifiers, it is possible to extract some information about influential features from an RF. This can be done by comparing misclassification rates when noising each feature (i.e. setting its value randomly) in turn, or when noising all features but one [57]. This tentatively shows which features are most important in determining classification. Quinlan's C4.5 program [63] provides another way of doing this via production rules, which give the user an intuitive interpretation of the splits critical to classification in the DT.

Unsurprisingly, all the above reasons make DT/RF a popular type of ML, and it has been used widely in computational biology. For example, RFs have been implemented in order to identify SSL mutations [64] and predict domain-based protein-protein interactions [65]. They also are used for VS.

When implementing DT/RF for VS, the input data is often expressed in terms of QSAR, and the screen conducted is therefore ligand-based. This means that each node queries different QSAR properties, which can be taken from molecular descriptors or fingerprints. Interestingly, some VS studies choose to use single DTs (which in this context are often called Recursive Partitioning classifiers, or RPs) rather than RFs, and those that use RFs often choose bagging over AdaBoost. It remains to be seen if VS will catch up to the most recent advancements in RF, or if RP is sufficient for its purposes.

While RF-QSAR setups vary, there are some commonalities among studies. Most use Discovery Studio 3.5 [38] for their implementations. The number of DTs in a forest can range from 100 to 1000, and their depths can range from 2 to 30 (or have no specified threshold). When RFs are used, they are often pruned to combat their tendency to overfit [9, 42].

Herrera-Acevedo *et al.* paired RF-QSAR with docking to search for alternative antichagasic drugs that work well in the chronic phase and do not have as many adverse side effects [29]. They used a CART-based RF model that was tweaked in order to have as low of a FP rate as possible. This approach proved to be successful, having been verified against known actives, and produced several candidate drugs for further analysis. Deshmukh *et al.* also found their RF model to have a low number of false positives, and for this reason paired it with the more-accurate SVM model in their final inhibitor search [28]. Lee *et al.* used RF-QSAR to study the polypharmacology of compounds, ultimately creating a target-fishing server which can be used to discover potential targets for a given compound [19]. Their approach used bagged RFs and obtained an overall AUC score of 0.97, in addition to outperforming NB-based methods on external testing.

2.5. Artificial Neural Networks

The Artificial Neural Network (ANN) is one of the original ML algorithms, with its original rough implementation being Rosenblatt's Perceptron in 1958 [66]. ANNs loosely imitate how learning occurs in human brains, which have real networks of neurons, though as their usage in ML has progressed they have become more distant from attempted accurate neurological models.

While SVMs follow the SRM principle by fixing empirical risk and minimizing the confidence interval, ANNs do so by fixing the confidence interval and minimizing empirical risk [15]. This is accomplished by their architecture. ANNs are made up of sequential connected layers of neurons. The first layer takes in the input and is followed by some number of hidden layers, which process the data until it is eventually fed into an output layer that gives a classification (or a set of probabilities for different possible classifications). This data processing occurs in each individual neuron. A neuron in some layer receives multiple inputs (either from the input layer, or from neurons in the previous layer). It transforms these inputs with an activation function, and takes the sum of these plus a bias term in order to produce an output to the next layer of the network. Sigmoid functions are the standard activation function. These are characterized by being both differentiable and real-valued over the domain of all real numbers, having exclusively nonnegative derivatives, and being bounded (often between -1 and 1, or between 0 and 1). Commonly used sigmoid functions are the logistic function (Eq. **12**) and the hyperbolic tangent function (Eq. **13**). In recent years, Rectified Linear Units (ReLUs) (Eq. **14**) have emerged as another useful activation function [67]. Because ReLUs do not have an upper bound, they are called "non-saturating." It has been shown that ANNs can be trained faster with ReLUs than with sigmoidals [11], and that ReLUs on average lead to more accurate models [68].

$$f(x) = \frac{L}{1+e^{-k(x-x_0)}} \#(12)$$

$$f(x) = \tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}} \#(13)$$

$$f(x) = \max(0, x) \#(14)$$

The hidden layers of an ANN get their name from the fact that they are not directly indicated by the training data's output; it is up to the network to "decide" how to best utilize them in order to minimize empirical risk. This minimization occurs by a process called back-propagation [69], which involves proceeding backward through the ANN (*i.e*. from the output layer back to the input layer) and for some training input/output, calculating the objective function. The objective function measures the difference between the predicted output and the real output – in other words, the empirical risk for some datapoints. The next step is to determine how to change the weight of each neuron in order to decrease the objective function the most. This process is called gradient descent.

Back-propagation must occur repeatedly in order to properly refine the weights of hidden neurons. However, undergoing back-propagation for every datapoint in the training set is unwieldy and time-consuming. For this reason, it usually happens over small sets of examples, which are called mini-batches. Training an ANN typically involves conducting back-propagation over the same mini-batches multiple times – each iteration over a specific set of data is called a training epoch. The fact that the training process does not equally involve all training data introduces some amount of noise into the setting of weights, and this is noted by specifying that the ANN has undergone *stochastic* gradient descent.

While the general idea of ANNs has been around for a long time, they have only recently had a resurgence in the ML commu-

nity. This is because the initial ANN implementation described above is very susceptible to over-fitting and often did not perform as well as other types of ML. Recent developments in ANN construction have changed this, leading to a much more widespread use of the structure. In 2006, Hinton and Osindero introduced the concept of unsupervised pre-training, which occurs before stochastic gradient descent and initializes neuron weights so as to be more generalizable and closer to the objective function minimum [70]. Four years later, Martens showed that ANNs with 2nd order Hessian-free optimization outperform those with pre-training [71]. This was bested by Sutskever *et al.*'s implementation with momentum in 2013 [72]. Yet another important advancement in ANNs came a year later with the concept of dropout [73], which introduces a parameter that gives every neuron some probability of having no output. This essentially "drops" it from the network for some number of runs. Dropout leads to an increase in accuracy and combats overfitting because it penalizes heavily weighting individual neurons (*i.e*. discourages the fate of classification unduly resting on a single neuron). It also simulates the averaging of many similar ANNs which are each made up of strict subsets of the neurons contained in the actual ANN, which leads to more generalizability.

