An Introduction to Machine Learning

Solveig Badillo[1,*,†], Balazs Banfai[1], Fabian Birzele[1], Iakov I. Davydov[1], Lucy Hutchinson[1], Tony Kam-Thong[1], Juliane Siebourg-Polster[1], Bernhard Steiert[1] and Jitao David Zhang[1]

In the last few years, machine learning (ML) and artificial intelligence have seen a new wave of publicity fueled by the huge and ever-increasing amount of data and computational power as well as the discovery of improved learning algorithms. However, the idea of a computer learning some abstract concept from data and applying them to yet unseen situations is not new and has been around at least since the 1950s. Many of these basic principles are very familiar to the pharmacometrics and clinical pharmacology community. In this paper, we want to introduce the foundational ideas of ML to this community such that readers obtain the essential tools they need to understand publications on the topic. Although we will not go into the very details and theoretical background, we aim to point readers to relevant literature and put applications of ML in molecular biology as well as the fields of pharmacometrics and clinical pharmacology into perspective.

The advent of data availability and growth of computational power, combined with the arrival of novel learning methods, has led to a number of breakthroughs in many scientific areas. This includes biological and clinical research, where applications range from molecular biology[1] to image data analysis[2] and clinical practice.[3] However, the idea of a computer learning some abstract concepts—like humans do constantly—has been around at least since the 1950s when the first neural networks[4] were developed. Even before that, other methods like Bayesian statistics and Markov chains were used with a similar idea in mind. Many of these methods are known to the pharmacometrics and clinical pharmacology community by different naming conventions. On the left, we indicate the machine learning terminology and, on the right, the usual statistics naming (based on Tibshirani https://statweb.stanford.edu/~tibs/stat315a/glossary.pdf):

- network, graphs ⇔ model
- weights ⇔ parameters
- learning ⇔ fitting
- generalization ⇔ test set performance
- supervised learning ⇔ regression or classification
- unsupervised learning ⇔ density estimation, clustering
- features ⇔ covariates or explanatory variables

The main difference to more traditional approaches lies very much in the two distinct cultures of statistical modeling. This has been eluded to nearly 2 decades ago by Breiman.[5] Here, we extend his definition by incorporating physiological models in one of the cultures. In particular, culture 1 involves specifying a model to describe the observed data, and culture 2 aims to solve the problem by taking an algorithmic modeling approach, thus inherently leading to models with a higher number of free parameters and complex interactions. This complexity can pose challenges to the interpretation of the model (so called "black box" problem). The approaches typically used in pharmacometric applications fall into culture 1, where an underlying model is assumed based on pharmacological principles and understanding of drug properties. Such models are usually physiologically interpretable. Most machine learning (ML) approaches fall into culture 2, where no explicit model is specified, and a computer is responsible for identifying associations in the observed data. These models tend to be difficult to interpret physiologically, however, significant progress was made over the years in the interpretability of ML models.[6,7] Today, many aspects of a black box model can be interpreted using proper tools.[8]

In this paper, we aim to support readers to develop the intuition needed to understand how computers can learn or help humans to identify patterns in data. The foundational ideas of ML are highlighted, but we do not describe the details and theoretical background of available ML methods. We point the interested readers to other articles or books, such as "The Elements of Statistical Learning"[9] (referred as ESL), and we refer to examples of their application in molecular biology, drug discovery, drug development, and clinical pharmacology.

We first introduce the concepts of data points, features, feature spaces, and similarity measures and then dive deeper into the two main domains of machine learning, namely unsupervised and supervised learning, touching key aspects and examples. In the case of unsupervised learning, computers are tasked to identify yet unknown patterns in data without pre-existing knowledge like groups or classes, whereas in the case of supervised learning, computers are tasked to learn how to predict the class or the value of yet unobserved data points based on a concept (often also called a "model") that has been derived from a training dataset. **Figure 1** shows a taxonomy of the different methods described in this paper and can be used as a reference, albeit nonexhaustive, on what scenario is suitable to apply which ML tool. Please note that all the unsupervised methods are also applicable in the case when labels are available.

[1]Pharmaceutical Sciences, Roche Pharma Research and Early Development (pRED), Roche Innovation Center Basel, Basel, Switzerland.
*Correspondence: Solveig Badillo (solveig.badillo@roche.com)
[†]S.B. was employed by Soladis Group during the time when the manuscript was written.

Received October 8, 2019; accepted January 15, 2020. doi:10.1002/cpt.1796

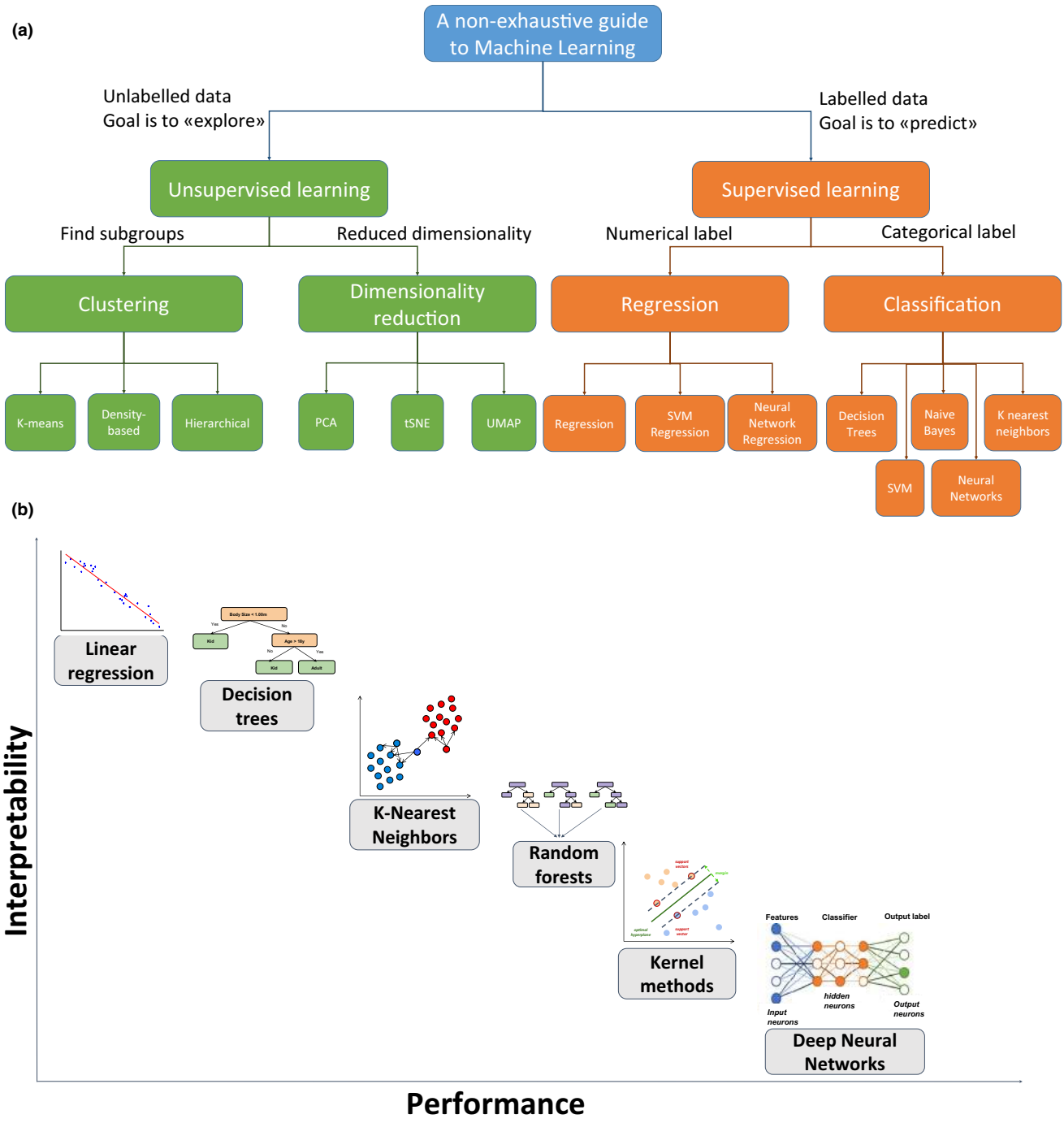

Figure 1 Taxonomy and overview of main machine learning (ML) algorithms. (**a**) Taxonomy of the different methods presented. (**b**) Overview of ML methods. The spectrum of available methods ranges from simpler and more interpretable to more advanced algorithms with potentially higher performance at the expense of less interpretability. Position of methods on the figure is qualitative and in practice depends on the number of free parameters, model complexity, data type, and the exact definition of interpretability used.[8] PCA, principal component analysis; SVM, support vector machine; tSNE, t-distributed stochastic neighbor embedding; UMAP, uniform manifold approximation and projection.

DATA AND FEATURES

In ML, we deal with data and datasets. A dataset is composed of multiple data points (sometimes also called samples), where each data point represents an entity we want to analyze. Therefore, a data point can represent anything like a patient or a sample taken from a cancer tissue. Many of the issues related to data are universal and affect not only ML approaches but any quantitative discipline, including pharmacometrics.

To compile the dataset, one has measured and collected a number of features (i.e., data that describe properties of the data points). Those features can be categorical (predefined values of no particular order like male and female), ordinal (predefined

values that have an intrinsic order to them like a disease stage), or numerical (e.g., real values). For a patient in a clinical setting, these could be (combinations of) the patient's demographics, disease history, results of blood tests, or more complex and high dimensional measures, like gene expression profiles in a particular tissue or all single nucleotide polymorphisms that represent the patient's unique genome.