In addition to repeated breakthroughs in software architecture, advances in hardware have also bolstered the usability of ANNs. GPUs can process computations much faster than their predecessors, especially when combined with parallelization. This enables training of very wide and deep ANNs to occur over the course of several hours, rather than days or weeks.

Of course, all the new nuances in ANN construction come at a cost: an increase in the number of meta-parameters. Employing a highly accurate ANN with several wide layers that uses the techniques described above requires many decisions about construction: number of layers, number of neurons in each layer, type of activation function, dropout probability, mini-batch size, number of epochs, momentum strength, etc. The sheer number of possible settings can be daunting, especially when knowing that some parameter choices will lead to much more accurate results than others. There are a few ways to handle this. Many researchers opt to train several ANNs with different parameter settings and using internal validation to determine which is optimal for testing. The settings may be chosen based on studies comparing the accuracy of models with small changes in parameters for the purpose of suggesting optimal combinations, as done by Ma *et al.* [68]. An alternative is to eliminate the need to hand-pick parameters by using Bayesian optimization to automate the selection process [25].

There are two general classifications of ANNs: feed-forward and recurrent. Recurrent ANNs (RNNs) include feedback connections between the outputs of some neuron and itself or neurons in previous layers. These are well-suited for sequential input and are often used for tasks like sentence generation or translation. While they have been employed for *de novo* drug discovery through a study attempting to iteratively generate SMILES strings of compounds that would bind to a particular target [74], RNNs are not the right choice for VS. For this reason, the remainder of this section will exclusively discuss feed-forward ANNs, which are characterized by their lack of feedback connections.

A concept that arises often when discussing feed-forward ANNs is that of "deepness." The term Deep Neural Network (DNN) is nearly as much of a buzzword as ML or AI. However, there is some disagreement on what exactly a DNN entails. Some sources say that all neural networks are deep, since they are based on the concept of Deep Learning (DL) – the idea that ML best occurs when expressing a complex input in terms of many simpler representations, like a composition of many mathematical functions [75]. However, others claim that an ANN is only "deep" when it has many hidden layers. Today, any number of layers greater than two is likely to be considered "deep," though this threshold may increase in the future. Since most ANNs applied to problems such as VS use multiple hidden layers in order to break a very complex input down into multiple simple representations, we believe it is accurate to call them DNNs.

An ANN can also be called a *Convolutional Neural Network* (CNN or ConvNet) if at least one of its hidden layers utilizes convolution. Broadly speaking, convolution is a deep learning process that discovers clusters of related values located throughout a multidimensional input (Fig. **3**). This is accomplished with a combination of convolutional layers and pooling layers [76]. Convolution uses filters that pass through input in order to detect similar features throughout a tensor. These filters can vary in size, and this size is usually an odd number to facilitate their application to the input (the middle weight in the filter is applied to each node, with the rest of the filter lining up with the adjacent nodes). Padding of extra zeroes can be used to maintain the size of the input tensor. Pooling layers reduce noise by only passing on the maximum value in a given stretch of nodes. For example, a max pooling layer of size 3 will output the highest value of every group of three adjacent input nodes. After one or more convolutional/pooling layers, CNNs usually contain at least one fully connected layer with a typical activation function such as ReLU.

Because CNNs are best suited for processing complex data that comes in multiple arrays and contains repeated features, they currently most widely used in image processing. The first breakthrough CNN for this purpose was introduced by Krizhevsky, Sutskever, and Hinton in 2012, who presented it as a solution to the ImageNet classification challenge [11]. The convolutional and pooling layers of their CNN was able to identify image features, such as edges and pixel motifs. Their CNN also utilized general ANN-enhancing techniques, such as the ReLU activation function, dropout, and momentum. This all led to unprecedented success: it won the ImageNet LSVRC-2012 competition with a top-5 test error rate of 15.3%, and classified images from the 2010 equivalent with a top-5 error rate of 17.0%. Since then, a multitude of other CNNs have emerged in the field of computer vision [77-79].

CNNs also pair very well with QSAR representations of compounds, meaning that they are applicable for VS. One of the first CNNs used for VS was AtomNet™, developed by the company Atomwise, Inc. [80]. Unlike the majority of ML-based VS, AtomNet™ runs an SBVS, which works well with convolution's ability of extracting local feature clusters from multidimensional input. This gives AtomNet™ the advantage of being able to make predictions for targets without requiring knowledge about any of their actives and without predetermining which molecular properties are possible to check. AtomNet™'s architecture consists of an input layer, 4 convolutional layers, 2 fully-connected (*i.e*. non-convolutional) layers, and a final logistic-cost layer that determines output probabilities. When examining the filters that were developed in the convolutional layers, it was found that they corresponded to chemical functions. For example, one filter in the first convolutional layer became specialized to detect sulfonyls/sulfonamides. This makes sense, since chemical features are roughly the molecular compound equivalent of the edges in an image. With this advancement, ANNs in VS are no longer black boxes. Rather, they gain the ability to identify features which aid binding – something which was previously unique to NB and RF classifiers. And in addition to feature identification, AtomNet™ is also incredibly accurate, consistently achieving AUC scores greater than 0.74 on a variety of compound benchmark datasets and outperforming many previous docking models.