Each feature represents one dimension of the feature space and the concrete value of a feature for a particular data point places the point in a defined place in this dimension of the space. Taken together, all the values of all features of a data point is called a feature vector. The more features we have collected for the dataset, the higher the dimensionality of the resulting feature vector and the feature space. Obviously, as the dimensionality increases, visualization of all dimensions of the feature space becomes intractable and we have to rely on the computer to identify the relevant patterns or have to apply dimensionality reduction methods, as explained later in the section "Dimensionality Reduction."

Clinical pharmacologists are usually familiar with longitudinal data, such as pharmacokinetic (PK) and pharmacodynamic (PD) profiles, where the time-dependency plays a central role. In fact, models used in pharmacometrics are based on equations that can be justified based on physiology and pharmacology, which yield insights into the time-evolution of the system. This is similar to, for example, physical problems, such as weather forecasts, where air flow and temperature lead to a certain temporal behavior of the system. In ML, including time as a distinguished continuous variable into respective algorithms, remains challenging and is an area of active research. As of now, several options exist to include time-dependent data in ML datasets: Either directly where each time point represents a feature, or via transformations, such as Fourier transform or B-splines, resulting in coefficients of basic functions that can be considered as features. Alternatively, Recurrent Neural Networks (RNNs) can be used to handle longitudinal data, as outlined in the section "Recurrent Neural Network." However, all these approaches have the limitation of—directly or indirectly—discretizing the time-dimension.

Most ML algorithms are designed to handle high-dimensional datasets. Hence, derived features from the existing data are often included, such as log-transformed data, products, and ratios of features, or more advanced combinations. Such data transformation is an important preprocessing step that can have a profound effect on the model performance. Therefore, it is always a good idea to use available domain knowledge and expertise to come up with relevant features, a process sometimes referred to as feature engineering.

Data quality plays a crucial role in ML. Carefully chosen ML methods and visual inspection defend against extreme values or outliers. Missing data, however, can be challenging. Not all the methods support data missingness, and again data transformation could be required as a preprocessing step in such cases. There are various ways to impute missing data, the performance of which depends on the dataset and the method used.[10] The most trivial approach to the imputation is to replace a missing value with the feature mean across all the samples where it is defined. This, however, sometimes can cause overfitting[11] (also see the section "Performance Measures and the Issue of Overfitting").

It is also essential to scrutinize any bias in the data (e.g., selection bias). Preferably, samples for the ML should be an unbiased random subset of the population. In practice, this is rarely the case, and there are some biases in the data. These biases can affect the ability of the model to generalize beyond the training dataset (and even the test dataset if both share a similar bias). An example of such a generalization problem is a model that is supposed to learn how to distinguish a wolf from a husky by animal characteristics, but eventually turns out to simply identify patches of snow on the photograph.[6] There are various approaches to mitigate bias (e.g., one could down-weight or completely exclude biased samples or features).[12] In particular, propensity scores are useful when estimating the effect of a therapeutic intervention.[13] Inspection of the feature importance provides valuable information about the magnitude and the effect of the bias,[6,7] which is recommend to be used for checking the trustworthiness of ML models.

Many clinical classification datasets are unbalanced, meaning that one or more classes are underrepresented. This could pose difficulties for many ML algorithms, including artificial neural networks and gradient boosting methods. One way to mitigate this problem is undersampling/oversampling the majority/minority class, respectively, or tweaking the misclassification cost in the objective function.[14]

Finally, for many applications, it is important to define a similarity or distance measure between two data points in the feature space. The simplest distance measure would be the Euclidean distance:

$$d(A,B) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \quad (1)$$

between the numerical feature vectors of two data points A and B, for features $i = 1 \ldots n$, but depending on the type of data we are dealing with there can be many other and sometimes much more complex distance or similarity measures, such as cosine similarity[15] or similarity scores of two biological sequences.[16]

Main takeaways

- Transforming input data and feature engineering may improve the model.
- Missing data requires imputation.
- Biases in the data should be scrutinized.
- Unbalanced datasets require amendment of the model.
- Meaningful measures of similarity between the samples should be defined.

UNSUPERVISED LEARNING

In exploratory data analysis, we often do not know the true "labels," or we might want to examine the naturally emerging patterns in the data. For this purpose, we can use unsupervised learning methods, like clustering, frequent pattern detection, and dimensionality reduction. Here, we will focus particularly

on clustering and dimensionality reduction as they have many applications in molecular biology and clinical practice.

Clustering

The goal of applying clustering methods is to identify relevant subgroups in a given dataset without having a predefined hypothesis on the properties subgroups might have. For example, in a cohort of patients with a particular disease, we might want to identify subtypes that represent distinct biological mechanisms driving the disease based on molecular measures taken.[17]

A cluster is a subset of the data which are "similar" to each other, whereas points belonging to different clusters are more "different." There are multiple approaches to clustering that use different underlying algorithms to group data points by their "similarity." All of them have advantages and disadvantages and needed to be selected carefully depending on the application and properties of the data.

One simple approach to clustering is *k*-means clustering.[18] Here, the number of clusters to be identified is predefined by a user-selected parameter *k*. Each cluster is represented by a cluster center, which is an artificial data point that represents the mean (or median) value of all points assigned to this cluster. In the beginning, *k* cluster centers, known as "seeds," are randomly placed in the feature space. The algorithm then iterates through two steps. In step one ("assignment"), data points are assigned to the cluster represented by the closest center. In step two ("center shift"), the position of each cluster center is updated based on the composition of the clusters after step one. After a number of iterations, this will usually converge to a local optimum where cluster assignments do not or only marginally change. The result of such a process is visualized in **Figure 2b**. Although the procedure is intuitive, its major drawback is that usually the clustering is strongly influenced by the value of *k*, and more often than not the true number of clusters in the data is unknown *a priori*. Because there is rarely a clear cut right or wrong answer in clustering, further cluster investigation is required to identify meaningful clusters, which can be challenging particularly in the light of a high-dimensional feature space.

Another group of methods for clustering is density-based clustering.[19] In density-based methods, a cluster represents a part of the feature space where data points are dense. Data points belonging to the regions of the feature space with low density are considered to be noise. One of the well-known density-based clustering algorithms is Density-Based Spatial Clustering of Applications with Noise.[20] Density-based clustering does not require a predefined value setting the number of clusters and provides a reproducible result. Further, it is able to also identify complex cluster shapes, like the one shown in **Figure 2c**.

In hierarchical clustering analysis, the goal is to build a hierarchy of clusters (ESL, chapter 14).[9] One simple approach to hierarchical clustering is neighbor joining. First, all pairwise distances between all data points in the dataset are computed. Later, in every step of an iterative process, the two data points with smallest distance are grouped together. This results in a tree-like cluster structure, as displayed in **Figure 2a** on the left side and top of the heatmap where the branch lengths of the tree represent the distances of samples. To arrive at a discrete set of clusters like with *k*-means a distance threshold has to be chosen at which the tree is cut horizontally. Again, there is no optimal way of selecting such a threshold and many reasonable solutions may exist. Hierarchical clustering can be used alone, or used in combinations with heatmaps (e.g., **Figure 2a**) to visualize selected or all features, for instance, gene expression data.

Dimensionality reduction

The number of features and, therefore, the dimensionality of the feature space can be very high with tens of thousands of measures per sample. Not only does this make data visualization challenging but also the analysis is challenging. In particular, analysis of high-dimensional datasets can be associated with a phenomenon known as the "curse of dimensionality,"[21] which refers to data sparsity and counterintuitive geometrical properties in high-dimensional

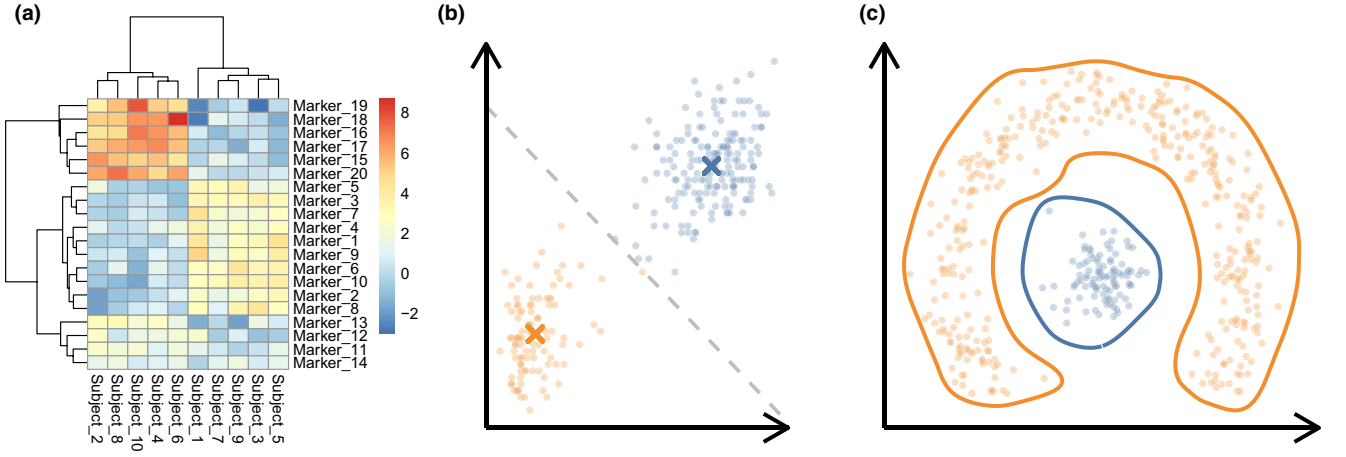

Figure 2 Overview of the results of different clustering approaches. (**a**) Shows the results of a two-dimensional hierarchical clustering. The two dendrograms visualize the similarity across samples and also across the markers measured. Such visualization is frequently used in biology for gene expression or other -omics technology readouts. (**b**) Shows the outcome of a classical clustering using *k*-means with a selected value of $k = 2$. Resulting clusters are usually convex and every point is assigned to one cluster, namely the one which is represented by the closest center point (marked by X). (**c**) Shows the result of a density-based clustering. Please note that the approach can identify nonconvex cluster forms, such as the orange cluster.

spaces. The "curse of dimensionality" poses challenges on most data analysis approaches, including but not limited to ML.