In addition to convolution, multi-task learning is another useful tool for ANN-QSAR. As discussed in the kNN section, MTL has proved to be useful in VS due to its ability to extract general rules about chemistry and the interactions between functional groups that are common to many compounds. MTL is able to be implemented with ANNs by using a common hidden architecture to produce multiple outputs (each related to some task – e.g. a specific target in

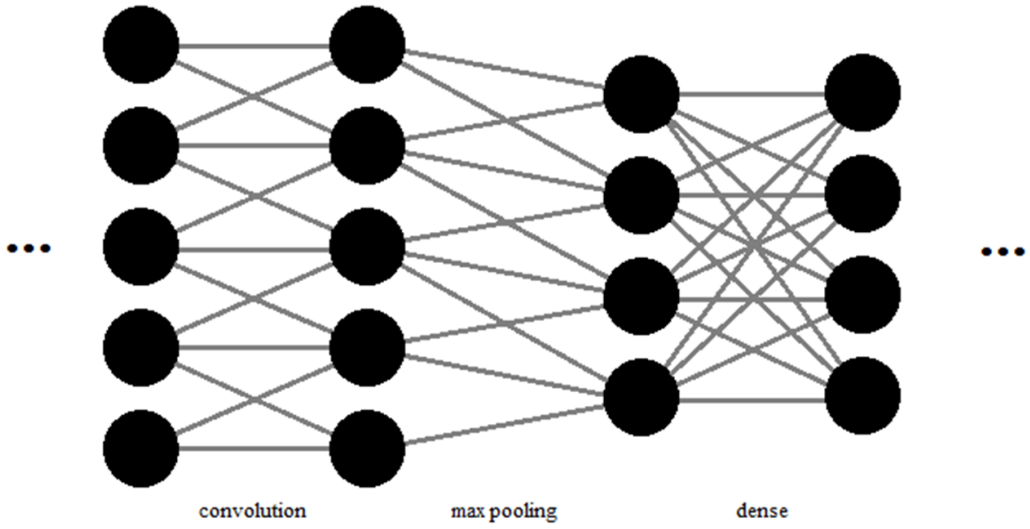

Fig. (3). Example section of a typical CNN architecture. Weighted connections between nodes are represented by gray lines. On the left is an input that gets convolved with a size 3 filter and has padding of 1, followed by max pooling with size 2. The output of this is fed into a fully-connected layer which will usually utilize a ReLU activation function. Ellipses indicate the presence of other layers before and after the depicted nodes.

the case of VS) for one input. Because the weights of the hidden neurons are determined by mini-batches from multiple QSAR targets, the network is not overly specialized toward one particular target. It also increases the likelihood of neurons encoding for general QSAR features which, as illustrated with AtomNet™, can be chemically interpreted. This was observed by Unterthiner *et al.*, who found that their multi-task DNN used for toxicology VS ended up with feature maps corresponding to chemical functional groups as well as toxicophore clusters [81].

It should be apparent that ANN has recently become one of the more dominant forms of ML in general, and the field of VS is certainly no exception to this. There are numerous examples of ANN models to conduct VS. Fjell *et al.* used QSAR and ANNs to screen for peptides that are likely to have antibiotic properties. They showed the applicability of their models by doing *in vitro* testing of hits, some of which showed significant antibiotic activity against a variety of drug-resistant bacterial strains [82]. Additionally, Durrant and McCammon developed NNScore, a program which uses a deep ANN with regression to predict the binding activity of compounds. NNScore has proven to be faster than existing docking programs that do not utilize ML, making it a better candidate for conducting SBVS [83].

Not only has there been an increase in VS studies using ANN, but comparison studies have also found that it consistently outperforms other types of ML. Lenselink *et al.* constructed a variety of ML models for testing the binding activity of different drugs in the ChEMBL library with the intent of directly comparing them against each other. Their DNNs had varying architectures – using one to three hidden layers, different levels of dropout, and some MTL – and generally had a statistically significant increase in accuracy (measured by BEDROC and MCC) over models based on RF, SVM, and NB [20]. Likewise, Dahl *et al.* compared multi-task and single-task ANNs with RFs and boosted RFs, and found that in the majority of assays, the best ANNs outperformed the best RFs. Additionally, the MTL models generally outperformed the STL ones [25]. In developing their DrugMiner web tool for finding viable drug targets, Jamali *et al.* compared the accuracies of a variety of common ML methods on their training and testing sets. They found that their ANN outperformed NB, kNN, RF, SVM, and DT in terms of classification accuracy, and subsequently chose that model for their launched software [84]. For these reasons, it is very likely that the future of VS will be dominated by the use of ANNs and CNNs.

2.6. Ensemble Methods

Instead of using only one type of classifiers, many studies that use ML opt to combine the results of an ensemble of models in hopes of increasing performance. This follows the same logic that underlies Random Forests – many similar but distinct models will make better predictions than one model. We have already mentioned some studies which utilized ensemble learning [28, 43]. In addition to these, MLViS is a tool that combines NB, NN, kNN, DT, SVM, and RF to produce predictions of whether given compounds will be druglike or not. MLViS is unusual in the sheer scope of models it used in training, as it was initially tested with 23 common classifiers and launched with 10 resulting algorithms for classification [85].

In a similar vein, some VS programs available online do not directly combine results to produce one classification, but rather present the user with several ML model options from which to pick. An example of this is MolClass, which is a toolkit that runs an uploaded dataset through RF, NB, SVM, and kNN models. The user can choose which models to pull their results from and view the overall activity profiles generated by MolClass [86].

3. APPLICATION: ALZHEIMER'S DISEASE

As described above, ML has proven to be incredibly useful for *in silico* drug screening. In this section, we conceptualize a workflow for applying ML-based VS to the search for potential therapeutic agents for Alzheimer's disease (AD).

AD is a neurodegenerative disease with no known cure and prevention. According to Alzheimer's Disease International, it afflicts approximately 44 million people worldwide as of 2016, and the number of AD patients is expected to only increase as time goes on. Unfortunately, AD and other neurological diseases are notoriously difficult to treat. Efforts to produce a drug capable of slowing neurodegeneration have been fruitless, and the most recent AD-related drug to pass clinical trials, Memantine, did so in 2003 [87]. The AD drugs currently available only alleviate symptoms, rather than reversing the course of the disease [88-90]. It is imperative to keep searching for a cure, and VS with ML is a promising method for AD drug discovery.