To mitigate such problems dimensionality reduction methods might be applied. Dimensionality reduction can aid data visualization by transforming each high-dimensional data point into two or more dimensions while keeping the majority of the variability and relative distances. Furthermore, dropping uninformative features could improve the model performance and convergence time. Although some of these methods, like principal component analysis, have even been developed long before the term ML has been coined,[22] others, like t-Distributed Stochastic Neighbor Embedding[23] or Uniform Manifold Approximation and Projection,[24] were developed recently and address complex challenges arising in data analysis. There is also a powerful neural network-based dimensionality reduction approach called autoencoder. For details on how to apply dimensionality reduction in biomedical data, we would like to refer the reader to a recent review.[25]

Examples of unsupervised ML applications

Clustering is widely used when analyzing high-dimensional data, such as transcriptomic, metabolomic, and proteomic experiments. Typically, hierarchical clustering would be used to identify main factors affecting the readouts as well as for identification of modules with high degree of coregulation. In single-cell sequencing, nonhierarchical clustering is used to understand which cell types are present in the sample. Clustering is also used to identify relationships among patients, tissues, diseases, or even disease symptoms.[26–29] Drug compounds themselves may also be clustered based on gene expression, sensitivity, and target protein properties[30–32] with the goal of guiding drug discovery.

Dimensionality reduction is routinely used in transcriptomic and other -omics experiments, usually to identify outliers and potential batch effects. In single-cell sequencing, Uniform Manifold Approximation and Projection or t-Distributed Stochastic Neighbor Embedding are used both for data visualization and for subsequent clustering.[24] Dimensionality reduction is also used to visualize the high-dimensional chemical space[33] or as a preprocessing step to improve performance of an ML model.[34]

Main takeaways

- Clustering can be used to understand structure in data by grouping similar observations together.
- *k*-means clustering is a simple yet powerful tool, however, the number of clusters must be specified in advance.
- Density-based methods do not require a prespecified number of clusters and allows identification of complex patterns in the data.
- Hierarchical clustering provides an overview of the relationship on multiple levels.
- Dimensionality reduction is used not only for data visualization but also to drop uninformative features.

SUPERVISED LEARNING

In a supervised learning problem, the computer is fed training data with observations and the corresponding known output values. The goal is to learn general rules (also often called a "model") that map inputs to outputs, so that it will be possible to predict the output for new unseen data, where we have observed input values but not their associated output.

There are two main categories of supervised learning: (i) classification where the output values are categorical, and (ii) regression where the output values are numeric.

In subsequent sections, the context of model fitting in supervised learning and the common issue of overfitting are introduced. Then, we explain how the performance is evaluated for classification and regression tools (i.e., how to assess the quality of mapping from inputs to outputs by the algorithm). This aspect is essential, as the merit of adopting ML methods often centers around the prospect of obtaining higher performance with the trade-off of interpretability. Understanding the different performance metrics enables better evaluation of the merits of a proposed model, as opposed to an assumption that an ML solution could always outperform a traditional approach.

We then dive into some of the existing classification and regression methods, starting off at the shallow end, where interpretation of the models is still straightforward, and progressing toward more ML-centric approaches where performance triumphs, often at the expense of interpretability. **Figure 1** summarizes the available spectrum of methods with respect to performance and interpretability. This section concludes with a nonexhaustive review of the applications of supervised learning methods in biology and, particularly, clinical pharmacology.

Performance measures and the issue of overfitting

The goal of a learning algorithm is to learn a concept or function (= a model) that describes the observed training data and is able to generalize on new independent data by avoiding both underfitting and overfitting.

The performance of a model is evaluated by methods that allow model assessment (i.e., estimating how well a given model performs in general and model selection; and the estimation of the performance of different models to choose the most adequate model). Some of these methods are highlighted in the next sections.

Model fitting. The model parameters are estimated based on observed data in the training set. To derive the optimal parameter values (e.g., for coefficients and weights), a distance measure between model and data is defined and minimized numerically. Independently of the metric chosen, the goal of model fitting is always to estimate the parameters by minimizing the distance, also called *loss function* or *cost function*, with two requirements:

- The model should provide predicted values that are close to observed ones on the training set, otherwise we say that it *underfits* and has a high *bias*.
- The model should generalize beyond the training set. A model that *overfits* predicts well on the training set but poorly on an independent test set, often because it is too complex for the data. In this case, we also talk about *high variance*.

In the following, we will call *objective function* any function that is optimized to estimate the model parameters.

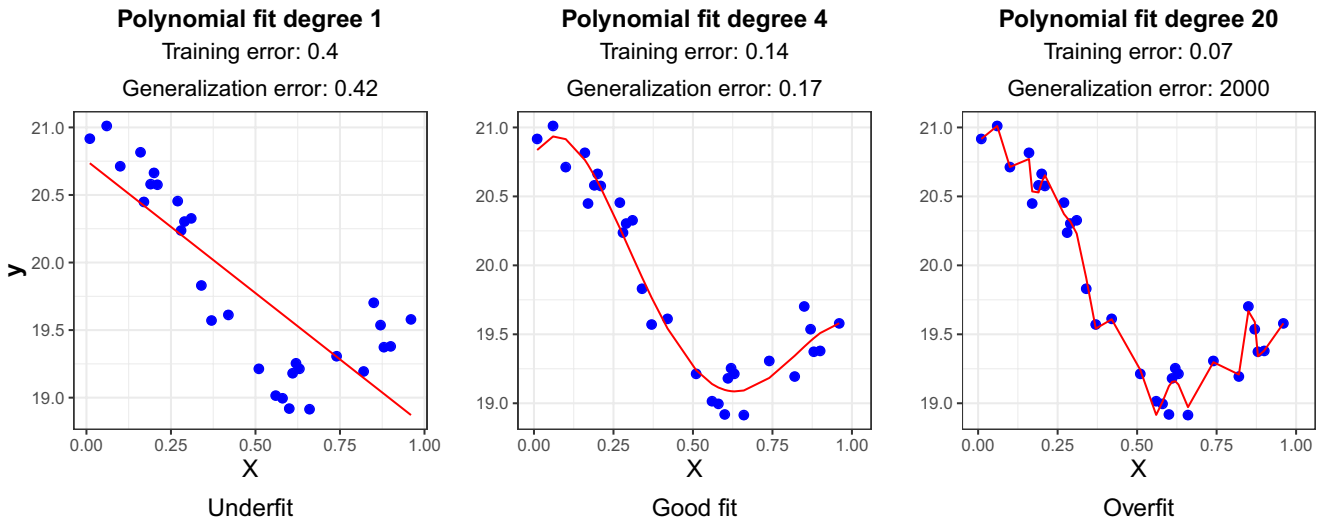

Figure 3 Illustration of the underfitting/overfitting issue on a simple regression case. Data points are shown as blue dots and model fits as red lines. Underfitting occurs with a linear model (left panel), a good fit with a polynomial of degree 4 (center panel), and overfitting with polynomial of degree 20 (right panel). Root mean squared error is chosen as objective function for evaluating the training error and the generalization error, assessed by using 10-fold cross-validation.

In the regression case, **Figure 3** illustrates the issue of underfitting and overfitting in the context of regression. Underfitting can occur when the model is too simple or when the features extracted from the data are not informative enough (**Figure 3**, left panel). Overfitting often occurs when the model is too complex or there are too many features over a small set of training examples (**Figure 3**, right panel).

This underfitting/overfitting issue is also often referred to as the bias/variance trade-off, which comes from the expression of the expected prediction error, including both bias and variance terms. The bias is an indication of the average error of the model for different training sets: It is the discrepancy between average of predicted values and the true mean we are trying to predict. The variance reflects the sensitivity of the model to the training set: For a given point, it corresponds to the spread of predicted values around their mean.

To minimize the predicted error, there is a trade-off between minimizing bias and variance: Increasing model complexity decreases bias but increases variance. To build less complex models, different techniques exist summarized under the term regularization. The principle consists in modifying the objective function by adding penalization terms that will influence parameter estimation. L1 and L2 regularization are the most common ones (ESL, sections 3.4.1 and 3.4.2).[9]

Different categories of loss functions. Different objective functions can be chosen to measure the distance between observed data and values predicted by the model. Some of the distance metrics used in practice can be associated to a *likelihood*. The likelihood indicates how probable it is to observe our data according to the selected model. The most common use of a likelihood is to find the parameters that make the model fit optimally to the data (i.e., the maximum likelihood parameter estimates). Usually, the negative logarithm of the likelihood is minimized and considered as objective function because it has favorable numerical properties. Similarly, in ML metrics, such as mean squared error, logistic objective, or cross-entropy, are used to find optimal parameters or assess the fitness of the model.

In practice, analytical calculation of maximum likelihood or minimal loss may not be feasible, and it is often necessary to use a numerical optimization algorithm to solve for the best parameter values. *Gradient descent* is such an algorithm, where we first define an objective function for which we want to minimize and then *iteratively* update the values of the parameters in the direction with the steepest decrease (first-order derivative) of the objective function until a convergence to a minimum distance is deemed reached. In the scenario of a nonconvex objective function, the success of finding a global minimum, as opposed to landing in some local minima, will depend on the choice of the initial set of parameter values, the learning rate (i.e., step size of each iteration) and the criterion for convergence. The reader can refer to ref. [35] for details on convex and nonconvex optimization processes. Stochastic gradient descent is an additional trick that can further speed up the optimization by randomly sampling a training dataset and summing the distances across this subset of training data points for approximating the objective function.

General principle of model selection and assessment. The problem of overfitting shows that the model performance on the training set is not a good indicator of its performance on a new dataset. We will detail below the principles of model performance evaluation in a supervised learning setting.

The general principle of model selection is as follows: When there are enough data, we separate them into three subsets—training, validation, and test sets. The training set is used to build different models, whereas the validation set is subsequently used to choose the algorithm and select the hyperparameters, if needed. Then, the model with the best performance on the validation set is selected. Finally, the test set enables to assess the generalization error, also called *test error*, which is the prediction error over a test dataset

that was not used during the training.[9] It is important to note here that the generalization error could be higher than expected when the original dataset is biased (see the section "Data and features"). Validating the model against a fully independent test dataset is the gold-standard method of assessing the generalizability of the model.

When the dataset is too small to extract a decent validation set, it is, for example, possible to use cross-validation techniques to select model hyperparameters. After putting aside a subset of the data for testing, k-fold validation consists of dividing the training set into k subsets, $k-1$ subsets being used for training and the last one to assess the performance. This process is repeated k times, each k subset being used once for validation, and the performance scores from each subset are then averaged for each set of hyperparameters to test. The k-fold cross-validation procedure is summarized in **Figure 4**. To choose between different learning algorithms[36] nested cross-validation can be used.

Indicators of model complexity vs. goodness of fit. In pharmacometrics, model selection is usually based on quantitative measures that summarize how well the model fits the data, often with penalties for overfitting. The most commonly used are the Akaike information criterion and Bayesian information criterion. They penalize the number of model parameters and reward goodness of fit, measured through likelihood. The Akaike information criterion is formalized as:

$$\mathrm{AIC} = 2M - 2\ln(\hat{\mathcal{L}}), \tag{2}$$

with the number of parameters M and the maximum likelihood $\hat{\mathcal{L}}$.

In contrast, the Bayesian information criterion:

$$\mathrm{BIC} = \ln(n)\cdot M - 2\ln(\hat{\mathcal{L}}), \tag{3}$$

takes into account the number of data points n.

These model selection approaches are rarely used in ML, partly due to the complexity of datasets and the associated violation of distributional assumptions. Instead, approaches like cross-validation are more commonly used (Clustering).

Performance measures for model assessment. For regression models, we typically use the mean squared error, or other types of average objective functions, to compare model performance on training and test set. For two-class classification problems, common performances measures are often derived from the "confusion matrix" shown in **Figure 5** and briefly described below.

- Precision, corresponding to the ratio of correctly predicted positive values to the total number of predicted positive values.
- Recall, also called true positive rate (TPR) corresponding to the ratio of correctly predicted positive values to the total number of positive values in the dataset.
- False Positive Rate (FPR), corresponds to the proportion of negative values predicted incorrectly.
- Accuracy, corresponding to the number of correctly predicted values divided by the total number of predicted values.
- Area under the ROC curve (AUC): Receiver operating characteristic (ROC) curves show the TPR (recall) and FPR dependence. In binary classification, each point on the ROC curve is located by choosing different thresholds for classification of y_i in positive or negative class. The top left corner of an ROC curve

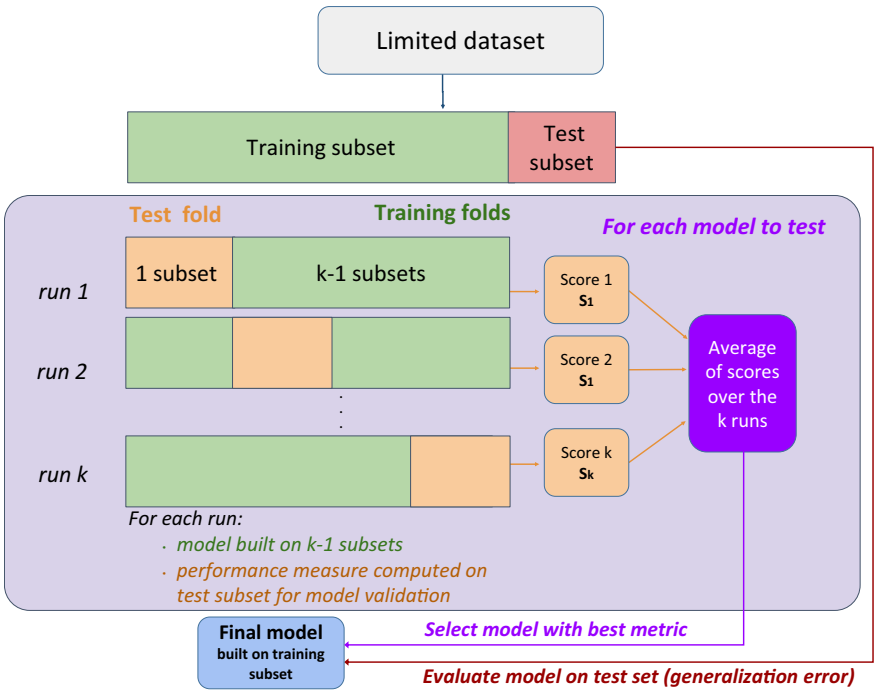

Figure 4 Illustration of the general principles of supervised learning in the case of a limited dataset. To assess the generalization ability of a supervised learning algorithm, data are separated into a training subset used for building the model and a test subset used to assess he generalization error.

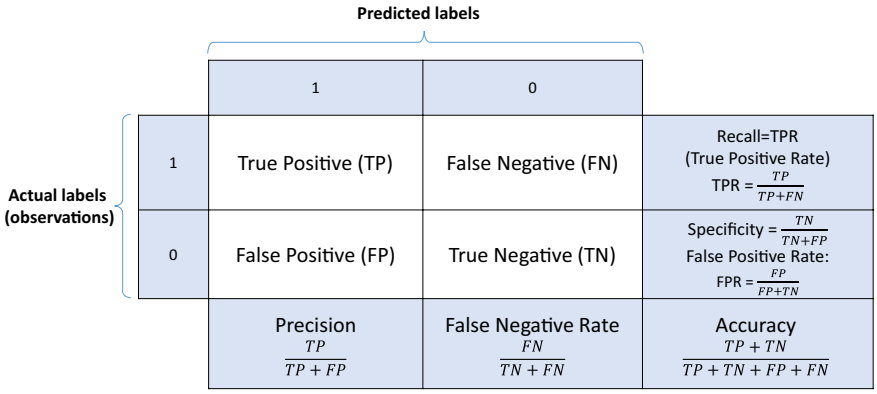

Figure 5 Confusion matrix for two-class problems. The confusion matrix indicates how successful the algorithm was at predicting labels in a binary classification problem where labels take values 0 (called "negative") or 1 (called "positive") by evaluating the predicted vs. the real labels. Every data point in the test set belongs to one of the four categories and different measures can be derived from these numbers.

is the ideal case with 100% of positive values correctly classified (TPR = 1) and 0% of positive values incorrectly predicted at 0 (FPR = 0). As it is ideal to maximize the TPR while minimizing the FPR, a larger area under the ROC curve (AUC) is better.

Some of these metrics could be generalized for multiclass problems, where there are more than two different labels in the dataset. However, the metrics mentioned above are noncontinuous with respect to model parameters, hence, parameter optimization may be challenging when they are used as objective function. A continuous alternative and widely used metric previously mentioned in the section "Model fitting" is cross-entropy (ESL, chapter 9),[9] which not only accounts for the most likely prediction but also for the prediction score (prediction confidence).

k-Nearest neighbors

We start our overview on existing learning methods with a method that skips the learning step completely and, therefore, does not lead to an explicit model that is being learned from the training data. As we will discuss later, this is also one of its biggest shortcomings. This type of learning is also often referred to as "instance-based learning" and, in our particular example, "*k*-nearest neighbor learning" (kNN).[37]

In these approaches, learning simply consists of storing all the existing, labeled data points (i.e., the training data) in a database. When a new, yet unclassified example is observed, the algorithm will place it in the *n*-dimensional feature space based on its feature values. For each data point in the database, we now compute the distance (e.g., a Euclidean distance or other, more complex ones) to this new data point in order to identify its *k* closest neighbors. In a second step, we examine the known labels of these *k*NNs in our database. Say we have chosen *k* to be nine and we observe seven of the nearest neighbors to be labeled as class X whereas two of them are labeled as class Y. In this case, we would assign our new data point to the class X as the majority of its neighbors are of this class. An extension of this simple approach would be to weight the importance of the neighbors to the classification by their distance to the new data point. Despite being very straightforward and simple, it proves to be a very effective classification method in practice. It is very efficient when it comes to training (i.e., storing the data in the database) and efficient implementations for computing the *k*NNs exist.