The first step in any VS is the identification of a target protein. As AD is a polygenic and multifactorial disease with complex ori-

gins, there is not an obvious target to choose. AD is characterized by aggregations of amyloid-beta (Aβ) plaques and neurofibrillary tangles (NFT) comprised of hyperphosphorylated tau protein [91, 92]. AD-affected brains also show a significantly reduced concentration of the neurotransmitter acetylcholine (ACh) [93, 94]. These two facts have sparked the main hypotheses around which AD treatments are based: the amyloid cascade hypothesis (the idea that the cognitive decline present in AD is caused by Aβ plaques) and the cholinergic hypothesis (the idea that it is caused by ACh loss). Early attempts to design an AD drug focused on the cholinergic hypothesis. Because of this, most existing AD treatments are cholinesterase inhibitors. However, since these drugs are all palliative and do not stop neurodegeneration, AD drug design going forward is paying more attention to the amyloid cascade hypothesis.

Many of the proposed Aβ-related targets are involved in the generation of Aβ. This process begins with beta-site amyloid precursor protein cleaving enzyme 1 (BACE1) cleaving the Amyloid Precursor Protein (APP), followed by γ-secretase making a second cut to produce Aβ [95]. A different enzyme, α-secretase, can cut APP in a different location, preventing creation of Aβ. With all this in mind, there is consensus that inhibitors of BACE1 or γ-secretase would make good AD drugs. Muscarinic ACh receptor (mAChR) agonists are also attractive due to the observation that mAChR stimulation increases the activity of α-secretase [96] – in addition to possibly degrading BACE1 [97]. Other potential targets are glycogen synthase kinase-3 beta (GSK-3β) and cyclin-dependent kinase 5 (CDK5), both of which are implicated in tau phosphorylation [98-100].

When choosing a target, it is imperative to consider potential side effects of its inhibition – especially when considering neurological processes. For example, numerous problematic phenotypes have arisen when breeding mice with the BACE1 gene knocked out [101-104]. While some of these results may not occur when inhibiting BACE1 late in life (rather than never having it from the beginning of development), these conclusions certainly raise a need for caution. Likewise, inhibition of γ-secretase may lead to complications in its associated Notch pathway and even a paradoxical decrease in cognition [105, 106].

Another possible cause of negative side effects is due to the polypharmacology of compounds. Often, a drug designed to inhibit one target will also be able to inhibit other proteins, leading to unforeseen consequences. However, polypharmacology can also have positive effects, particularly when attempting to treat a polygenic and multifactorial disease like AD, because it entails that one drug could have increased effectiveness by inhibiting multiple targets. This train of thought has led to an increase in screening for Multi-Target Directed Ligands (MTDLs) [107]. In fact, most recent VS studies for AD drug screen for MTDLs that inhibit some combination of the aforementioned targets. Xie *et al.* performed sequential dockings to screen for compounds that could inhibit both GSK-3β and CDK5 [108], and Kumar *et al.* did an LBVS for MTDLs that inhibit BACE1 and GSK-3β [109]. One combined LBVS/SBVS did not choose particular targets, but rather started with the scaffold of a preexisting AD drug to attempt to find similar MTDLs [110].

None of these VSs used ML, but an ML-based VS for AD is not completely unheard of. Fang *et al.* used NB and RP classifiers to conduct an LBVS for MTDLs [111]. They had a total of 25 targets in the screen, including BACE1, the M1 subtype of mAChR, APP, CDK5, and GSK-3β, and searched for compounds that bound to as many targets as possible. Current AD-related drugs were used to validate the model, which produced predicted MTDLs for further development.

It is interesting to note the absence of ANN-based VSs for AD; the general high performance of ANNs should make them an attractive method for this purpose. The ease of which MTL can be implemented with ANNs and the clear connections between MTL and screens for MTDLs further rationalize the approach. Furthermore, the emergence of CNNs and the applications of convolution in QSAR modeling make CNNs such as AtomNet™ promising. We propose using a multi-task, deep CNN for a VS of potential MTDLs that inhibit a combination of AD-related targets. These targets should include the enzymes and receptors mentioned above, though it is crucial to perform additional target-fishing to ensure that any predicted MTDLs do not interfere with critical neurological functions. Training and testing data should be pulled from some chemogenomics libraries, with include many examples of known inhibitors of each proposed target, and the dataset should be preprocessed as described in the background sections before using it to train and internally validate the CNN. At the moment, we do not have recommendations for a particular CNN architecture.

If a model constructed in this manner is sufficiently accurate, it should be used on a large dataset comprised of compounds distinct from the initial training and testing sets. The top hits from this screen should be purchased and put through *in vitro* assays in order to ascertain their effectiveness. Ideally, at least one compound would have the desired activity and could be further developed from a lead into a drug suitable for clinical trials.

CONCLUSION

At a time when drug development is steadily getting slower and costlier, it is vital to turn to cutting-edge technologies for aid. ML-based VS enables medicinal chemists to efficiently find potential lead molecules among millions of compounds in chemogenomics libraries, greatly increasing the yield of HTS and speeding up the initial stages of drug development. This review described the workings of Naïve Bayesian classifiers, k-Nearest Neighbors, Support Vector Machines, Random Forests, and Artificial Neural Networks, all of which are viable implementations of ML for the purposes of VS. While comparison studies have generally pointed to SVMs and ANNs as the most accurate VS models, it is important to note that each technique has its own advantages and disadvantages, which should be considered when designing a VS. For instance, NB can identify favorable scaffold fragments and is not susceptible to the Curse of Dimensionality, RFs are easily parallelizable and can be enhanced with boosting or bagging, and kNN is simple to implement and can utilize MTL. Of course, it is possible and perhaps even preferable to employ an ensemble of ML models, as this generally increases performance.

As illustrated in the Application section, it is very feasible to design an ML-based VS workflow to search for disease-specific drugs. While complex diseases like AD do not have absolutely definitive targets on which to conduct the VS, thinking about their underlying mechanisms and examining prior research may provide starting points. Coupling MTL approaches through ANN or kNN with the idea of screening for MTDLs has the promise of creating high-performing classifiers that output compounds that could potentially bind to multiple targets involved in a disease phenotype.