So, what are the challenges to this approach? The most obvious one is that because there is no "learning step," the *k*NNs approach does not identify the features that are really relevant to predict the class of a new case. Therefore, even though in a 20-dimensional feature space, where only 2 might be really relevant for the classification, the distance will be computed taking all 20 dimensions into account. Thus, the *k* nearest data points returned by the query will be highly influenced by irrelevant features or noise (see also "Dimensionality reduction" on how to remove some of those features). As a consequence, the resulting classification will be driven by noise rather than the real underlying pattern in the data. In this aspect, the approach suffers from the same challenge that also clustering approaches (see the section "Clustering") are facing, which are often summarized as the "curse of dimensionality."[21]

Naive Bayes

The second and very intuitive learning approach we would like to introduce is naive Bayes. It is based on computing simple statistics from a given training dataset as the learning step following a straightforward (but naive) application of the Bayesian formula for conditional probability in order to obtain a classification. Due to its simplicity it is also often used to obtain a baseline classification performance that other, more involved methods have to improve upon. It can best be explained by a simple example.

Let us assume we have training dataset with patients suffering either from a harmless cold or an influenza (flu) infection. We have measured two features for each patient, namely fever (high, low, or no) and pain (strong, low, or no). For each patient, we know through a laboratory test if the patient had an influenza infection or not. We now want to learn from these data and apply it to diagnose a new patient (where we have no laboratory test available) using the naive Bayes approach.

As a learning step, we count for each feature value how often it occurs in the influenza and in the cold patient group (e.g., to obtain the probability for high fever under the condition of the patient having a flu and so on). The result of this learning step might be seen in **Table 1**.

Table 1 Illustration of naive Bayes: Example of learning step results on flu dataset, showing the probabilities of features values given the patient category

Features	Fever			Pain		
Classes	High	Low	No	Strong	Low	No
Influenza (Flu)						
P (Flu) = 0.1	P(Fever = High|Flu) = 0.95	P(Fever = Low|Flu) = 0.05	P(Fever = No|Flu) = 0	P(Pain = Strong|Flu) = 0.75	P(Pain = Low|Flu) = 0.20	P(Pain = No|Flu) = 0.05
Cold						
P(Cold) = 0.9	P(Fever = High|Cold) = 0.1	P(Fever = Low|Cold) = 0.4	P(Fever = No|Cold) = 0.5	P(Pain = Strong|Cold) = 0.3	P(Pain = Low|Cold) = 0.3	P(Pain = No|Cold) = 0.4

Table 1 summarizes probability of each feature given the category of patient and shows that in the whole patient population the probability for a patient having an influenza infection is 0.1, whereas the probability for a normal cold is 0.9.

Once we have generated these values and, therefore, completed the "learning step" by analyzing our dataset, naive Bayes makes a now naive assumption, which is that all these features are conditionally independent of one another. In reality, this is rarely true and there are more advanced Bayesian learning methods that do not make this assumption. However, the assumption allows for a straightforward application of the Bayesian theorem. For details (i.e., formulas) on how to derive this classifier, we would like to refer the reader to further reading material (ESL, chapter 6).[9] In brief, the probability of a certain label (flu or cold) for a new test item can be computed as the product of the single conditional feature probabilities (fever and pain) that are observed for the data point times the probability for the class (flu or cold). The class with the maximal posterior likelihood is selected as the predicted class for the test item. Assuming we have a test person with an unknown diagnosis for influenza or cold, and we know that this person shows up with high fever and a high level of pain, we would compute the likelihood for influenza as:

$$P(\text{Fever} = \text{High}|\text{Flu}) \cdot P(\text{Pain} = \text{Strong}|\text{Flu}) \cdot P(\text{Flu}) = 0.95 \cdot 0.75 \cdot 0.1 = 0.07125. \quad (4)$$

In the same way we would compute the likelihood for a cold as:

$$P(\text{Fever} = \text{High}|\text{Cold}) \cdot P(\text{Pain} = \text{Strong}|\text{Cold}) \cdot P(\text{Cold}) = 0.1 \cdot 0.3 \cdot 0.9 = 0.027. \quad (5)$$

For a patient that presents to the doctor with high fever and strong muscular pain or headache, this results in a (nonnormalized) posterior probability for an influenza infection of 7.125% and in a probability of 2.7% for a normal cold. Therefore, the patient suffers more likely from a flu than from a cold.

In many aspects, naive Bayes, therefore, formalizes how humans might learn from experience.

Decision trees, random forests, and gradient boosting

Decision trees are an essential building block for many ML algorithms. They have been used for at least 50 years.[38,39] The idea behind decision trees is very intuitive and best represented in a visual form (e.g., **Figure 1**). Depending on the problem, decision tree leaf nodes have classes, probabilities, or continuous values in case of regression. In the early days of ML, decision trees have been used to solve pharmacological problems, such as dosing, toxicology, and diagnostics.[40–42] Although usage of decision trees is intuitive, the question is how to construct such trees from the available data. A few famous approaches worth mentioning are CART[43] and ID3.[44]

Currently, decision trees are almost never used in ML in their original form. One of the reasons being is the fact that decision trees are prone to overfitting. Nevertheless, decision trees became the building block for two widely used approaches: Random decision forests and gradient boosting frameworks.

Both random decision forests and tree-based gradient boosting use a set (ensemble) of trained decision trees to predict the outcome variable. The crucial difference between tree-based gradient boosting and random decision forests is on how trees are created.

In case of random forests, the algorithm constructs hundreds or thousands of deep decision trees ("strong predictors"). Each of those trees is likely overfitted, however, by combining the outputs of multiple trees we can solve the overtraining problem. On the contrary, in a gradient boosting algorithm, such as XGBoost or CatBoost, each of the trees is a shallow decision tree ("weak predictor"), and the algorithm iteratively decreases the classification error over time by adding more and more trees.

Today, gradient boosting methods show a great performance both in publications and ML competitions. Even without hyperparameter tuning, they usually provide excellent performance with a relatively low computational cost.[11] On the other hand, random forests are usually less prone to overfitting[45] and require less parameter tuning.[46] This makes random decision forests attractive for smaller datasets or as a baseline method for benchmarking.

Tree ensemble methods can be used for classification tasks, as well as for regression. In both cases, tree outputs are averaged, which can create a smooth output function.

Kernel methods: Support vector machines and regression

Kernel methods and, more specifically, support vector machine (SVM) for classification and support vector regression (SVR) for continuous output have found applications in computational biology for their ability to be robust against noise and to work with high-dimensional datasets found in genetics, transcriptomics, and proteomics.[47] Concretely in a more recent example, SVR was used for delineating cell compositions from bulk transcriptomics data.[48]

This section first offers a brief overview of the key concepts highlighting the notions of kernel transformations, an objective function with a lossless region, and a regularization term.[49,50] The emphasis will be placed on providing the reasoning behind why this is a more versatile method in dealing with multiple inputs where their effects on the output are unknown and can be postulated to span into nonlinear functions.

Background. Similar to all regression methods, the objective of SVR is to postulate a function on the input(s) that can help estimate for the observed output. Likewise, for SVM, the goal is to find the optimal decision boundary that separates the classes. As the name suggests, the core concept behind SVM/regression is the ability to objectively choose a subset of training data called *support vectors*. These support vectors define the model, which is usually a hyperplane in some feature space. To achieve this, several notions need to be introduced.

- An ε-*insensitive loss* function allows for residual less than ε, to be considered lossless and, thus, not part of the support vectors factored in to estimate the output-input function.
- A *regularization term* is added to the objective function with the aim of searching for a model to describe the relationship between the input and output variables such that the hyperplane is kept as flat as possible.
- *Slack variables* can be introduced to allow for training errors, termed soft margin, when the output is found outside the ε-insensitive region. By introducing slack variables, tolerance for the residual term to be greater than ε is made.
- A *kernel function* allows us to work in a higher dimension space, feature space. A kernel function applied in the input space corresponds to a dot product in the feature space where similarity measures are computed. This is achieved without having to explicitly map the input data from the input space to some feature space by some mapping function ϕ.

With all these concepts at hand, we are now capable of fitting a model with some thickness, known as a tube introduced by the ε-insensitive loss function, whereas the regularization term controls for the flatness of this hyperplane in some feature space defined by the kernel function. **Figure 6** illustrates these basic concepts of SVM.