The multidisciplinary nature of ML-based VS has led to partnerships between powerhouses in the artificial intelligence and pharmaceutical industries. Such collaborations benefit from the combination of state-of-the-art hardware and ML technologies combined with vast in-house chemogenomics libraries. Already, IBM has partnered with several pharmaceutical companies such as Teva Pharma, Sage Bionetworks, and Pfizer [112]; the aforementioned creator of AtomNet™- Atomwise, Inc., is collaborating with Merck; and GNS healthcare has paired with Genentech for oncology drug discovery [113] – these are only a few examples, and it is likely that other partnerships will continue to form.

We expect the use of ML in VS for drug discovery to only grow as the scientific world realizes the power that it brings to the field. The joint efforts of computer science and medicinal chemistry are sure to make drug discovery process more efficient and less costly.

LIST OF ABBREVIATIONS

Aβ	=	Amyloid-beta
ACh	=	Acetylcholine
AD	=	Alzheimer's Disease
ANN	=	Artificial Neural Network
APP	=	Amyloid Precursor Protein
AI	=	Artificial Intelligence
AUC	=	Area Under Receiving Operator Characteristic Curve
BACE1	=	Beta-Site Amyloid Precursor Protein Cleaving Enzyme 1
BEDROC	=	Boltzmann-Enhanced Discrimination of Receiving Operator Characteristic
CDK5	=	Cyclin-dependent Kinase 5
DL	=	Deep Learning
CNN	=	Convolutional Neural Network
DNN	=	Deep Neural Network
DT	=	Decision Tree
FN	=	False Negative
FP	=	False Positive
GSK-3β	=	Glycogen Synthase Kinase-3 Beta
HTS	=	High-throughput Screening
kNN	=	k-nearest Neighbors
LBVS	=	Ligand-based Virtual Screening
mAChR	=	Muscarinic Acetylcholine Receptor
MCC	=	Matthews' Correlation Coefficient
ML	=	Machine Learning
MTDL	=	Multi-target Directed Ligand
MTL	=	Multi-task Learning
NB	=	Naïve Bayesian
NFT	=	Neurofibrillary Tangle
PAINS	=	Pan-assay Interference Compounds
Q	=	Accuracy
QSAR	=	Quantitative Structure-activity Relationship
RBF	=	Radial Basis Function
ReLU	=	Rectified Linear Unit
RF	=	Random Forest
RNN	=	Recurrent Neural Network
RP	=	Recursive Partitioning
SBVS	=	Structure-based Virtual Screening
SE	=	Selectivity
SP	=	Specificity
SRM	=	Structural Risk Minimization
STL	=	Single-task Learning
SVM	=	Support Vector Machine
TN	=	True Negative
TP	=	True Positive
VS	=	Virtual Screening

CONSENT FOR PUBLICATION

Not applicable.

CONFLICT OF INTEREST

The authors declare no conflict of interest, financial or otherwise.

ACKNOWLEDGEMENTS

This work was partially funded by a NIH grant R01AG056614 (to XH). The authors would like to thank the MIT externship program for allowing KAC to be a student intern at MGH.

REFERENCES

[1] DiMasi JA, Grabowski HG, Hansen RW. Innovation in the pharmaceutical industry: New estimates of R&D costs. J Health Econ 2016; 47: 20-33.

[2] Bento AP, Gaulton A, Hersey A, *et al*. The ChEMBL bioactivity database: an update. Nucleic Acids Res 2014; 42(Database issue): D1083-90.

[3] Kim S, Thiessen PA, Bolton EE, *et al*. PubChem substance and compound databases. Nucleic Acids Res 2016; 44(D1): D1202-13.

[4] Sterling T, Irwin JJ. ZINC 15--ligand discovery for everyone. J Chem Inf Model 2015; 55(11): 2324-37.

[5] Lipinski CA, Lombardo F, Dominy BW, Feeney PJ. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. Adv Drug Deliv Rev 2001; 46(1-3): 3-26.

[6] Verheij HJ. Leadlikeness and structural diversity of synthetic screening libraries. Mol Divers 2006; 10(3): 377-88.

[7] Baell JB, Holloway GA. New substructure filters for removal of pan assay interference compounds (PAINS) from screening libraries and for their exclusion in bioassays. J Med Chem 2010; 53: 22.

[8] Pitt WR, Calmiano MD, Kroeplien B, Taylor RD, Turner JP, King MA. Structure-based virtual screening for novel ligands. In: Williams MA, Daviter T, eds. Protein-Ligand Interactions: Methods and Applications. Totowa, NJ: Humana Press; 2013. p. 501-19.

[9] Xia J, Jin H, Liu Z, Zhang L, Wang XS. An unbiased method to build benchmarking sets for ligand-based virtual screening and its application to GPCRs. J Chem Inf Model 2014; 54(5): 1433-50.

[10] Hawkins PCD, Skillman AG, Nicholls A. Comparison of shape-matching and docking as virtual screening tools. J Med Chem 2007; 50: 9.

[11] Krizhevsky A, Sutskever I, Hinton GE, editors. Imagenet classification with deep convolutional neural networks. 26th Annual Conference on Neural Information Processing Systems; 2012; Lake Tahoe, Nevada, USA: Neural Information Processing Systems.

[12] Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P. Natural language processing (Almost) from scratch. JMLR 2011; 12: 45.

[13] Kolog EA, Montero CS, Toivonen T, Eds. Using machine learning for sentiment and social influence analysis in text2018; Cham: Springer International Publishing.

[14] Prado Álvaro J, Michałek Maciej M, Cheein Fernando A. Machine-learning based approaches for self-tuning trajectory tracking controllers under terrain changes in repetitive tasks. Eng Appl Artif Intell 2018; 67: 63-80.

[15] Vapnik VN. The nature of statistical learning theory. New York: Springer-Verlag; 1995.