Kernel trick and choice. SVR can capture nonlinear target functions, which map the multivariate inputs to the output. More precisely, the kernel trick means that a kernel:

$$k(\mathbf{x}_i,\mathbf{x}_j) := \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \tag{6}$$

applied to a set of inputs in the input space is equivalent to computing the dot product as a similarity measure in some feature space. This is achieved without having to explicitly perform a pre-mapping of the inputs, $\mathbf{x}_i$, with a mapping function Φ. A kernel function calculated in the input space corresponds to a dot product in some feature space if and only if it is a symmetric positive definite function.[51,52]

The choice of the radial basis function kernel,

$$\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle := k(\mathbf{x}_i,\mathbf{x}_j)_{\mathrm{RBF}} = \exp^{-\gamma(\|x_i - x_j\|)^2} \tag{7}$$

is often made as it can be expanded to a feature space of infinite dimensions. Although radial basis function covers a wide range

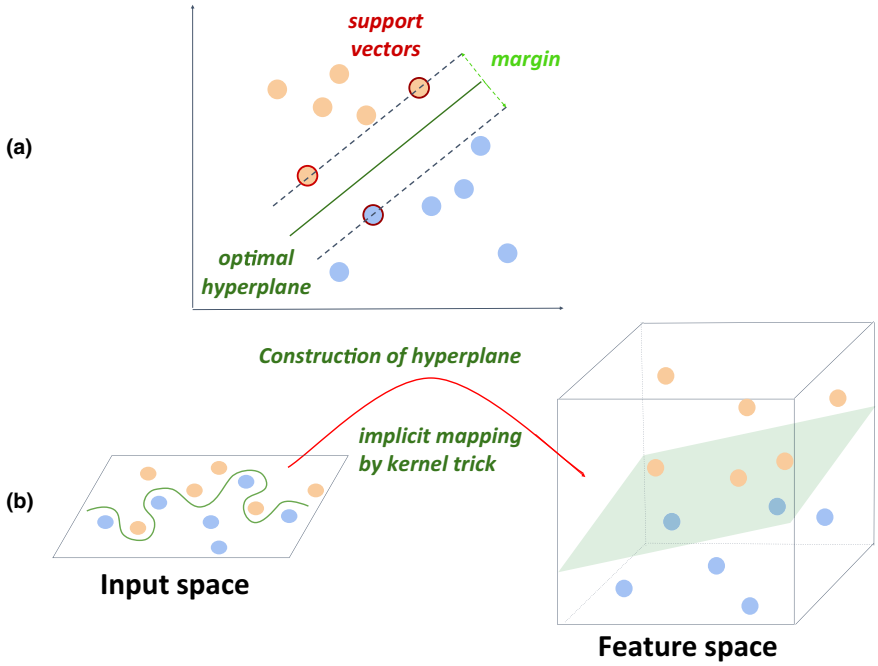

Figure 6 Illustration of support vector machine (SVM) principles. (**a**) Illustration of a simple case where hyperplane separate two groups directly in inputs space. (**b**) Illustration of performing nonlinear classification by implicitly mapping inputs into high-dimensional feature spaces where data points can be separated by a hyperplane.

of possible effects, it leads to harder interpretation of the eventual model. In practice, the selection of the kernel function is based on computational efficiency. Other popular kernels include linear and polynomial kernels.[47]

Neural networks

Background. Neural networks constitute a collection of neurons and edges, drawing its origins from circuit analysis. Different weights can be applied to each edge connecting the neurons. At each neuron, an activation function is applied to a weighed input signal to generate an output signal. A sigmoidal function is often used, consisting of a first order lowpass filter of a unit step function. Such sigmoidal function has the advantages of yielding bounded output and of being continuously differentiable, which is needed in the backward propagation step to tune the weights (parameters of the model), see steps defined in the section "Recurrent Neural Network."

Neurons are further subdivided into an input layer, hidden layer(s), and output layer, as shown in **Figure 7a**. The hidden layers perform the layer of abstraction needed to go from the input layer to the output layer. The number of hidden layers define whether the system is a shallow learning system (with one or a few hidden layer) or deep learning (with many hidden layers). There is an inherent trade-off between the number of hidden layers and time required to train the model. For this reason, although the core concept embedded in the neural network is not a novel one, it has found a resurgence of applications due to recent advances in computational power.

The most basic type is known as *feedforward* neural network, as information is just propagated from the input layer to the hidden layer(s) and finally to the output layer. The current state of the system is not defined by any past state; hence, it represents a memoryless system.

In the following, illustrative examples of neural networks are described: recurrent neural networks, long short-term memory networks, and gated recurrent networks. Further notable neural networks that are out of scope for this article but we recommend further reading on are convolutional neural networks,[53] encoder-decoder networks,[54,55] and generative models.[56]

Recurrent neural network. Recurrent neural networks are a class of neural networks dedicated to time series datasets as they factor in the inherent sequential relationship observed in the data of one time point to another. It has found success in what is known in the field as *sequential data*, where the order or time sequence of the signal plays a role, namely in natural language processing and time series forecasting. More closely related to our field of research, it has found application in predicting outcomes from electronic health records, where the richness comes inherently from the sequence correlation structure of the data to recommend swift and even anticipatory actions to be taken by the medical staff.[57] Rephrasing the question to solve a modeling conundrum in the pharmacometrics field is only starting to emerge at the time when this paper was drafted. Tang *et al*. present one of the rare attempts on how to use ML (here: RNNs) to characterize the PK of remifentanil and compared the results to the pharmacometrics gold-standard method NONMEM.[58] Although nonstandard PK models were used for the comparison and the generalizability of the results can be challenged, Tang *et al*. make a valuable contribution in exemplifying where RNNs could be used in pharmacometrics.

The basic form of an RNN is shown in **Figure 7b**, where each current state (at time t) is defined by a combination of the previous state of the system and the current input, which is similar to the concept of classical dynamic systems. The weights for each edge can be determined as to how far back to look into, similar to a time constant. Contrary to feedforward neural network, an identical weight is shared across in the individual neuron unit block across all the earlier discrete time steps.

At the core of the RNN, it consists of an input sequence defined by $x(t)$, an output sequence as defined by $o(t)$, a hidden or system state sequence as defined by $h(t)$, as well as a chained submodules of repeated units.

The steps needed to train an RNN model are as follows:

1. Define a network architecture and initialize the model with random weights and biases.
2. Perform a forward propagation to compute the estimated output.
3. Calculate the error at the output layer.

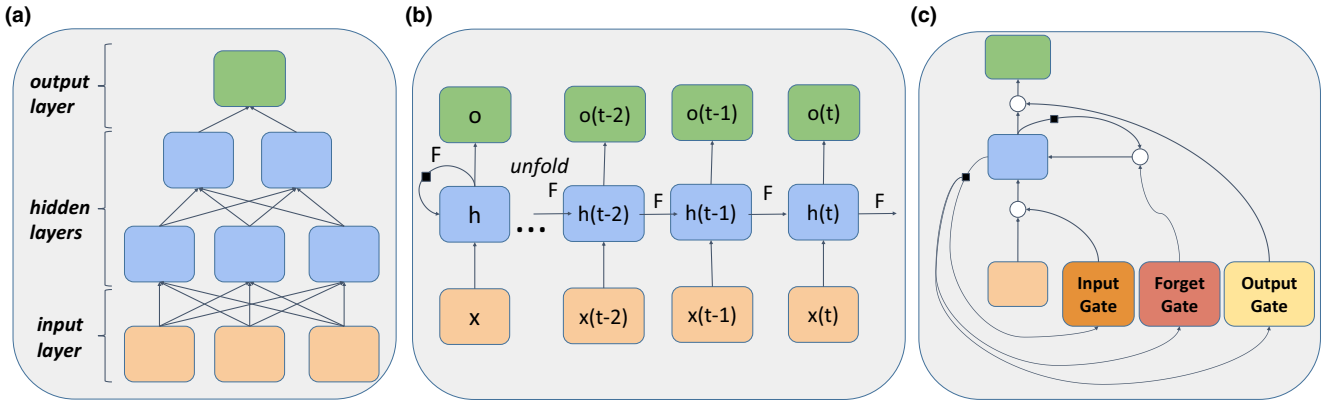

Figure 7 Neural networks. (**a**) Basics of feedforward neural networks. (**b**) Unfolding of recurrent neural networks. (**c**) Extensions of recurrent neural networks with gating units. Black square represents a delay of one discrete time step.

4. Perform a backward propagation to update the weights using an optimization approach.
5. Repeat steps 2–4 for the number of epochs (or iterations) until the loss function value is deemed minimized.

Extensions from this *Vanilla* RNN were developed to address the problems of unstable gradient problem (e.g., the vanishing gradient problem and the more serious counterpart of instability caused by an exploding gradient). These problems at their core are due to multiplications (under the influence of numerical errors) introduced in the backward propagation in relation of the error estimates to the parameters along each layer of the neural network. In other words, the vanishing gradient causes information that needs to be captured from a time point further away from the current time and, thus, renders the model weak to capture valuable stored memory with longer time lag. In the less common event that at least one partial derivative violates the requirement for stability, translating to the state matrix of having at least one eigenvalue > 1, this will lead to an exploding gradient problem, a known problem in traditional dynamic system for discrete time. The remit used to address this fundamental problem will be described more in two well-known extensions of RNN (long short-term memory (LSTM) and gated recurrent network (GRU)).

There has been many different variants and development in RNN research, each novel method serves to address a different problem ultimately leading to the development of more robust models. For example, to circumvent the unstable gradient problem, gradient clipping of forcing the gradient to a threshold has been proposed in ref. [59,60], but by far the most widely accepted method is the inclusion of *gating units*.

Long short-term memory and gated recurrent network. LSTM is part of a larger family of gated RNNs that retain and forget information with the introduction of gating units. More specifically, three gating units can be included in the system, as shown in **Figure 7c**. First, a direct copying or clearing of the state altogether can be controlled by the forget gate. A similar approach is also handled by the input gate to decide whether to include the current input signal as part of the update of the state. The amount of information to retain from the previous state signal and from the perturbation input signal is learned at each time step.[61] The system needs to learn long-term time dependencies by retaining information but it must also occasionally learn to clear information from its current state.[62] Consequently, solving the vanishing and exploding gradient problems. Finally, an output gate can be introduced, although less common, as a gating mechanism to decide which output signal gets fed back to the system.

A simpler rendition and, thus, faster training implementation can be found in GRU. GRUs address the same problem of unstable gradients and represent a new addition to this family of RNN extensions. The core difference between LSTM and GRU is that the latter omits the output gate and uses simpler reset and update gates.[63] In theory, however, LSTM should perform better as it can up-weigh or down-weigh information from longer time-distance/lag.