[16] Mysinger MM, Carchia M, Irwin JJ, Shoichet BK. Directory of useful decoys, enhanced (DUD-E): better ligands and decoys for better benchmarking. J Med Chem 2012; 55(14): 6582-94.

[17] Tanimoto TT. An elementary mathematical theory of classification and prediction: International business machines corporation; 1958.

[18] Weiss GM, Provost F. The effect of class distribution on classifier learning: An empirical study. Technical Report ML-TR-43, Dept of Computer Science, Rutgers Univ. 2001.

[19] Lee K, Lee M, Kim D. Utilizing random Forest QSAR models with optimized parameters for target identification and its application to target-fishing server. BMC Bioinformatics 2017; 18(Suppl 16): 567.

[20] Lenselink EB, Ten Dijke N, Bongers B, *et al*. Beyond the hype: deep neural networks outperform established methods using a ChEMBL bioactivity benchmark set. J Cheminform 2017; 9(1): 45.

[21] Jang JW, Cho NC, Min SJ, *et al*. Novel scaffold identification of mglu1 receptor negative allosteric modulators using a hierarchical virtual screening approach. Chem Biol Drug Des 2016; 87(2): 239-56.

[22] Yu M, Gu Q, Xu J. Discovering new PI3Kalpha inhibitors with a strategy of combining ligand-based and structure-based virtual screening. J Comput Aided Mol Des 2018; 32(2): 347-61.

[23] Cai J, Li C, Liu Z, *et al*. Predicting DPP-IV inhibitors with machine learning approaches. J Comput Aided Mol Des 2017; 31(4): 393-402.
[24] Baba H, Takahara J-i, Mamitsuka H. *In silico* predictions of human skin permeability using nonlinear quantitative structure–property relationship models. Pharm Res 2015; 32(7): 2360-71.
[25] Multi-task Neural Networks for QSAR Predictions [Internet]. 2014.
[26] Li Y, Wang L, Liu Z, *et al*. Predicting selective liver X receptor beta agonists using multiple machine learning methods. Mol Biosyst 2015; 11(5): 1241-50.
[27] Wang L, Chen L, Liu Z, Zheng M, Gu Q, Xu J. Predicting mTOR inhibitors with a classifier using recursive partitioning and Naive Bayesian approaches. PLoS One 2014; 9(5): e95221.
[28] Deshmukh AL, Chandra S, Singh DK, Siddiqi MI, Banerjee D. Identification of human flap endonuclease 1 (FEN1) inhibitors using a machine learning based consensus virtual screening. Mol Biosyst 2017; 13(8): 1630-9.
[29] Herrera-Acevedo C, Scotti L, Scotti MT. *In silico* studies designed to select sesquiterpene lactones with potential antichagasic activity from an in-house Asteraceae database. ChemMedChem 2018; 13(6): 634-45.
[30] Watson P. Naive bayes classification using 2D pharmacophore feature triplet vectors. J Chem Inf Model 2008; 48: 13.
[31] Sheridan RP. Time-split cross-validation as a method for estimating the goodness of prospective prediction. J Chem Inf Model 2013; 53(4): 783-90.
[32] Baldi P, Brunak S, Chauvin Y, Andersen CAF, Nielsen H. Assessing the accuracy of prediction algorithms for classification: an overview. Bioinformatics 2000; 16(5): 412-24.
[33] Matthews BW. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochim Biophys Acta 1975; 405(2): 442-51.
[34] Truchon JF, Bayly CI. Evaluating virtual screening methods: good and bad metrics for the "early recognition" problem. J Chem Inf Model 2007; 47(2): 488-508.
[35] Puga JL, Krzywinski M, Altman N. Points of significance: Bayes' theorem. Nat Methods 2015; 12: 277-8.
[36] Molecular Operating Environment (MOE). 1010 Sherbooke St. West, Suite #910, Montreal, QC, Canada, H3A 2R7: Chemical Computing Group ULC; 2018.
[37] Yap CW. PaDEL-descriptor: an open source software to calculate molecular descriptors and fingerprints. J Comput Chem 2011; 32(7): 1466-74.
[38] BIOVIA RDS. Discovery Studio Modeling Environment. San Diego: Dassault Systemes; 2016.
[39] Cereto-Massague A, Ojeda MJ, Valls C, Mulero M, Garcia-Vallve S, Pujadas G. Molecular fingerprint similarity search in virtual screening. Methods 2015; 71: 58-63.
[40] Bellman RE. Adaptive Control Processes: A Guided Tour: Princeton University Press; 2015.
[41] Chandra S, Pandey J, Tamrakar AK, Siddiqi MI. Multiple machine learning based descriptive and predictive workflow for the identification of potential PTP1B inhibitors. J Mol Graph Model 2017; 71: 15.
[42] Wang L, Le X, Li L, *et al*. Discovering new agents active against methicillin-resistant *Staphylococcus aureus* with ligand-based approaches. J Chem Inf Model 2014; 54(11): 3186-97.
[43] Lian W, Fang J, Li C, Pang X, Liu AL, Du GH. Discovery of Influenza A virus neuraminidase inhibitors using support vector machine and Naive Bayesian models. Mol Divers 2016; 20(2): 439-51.
[44] Cover TM, Hart PE. Nearest neighbor pattern classification. IEEE Transactions on Information Theory 1967; 13(1): 7.
[45] Zheng W, Tropsha A. Novel variable selection quantitative structure-property relationship approach based on the k-nearest-neighbor principle. J Chem Inf Comput Sci 2000; 40: 10.
[46] Tropsha A. Best practices for QSAR model development, validation, and exploitation. Mol Inform 2010; 29(6-7): 476-88.
[47] Luo M, Wang XS, Tropsha A. Comparative analysis of QSAR-based vs. chemical similarity based predictors of GPCRs binding affinity. Mol Inform 2016; 35(1): 36-41.
[48] Caruana RA, editor Multitask Learning: A Knowledge-Based Source of Inductive Bias. Tenth International Conference on Machine Learning; 1993: Morgan Kaufmann.
[49] Caruana RA. Multitask Learning. Machine Learning 1997; 28: 35.
[50] Zhang L, Sedykh A, Tripathi A, *et al*. Identification of putative estrogen receptor-mediated endocrine disrupting chemicals using QSAR- and structure-based virtual screening approaches. Toxicol Appl Pharmacol 2013; 272(1): 67-76.
[51] Cortes C, Vapnik V. Support-vector networks. Machine Learning 1995; 20(3): 273-97.
[52] Winters-Hilt S, Merat S. SVM clustering. BMC Bioinformatics 2007; 8 Suppl 7: S18.
[53] Chang C-C, Lin C-J. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2011; 2(3):1.
[54] Quinlan JR. Learning Efficient Classification Procedures and their Application to Chess End Games. In: Michalski RS, Carbonell JG, Mitchell TM, editors. Machine Learning: An Artificial Intelligence Approach: Springer-Verlag; 1986.
[55] Gini C. Variabilità e mutabilità: contributo allo studio delle distribuzioni e delle relazioni statistiche. [|.]: Tipogr. di P. Cuppini; 1912.
[56] Breiman L, Friedman J, Stone CJ, Olshen RA. Classification and Regression Trees: Taylor & Francis; 1984.
[57] Breiman L. Random Forests. Machine Learning 2001; 45: 28.
[58] Ho TK. The Random Subspace Method for Constructing Decision Forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 1998; 20(8): 13.
[59] Ho TK, editor Random Decision Forests. 3rd International Conference on Document Analysis and Recognition; 1995; Montral, Que., Canada: IEEE.
[60] Breiman L. Bagging Predictors. Machine Learning. 1996;24:18.
[61] Schapire RE. The Boosting Approach to Machine Learning: An Overview. In: Denison DD, Hansen MH, Holmes CC, Mallick B, Yu B, editors. Nonlinear Estimation and Classification Lecture Notes in Statistics. 171. New York, NY: Springer; 2003.
[62] Freund Y, Schapire RE. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. J Comput Sys Sci 1997; 55: 21.
[63] Quinlan JR. C4.5: Programs for Machine Learning: Elsevier Science; 2014.
[64] Wong SL, Zhang LV, Tong AHY, *et al*. Combining biological networks to predict genetic interactions. PNAS 2004; 101(44): 6.
[65] Chen XW, Liu M. Prediction of protein-protein interactions using random decision forest framework. Bioinformatics. 2005;21(24):4394-400.
[66] Rosenblatt F. The Perceptron: A probabilistic model for information storage and organization in the brain. Psychol Rev 1958; 65(6): 23.
[67] Nair V, Hinton GE, editors. Rectified Linear Units Improve Restricted Boltzmann Machines. 27th International Conference on Machine Learning; 2010; Haifa, Israel.
[68] Lombardo S, Maskos U. Role of the nicotinic acetylcholine receptor in Alzheimer's disease pathology and treatment. Neuropharmacology 2015; 96(Pt B): 255-62.
[69] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature 1986; 323(9): 4.
[70] Hinton GE, Osindero S, Teh Y-W. A fast learning algorithm for deep belief nets. Neural Comput 2006; 18(7): 8.
[71] Martens J, editor Deep learning *via* Hessian-free optimization. 27th International Conference on Machine Learning; 2010; Haifa, Israel.
[72] Sutskever I, Martens J, Dahl G, Hinton GE, editors. On the importance of initialization and momentum in deep learning. 30th International Conference on Machine Learning; 2013; Atlanta, Georgia, USA: JMLR; W&CP.
[73] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. J of Machine Learning Research. 2014;15:30.
[74] Gupta A, Muller AT, Huisman BJH, Fuchs JA, Schneider P, Schneider G. Generative Recurrent Networks for De Novo Drug Design. Mol Inform 2017.
[75] Goodfellow I, Bengio Y, Courville A. Deep Learning: MIT Press; 2016.
[76] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015; 521(7553): 436-44.
[77] Chen X, Xu Y, W. KWD, Y. WT, Liu J, editors. Glaucoma detection based on deep convolutional neural network. 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC); 2015; Milan.
[78] Hong S, You T, Kwak S, Han B. Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network. In: Francis B, David B, editors. Proceedings of the 32nd Interna-