Examples of supervised ML applications in clinical pharmacology

Models in clinical pharmacology have typically been established by translating physiological and pharmacological principles to systems of differential equations and using expectation-maximization algorithms to estimate the model parameters. This mechanistically motivated approach has proven useful in many applications and is a well-established component of drug development programs. Potentially due to the success of these established approaches, only few examples of applying ML methods to clinical pharmacology problems exist up to now. Ryu *et al*. trained a deep neural network on a large curated database covering 192,284 drug-drug interactions in order to predict drug-drug and drug-food interactions for prescriptions, dietary recommendations, and new molecules.[64] Combining datasets from multiple studies to create large databases increases the potential to use ML to tackle broad clinical pharmacology questions.

ML has also been used to bridge drug discovery and clinical development. For example, Hammann *et al*. were able to predict incidence of adverse events from a molecule's chemical structure using a decision tree method.[65] Similarly, Lancaster and Sobie implemented SVMs to predict risk of Torsades de Pointes from *in vitro* data.[66]

In the area of personalized safety, ML has been used by Daunhawer *et al*. to personalize safety in the context of hyperbilirubinemia in neonates.[67] The authors used lasso and random forests to make predictions from clinical datasets. Furthermore, reinforcement learning was used by Gaweda *et al*. to personalize pharmacological anemia management.[68] A similar approach was used to develop a "closed loop" system for glucose control by combining a mathematical model, a glucose sensor, and a reinforcement learning model.[69] Chavada *et al*. and Hennig *et al*. investigated the feasibility of Bayesian feedback for dose adjustment of antibiotics.[70,71] The area of personalized healthcare could greatly benefit from using ML models that recommend dose adjustments in real time. In a recent study, an ML-type control algorithm was integrated with existing structural PK/PD models that are familiar to pharmacometricians and the resulting closed-loop control system was found to outperform a sensor-assisted pump.[69]

Main takeaways

- Supervised learning methods infer models based on labeled output-input pairs of the training dataset.
- Performance metrics are used to assess the classification and regression models to avoid overfitting of the training dataset.
- Many supervised learning methods exist with different trade-off between interpretability and performance.
- RNN is a special form of neural network that represents a dynamic system in discrete time.
- Examples of the applications of these supervised learning methods in computational biology and particularly clinical pharmacology are beginning to emerge.

DISCUSSION

In this tutorial, we have introduced some fundamental methods of ML that are likely to be of interest to the clinical pharmacology

and pharmacometrics community. Our brief introduction is supplemented with a range of relevant references. We have provided context by mentioning examples relevant to drug development. We conclude by summarizing how the fields of ML and clinical pharmacology are currently situated and by providing an outlook on how we expect to see further integration of the fields in the future. Advanced statistical methods are not new to pharmacometricians; in fact, such methods have been used to describe PK and PD phenomena for some time. For example, Bayesian methods are a well-established component of pharmacometric approaches.[72,73] It seems, therefore, likely that as statistical and ML approaches become more established and more prominent in the pharmaceutical industry, pharmacometricians will be among those who take advantage of these methods. Furthermore, new opportunities to investigate other clinical questions, such as patient stratification from high-dimensional baseline characteristics, may become possible in clinical pharmacology using ML approaches.

Several of the examples where ML approaches have been applied to clinical pharmacology questions include the integration of "classical" modeling techniques, such as specifying a structural model based on mechanistic understanding, and ML approaches.[69–71] Classical pharmacometric approaches are based on pharmacological principles that reflect hypotheses generated from the understanding of physiology and drug properties. It is unlikely that these models will be completely replaced by ML approaches in the near future. However, when the datasets and problems are more complex, many unknown influences and relationships exist and the focus is on interpolation and fast evaluation, pharmacometrics might benefit from applying ML-type methods. Going forward, we expect that fusing this understanding with ML models could lead to very effective models in the future. A recent perspective article provides more detail on applications of ML in clinical pharmacology.[74]

In the age of big data, there are many new opportunities for ML in clinical pharmacology. For example, data generated from wearable devices pose new challenges on how they can be linked to PK data in the future. In addition, access to real-world data could provide strong evidence for covariates, supplement control datasets, and bolster models that have been trained on small datasets.

In pharmacometric approaches, a predictive model is typically established by integrating a structural model and relevant data. The structural model substantially constrains the solution space and, therefore, relatively little data are required to fit the model. On the contrary, in neural networks, model structure is not prespecified and, thus, comparatively much more data are required for building a predictive model. It is also important to note that we are still very much at the infancy stage of understanding at which point the merger of larger data with these novel ML methods can be beneficial for performance as compared with more traditional methods. The following challenge[75] on time series forecasting shows that combinations of classic statistical and ML methods produce the most accurate forecasting and, thus, suggest it as a way forward. One of the main drivers of success of the pharmacometric approaches is that the models include a thorough understanding of the processes of drug absorption, distribution, metabolism, and elimination. The established models are highly predictive and, thus, find wide use in supporting drug development. Due to this success, despite the arrival of ML, classical pharmacometrics approaches are not expected to decrease in importance and activity. In contrast, they can be enhanced and improved by knowledge and insight distilled by ML methods and models.

An ongoing challenge for members of the clinical pharmacology community who wish to use ML methods is the inherent prevalence of longitudinal data. So far, there are many ML methods that rely on baseline features to make predictions, but relatively few examples where longitudinal data are used.

Overall, we expect that there will never be a universal, one-size-fits-all approach to which modelers from different fields converge. We note that there are many areas of potential synergy where modeling fields overlap in the remit of drug development. The clinical pharmacology community will continue to base their analyses on pharmacological principles and will gradually build in new ML elements to their workflow, strengthening their models further. In addition, the clinical pharmacology community will be able to enhance the range of questions they are able to address by using ML approaches.

FUNDING

I.I.D. is a recipient of the Roche Postdoctoral Fellowship. This study is funded by F. Hoffmann-La Roche Ltd.

CONFLICT OF INTEREST

All authors declared no competing interests for this work.

1. Camacho, D.M., Collins, K.M., Powers, R.K., Costello, J.C. & Collins, J.J. Next-generation machine learning for biological networks. *Cell* **173**, 1581–1592 (2018).
2. Shen, D., Wu, G. & Suk, H.-I. Deep learning in medical image analysis. *Annu. Rev. Biomed. Eng.* **19**, 221–248 (2017).
3. Rajkomar, A., Dean, J. & Kohane, I. Machine learning in medicine. *N. Engl. J. Med.* **380**, 1347–1358 (2019).
4. Kleene, S.C. Representation of Events in Nerve Nets and Finite Automata (RAND Project Air Force, Santa Monica, CA, 1951) <https://apps.dtic.mil/docs/citations/ADA596138>.
5. Breiman, L. Statistical modeling: the two cultures (with comments and a rejoinder by the author). *Stat. Sci.* **16**, 199–231 (2001).
6. Ribeiro, M.T., Singh, S. & Guestrin, C. "Why should I trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA*, August 13–17, 2016.
7. Štrumbelj, E. & Kononenko, I. Explaining prediction models and individual predictions with feature contributions. *Knowl. Inf. Syst.* **41**, 647–665 (2014).
8. Lipton, Z.C. The mythos of model interpretability. *ACM Queue* **16**, 31–57 (2018).
9. Hastie, T., Tibshirani, R. & Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference and Prediction* (Springer New York Inc., New York, NY, 2008).
10. Jerez, J.M. *et al*. Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artif. Intell. Med.* **50**, 105–115 (2010).