tional Conference on Machine Learning; Proceedings of Machine Learning Research: PMLR; 2015. p. 597-606.
[79] Pinheiro P, Collobert R. Recurrent Convolutional Neural Networks for Scene Labeling. In: Eric PX, Tony J, editors. Proceedings of the 31st International Conference on Machine Learning; Proceedings of Machine Learning Research: PMLR; 2014. p. 82-90.
[80] Wallach I, Dzamba M, Heifets A. AtomNet: A Deep Convolutional Neural Network for Bioactivity Prediction in Structure-based Drug Discovery. CoRR 2015: 11.
[81] Toxicity Prediction using Deep Learning [Internet]. 2015.
[82] Fjell CD, Jenssen H, Hilpert K, *et al*. Identification of Novel Antibacterial Peptides by Chemoinformatics and Machine Learning. J Med Chem 2009; 52(7): 2006-15.
[83] Durrant JD, McCammon JA. NNScore 2.0: A Neural-Network Receptor–Ligand Scoring Function. J Chem Inf Model 2011; 51(11): 2897-903.
[84] Jamali AA, Ferdousi R, Razzaghi S, Li J, Safdari R, Ebrahimie E. DrugMiner: comparative analysis of machine learning algorithms for prediction of potential druggable proteins. Drug Discov Today 2016; 21(5): 718-24.
[85] Korkmaz S, Zararsiz G, Goksuluk D. MLViS: A Web Tool for Machine Learning-Based Virtual Screening in Early-Phase of Drug Discovery and Development. PLoS ONE 2015; 10(4): e0124600.
[86] Wildenhain J, FitzGerald N, Tyers M. MolClass: a web portal to interrogate diverse small molecule screen datasets with different computational models. Bioinformatics 2012; 28(16): 2200-1.
[87] Rogawski MA, Wenk GL. The Neuropharmacological Basis for the Use of Memantine in the Treatment of Alzheimer's Disease. CNS Drug Rev 2003; 9(3): 275-308.
[88] Richarz U, Gaudig M, Rettig K, Schauble B. Galantamine treatment in outpatients with mild Alzheimer's disease. Acta Neurol Scand 2014; 129(6): 382-92.
[89] Schneider LS, Dagerman KS, Higgins JT, McShane R. Lack of evidence for the efficacy of memantine in mild alzheimer disease. Arch Neurol 2011; 68(8): 991-8.
[90] Qaseem A, Snow V, Cross J, Jr, *et al*. Current pharmacologic treatment of dementia: A clinical practice guideline from the american college of physicians and the american academy of family physicians. Ann Intern Med 2008; 148(5): 370-8.
[91] Hardy J, Selkoe DJ. The amyloid hypothesis of Alzheimer's Disease: progress and problems on the road to therapeutics. Science 2002; 297(5580): 4.
[92] Rapoport M, Dawson HN, Binder LI, Vitek MP, Ferreira A. Tau is essential to β-amyloid-induced neurotoxicity. Proc Natl Acad Sci USA 2002; 99(9): 6364-9.
[93] Bartus RT, Dean RL, Beer B, Lippa AS. The cholinergic hypothesis of geriatric memory dysfunction. Science 1982; 217(4558): 408-14.
[94] Woolf NJ. The critical role of cholinergic basal forebrain neurons in morphological change and memory encoding: A hypothesis. Neurobiol Learn Mem 1996; 66(3): 258-66.
[95] Vassar R, Bennett BD, Babu-Khan S, *et al*. Beta-secretase cleavage of Alzheimer's amyloid precursor protein by the transmembrane aspartic protease BACE. Science 1999; 286(5440): 735-41.
[96] Wolf BA, Wertkin AM, Jolly YC, *et al*. Muscarinic regulation of Alzheimer's Disease amyloid precursor protein secretion and amyloid beta-protein production in human neuronal NT2N cells. J Biol Chem 1995; 270(9): 4916-22.
[97] Jiang S, Wang Y, Ma Q, Zhou A, Zhang X, Zhang YW. M1 muscarinic acetylcholine receptor interacts with BACE1 and regulates its proteosomal degradation. Neurosci Lett 2012; 515(2): 125-30.
[98] Mazanetz MP, Fischer PM. Untangling tau hyperphosphorylation in drug design for neurodegenerative diseases. Nat Rev Drug Discov 2007; 6: 464-79.
[99] Sun W, Qureshi HY, Cafferty PW, *et al*. Glycogen synthase kinase-3beta is complexed with tau protein in brain microtubules. J Biol Chem 2002; 277(14): 11933-40.
[100] Arif A. Extraneuronal activities and regulatory mechanisms of the atypical cyclin-dependent kinase Cdk5. Biochem Pharmacol 2012; 84(8): 985-93.
[101] Hitt BD, Jaramillo TC, Chetkovich DM, Vassar R. BACE1(-/-)mice exhibit seizure activity that does not correlate with sodium channel level or axonal localization. Mol Neurodegener 2010; 5: 31.
[102] Hu X, He W, Luo X, Tsubota KE, Yan R. BACE1 regulates hippocampal astrogenesis via the Jagged1-Notch pathway. Cell Rep 2013; 4(1): 40-9.
[103] Laird FM, Cai H, Savonenko AV, *et al*. BACE1, a Major Determinant of Selective Vulnerability of the Brain to Amyloid-β Amyloidogenesis, is Essential for Cognitive, Emotional, and Synaptic Functions. J Neurosci 2005; 25(50): 11693-709.
[104] Rajapaksha TW, Eimer WA, Bozza TC, Vassar R. The Alzheimer's β-secretase enzyme BACE1 is required for accurate axon guidance of olfactory sensory neurons and normal glomerulus formation in the olfactory bulb. Mol Neurodegener 2011; 6(1): 88.
[105] Mitani Y, Yarimizu J, Saita K, *et al*. Differential Effects between γ-Secretase Inhibitors and Modulators on Cognitive Function in Amyloid Precursor Protein-Transgenic and Nontransgenic Mice. J Neurosci 2012; 32(6): 2037-50.
[106] Timme CR, Gruidl M, Yeatman TJ. Gamma-secretase inhibition attenuates oxaliplatin-induced apoptosis through increased Mcl-1 and/or Bcl-xL in human colon cancer cells. Apoptosis 2013; 18(10): 1163-74.
[107] Ma XH, Shi Z, Tan C, *et al*. *In-silico* approaches to multi-target drug discovery. Pharm Res 2010; 27(5): 739-49.
[108] Xie H, Wen H, Zhang D, *et al*. Designing of dual inhibitors for GSK-3β and CDK5: Virtual screening and in vitro biological activities study. Oncotarget 2017; 8(11): 18118-28.
[109] Kumar A, Srivastava G, Sharma A. A physicochemical descriptor based method for effective and rapid screening of dual inhibitors against BACE-1 and GSK-3β as targets for Alzheimer's disease. Comput Biol Chem 2017; 71: 1-9.
[110] Chen Y, Liu Z-l, Fu T-m, Li W, Xu X-l, Sun H-p. Discovery of new acetylcholinesterase inhibitors with small core structures through shape-based virtual screening. Bioorg Med Chem Lett 2015; 25(17): 3442-6.
[111] Fang J, Li Y, Liu R, *et al*. Discovery of multitarget-directed ligands against Alzheimer's disease through systematic prediction of chemical-protein interactions. J Chem Inf Model 2015; 55(1): 149-64.
[112] Laursen L. IIBM debuts hyped 'cognitive cloud' biotech HQ in Cambridge. Nat Biotechnol 2015; 33(12): 1219-20.
[113] Smalley E. AI-powered drug discovery captures pharma interest. Nat Biotechnol 2017; 35(7): 604-5.