11. Dorogush, A.V., Ershov, V. & Gulin, A. CatBoost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363* (2018).
12. Cortes, C., Mohri, M., Riley, M., Rostamizadeh, A., Freund, Y., Györfi, L., Turán, G. & Zeugmann, T. (eds.) Sample selection bias correction theory. In *Algorithmic Learning Theory*. 38–53 (Springer, Berlin, Heidelberg, 2008).
13. Lee, B.K., Lessler, J. & Stuart, E.A. Improving propensity score weighting using machine learning. *Stat. Med.* **29**, 337–346 (2010).
14. Newby, D., Freitas, A.A. & Ghafourian, T. Coping with unbalanced class data sets in oral absorption models. *J. Chem. Inf. Model.* **53**, 461–474 (2013).
15. Hu, L.-H., Huang, M.-W., Ke, S.-W. & Tsai, C.-F. The distance function effect on k-nearest neighbor classification for medical datasets. *Springerplus* **5**, 1304 (2016).
16. Borozan, I., Watt, S. & Ferretti, V. Integrating alignment-based and alignment-free sequence similarity measures for biological sequence classification. *Bioinformatics* **31**, 1396–1404 (2015).
17. Collisson, E.A., Bailey, P., Chang, D.K. & Biankin, A.V. Molecular subtypes of pancreatic cancer. *Nat. Rev. Gastroenterol. Hepatol.* **16**, 207–220 (2019).
18. Lloyd, S. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **28**, 129–137 (1982).
19. Kriegel, H.-P., Kröger, P., Sander, J., Zimek, A. Density-based clustering. *Wiley Interdisc. Rev Data Min. Knowl. Discov.* **1**, 231–240 (2011).
20. Ester, M., Kriegel, H.-P., Sander, J. & Xu, X.A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. 226–231 (AAAI Press, Portland, OR, 1996).
21. Zimek, A., Schubert, E. & Kriegel, H.-P. A survey on unsupervised outlier detection in high-dimensional numerical data. *Stat. Anal. Data Min.* **5**, 363–387 (2012).
22. Pearson, K. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **2**, 559–572 (1901).
23. van der Maaten, L. & Hinton, G. Visualizing data using t-SNE. *J. Mach. Learning Res.* **9**, 2579–2605 (2008).
24. Becht, E. *et al*. Dimensionality reduction for visualizing single-cell data using UMAP. *Nat. Biotechnol.* **37**, 38–44 (2019).
25. Nguyen, L.H. & Holmes, S. Ten quick tips for effective dimensionality reduction. *PLoS Comput. Biol.* **15**, e1006907 (2019).
26. Wang, C., Machiraju, R. & Huang, K. Breast cancer patient stratification using a molecular regularized consensus clustering method. *Methods* **67**, 304–312 (2014).
27. Cooper, G.S., Bynum, M.L. & Somers, E.C. Recent insights in the epidemiology of autoimmune diseases: improved prevalence estimates and understanding of clustering of diseases. *J. Autoimmun.* **33**, 197–207 (2009).
28. Walsh, D. & Rybicki, L. Symptom clustering in advanced cancer. *Support. Care Cancer* **14**, 831–836 (2006).
29. Genotype-Tissue Expression (GTEx) Consortium *et al.* The genotype-tissue expression (GTEx) pilot analysis: multitissue gene regulation in humans. *Science* **348**, 648–660 (2015).
30. Zhang, J.D., Berntenis, N., Roth, A. & Ebeling, M. Data mining reveals a network of early-response genes as a consensus signature of drug-induced in vitro and in vivo toxicity. *Pharmacogenomics J.* **14**, 208–216 (2014).
31. Gemma, A. *et al*. Anticancer drug clustering in lung cancer based on gene expression profiles and sensitivity database. *BMC Cancer* **6**, 174 (2006).
32. Koch, M.A. & Waldmann, H. Protein structure similarity clustering and natural product structure as guiding principles in drug discovery. *Drug Discov. Today* **10**, 471–483 (2005).
33. Reutlinger, M. & Schneider, G. Nonlinear dimensionality reduction and mapping of compound libraries for drug discovery. *J. Mol. Graph. Model.* **34**, 108–117 (2012).
34. Ezzat, A., Wu, M., Li, X.-L. & Kwoh, C.-K. Drug-target interaction prediction using ensemble learning and dimensionality reduction. *Methods* **129**, 81–88 (2017).
35. Nesterov, Y. *Lectures on Convex Optimization*. Vol. **137** (Springer, Berlin, Germany, 2018).
36. Cawley, G.C. & Talbot, N.L.C. On over-fitting in model selection and subsequent selection bias in performance evaluation. *J. Mach. Learn. Res.* **11**, 2079–2107 (2010).
37. Mitchell, T.M. *Machine Learning* (McGraw-Hill Inc., New York, NY, 1997).
38. Belson, W.A. Matching and prediction on the principle of biological classification. *J. R. Stat. Soc. Ser. C Appl. Stat.* **8**, 65–75 (1959).
39. Krischer, J.P. An annotated bibliography of decision analytic applications to health care. *Oper. Res.* **28**, 97–113 (1980).
40. Shortliffe, E.H., Buchanan, B.G. & Feigenbaum, E.A. Knowledge engineering for medical decision making: A review of computer-based clinical decision aids. *Proc. IEEE* **67**, 1207–1224 (1979).
41. Bach, P.H. & Bridges, J.W. A decision tree approach for the application of drug metabolism and kinetic studies to in vivo and in vitro toxicological and pharmacological testing. *Arch. Toxicol. Suppl.* **8**, 173–188 (1985).
42. Jordan, T.J. & Reichman, L.B. Once-daily versus twice-daily dosing of theophylline: a decision analysis approach to evaluating theophylline blood levels and compliance. *Am. Rev. Respir. Dis.* **140**, 1573–1577 (1989).
43. Breiman, L., Friedman, J., Olshen, R. & Stone, C. Classification and regression trees. *Wadsworth Int. Group* **37**, 237–251 (1984).
44. Quinlan, J.R. Induction of decision trees. *Mach. Learn.* **1**, 81–106 (1986).
45. Breiman, L. Random forests. *Mach. Learn.* **45**, 5–32 (2001).
46. Segal, M.R. Machine Learning Benchmarks and Random Forest Regression. Technical Report, (Center for Bioinformatics & Molecular Biostatistics, University of California, San Francisco, CA, 2003).
47. Ben-Hur, A., Ong, C.S., Sonnenburg, S., Schölkopf, B. & Rätsch, G. Support vector machines and kernels for computational biology. *PLoS Comput. Biol.* **4**, e1000173 (2008).
48. Newman, A. *et al*. Robust enumeration of cell subsets from tissue expression profiles. *Nat. Methods* **12**, 453–457 (2015).
49. Vapnik, V. & Lerner, A. Pattern recognition using generalized portrait method. *Automat. Rem. Contr.* **24**, 774–780 (1963).
50. Smola, A.J. & Schölkopf, B. A tutorial on support vector regression. *Stat. Comput.* **14**, 199–222 (2004).
51. Mercer, J. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. R. Soc. Lond. B Bio. Sci.* **415–446**, (1909).
52. Schölkopf, B. & Smola, A.J. *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond* (The MIT Press, Cambridge, MA, 2002).
53. Krizhevsky, A., Sutskever, I. & Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems* **25**, 1097–1105 (2012).
54. Cho, K.*et al*. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).
55. Cho, K., Van Merriënboer, B., Bahdanau, D. & Bengio, Y.On the properties of neural machine translation: encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).
56. Kingma, D.P., Mohamed, S., Rezende, D.J. & Welling, M. Semi-supervised learning with deep generative models. NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems, **2**, 3581–3589 (2014).
57. Choi, E.*et al*. Using recurrent neural network models for early detection of heart failure onset. *OUP Academic* (2016) <https://academic.oup.com/jamia/article/24/2/361/2631499>
58. Tang, J.-T., Cao, Y., Xiao, J.-Y. & Guo, Q.-L. Predication of plasma concentration of remifentanil based on Elman neural network. *J. Cent. South Univ.* **20**, 3187–3192 (2013).
59. Pascanu, R., Mikolov, T. & Bengio, Y. On the difficulty of training recurrent neural networks. International conference on machine learning, Edinburgh, June 26–July 1, 1310–1318 (2013).

60. Bengio, Y., Boulanger-Lewandowski, N. & Pascanu, R. Advances in optimizing recurrent networks. IEEE International Conference on Acoustics, Speech and Signal Processing, Kyoto, March 25–30, 8624–8628 (2012).
61. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).
62. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT Press, Cambridge, MA, 2016) <http://www.deeplearningbook.org>
63. Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. Workshop on Deep Learning in NIPS, Montreal, December 8–13, (2014).
64. Ryu, J.Y., Kim, H.U. & Lee, S.Y. Deep learning improves prediction of drugdrug and drugfood interactions. *Proc. Natl. Acad. Sci.* **115**, E4304–E4311 (2018).
65. Hammann, F., Gutmann, H., Vogt, N., Helma, C. & Drewe, J. Prediction of adverse drug reactions using decision tree modeling. *Clin. Pharmacol. Ther.* **88**, 52–59 (2010).
66. Lancaster, M.C. & Sobie, E. Improved prediction of drug-induced torsades de pointes through simulations of dynamics and machine learning algorithms. *Clin. Pharmacol. Ther.* **100**, 371–379 (2016).
67. Daunhawer, I. *et al*. Enhanced early prediction of clinically relevant neonatal hyperbilirubinemia with machine learning. *Pediatr. Res.* **86**, 122 (2019).
68. Gaweda, A.E. *et al*. Individualization of pharmacological anemia management using reinforcement learning. *Neural Networks* **18**, 826–834 (2005).
69. Benhamou, P.-Y. *et al*. Closed-loop insulin delivery in adults with type 1 diabetes in real-life conditions: a 12-week multicentre, open-label randomised controlled crossover trial. *Lancet Dig. Health* **1**, e17–e25 (2019).
70. Chavada, R., Ghosh, N., Sandaradura, I., Maley, M. & Van Hal, S.J. Establishment of an $AUC_{0\text{-}24}$ threshold for nephrotoxicity is a step towards individualized vancomycin dosing for methicillin-resistant staphylococcus aureus bacteremia. *Antimicrob. Agents Chemother.* **61**, e02535–16 (2017).
71. Hennig, S., Holthouse, F. & Staatz, C.E. Comparing dosage adjustment methods for once-daily tobramycin in paediatric and adolescent patients with cystic fibrosis. *Clin. Pharmacokinet.* **54**, 409–421 (2015).
72. Dansirikul, C., Morris, R.G., Tett, S.E. & Duffull, S.B. A Bayesian approach for population pharmacokinetic modelling of sirolimus. *Br. J. Clin. Pharmacol.* **62**, 420–434 (2006).
73. Lunn, D.J., Best, N., Thomas, A., Wakefield, J. & Spiegelhalter, D. Bayesian analysis of population PK/PD models: general concepts and software. *J. Pharmacokinet. Pharmacodyn.* **29**, 271–307 (2002).
74. Hutchinson, L. *et al*. Models and machines: how deep learning will take clinical pharmacology to the next level. *CPT Pharmacomet. Syst. Pharmacol.* **8**, 131–134 (2019).
75. Makridakis, S., Spiliotis, E. & Assimakopoulos, V. The M4 competition: results, findings, conclusion and way forward. *Int. J. Forecast.* **34**, 802–808 (2018).