KJA
Korean Journal of Anesthesiology

Data science and machine learning in anesthesiology

Dongwoo Chae

Department of Pharmacology, Yonsei University College of Medicine, Seoul, Korea

Review Article

Korean J Anesthesiol 2020;73(4):285-295
https://doi.org/10.4097/kja.20124
pISSN 2005-6419 • eISSN 2005-7563

Received: March 19, 2020
Accepted: March 21, 2020

Corresponding author:
Dongwoo Chae, M.D. Ph.D.
Department of Pharmacology, Yonsei University College of Medicine, 50-1 Yonsei-ro, Seodaemun-gu, Seoul 03722, Korea
Tel: +82-2-2228-1741
Fax: +82-2-313-1894
Email: dongy@yuhs.ac
ORCID: https://orcid.org/0000-0002-7675-3821

Machine learning (ML) is revolutionizing anesthesiology research. Unlike classical research methods that are largely inference-based, ML is geared more towards making accurate predictions. ML is a field of artificial intelligence concerned with developing algorithms and models to perform prediction tasks in the absence of explicit instructions. Most ML applications, despite being highly variable in the topics that they deal with, generally follow a common workflow. For classification tasks, a researcher typically tests various ML models and compares the predictive performance with the reference logistic regression model. The main advantage of ML lies in its ability to deal with many features with complex interactions and its specific focus on maximizing predictive performance. However, emphasis on data-driven prediction can sometimes neglect mechanistic understanding. This article mainly focuses on the application of supervised ML to electronic health record (EHR) data. The main limitation of EHR-based studies is in the difficulty of establishing causal relationships. However, the associated low cost and rich information content provide great potential to uncover hitherto unknown correlations. In this review, the basic concepts of ML are introduced along with important terms that any ML researcher should know. Practical tips regarding the choice of software and computing devices are also provided. Towards the end, several examples of successful ML applications in anesthesiology are discussed. The goal of this article is to provide a basic roadmap to novice ML researchers working in the field of anesthesiology.

Keywords: Artificial intelligence; Data science; Electronic health record; Machine learning; Predictive analytics; Risk score system.

Introduction

Data science and machine learning (ML) are topics that have received enormous attention in recent years. Their strongest driving force seems to be in the explosion of data, both in size and scope. Several success stories of applying ML technology have increased worldwide investment in this area [1], and medicine is no exception.

As with other medical fields, ML is revolutionizing anesthesiology research. Unlike classical research methods that are largely inference-based, ML is more geared toward making accurate predictions [2]. Medical textbooks are full of well-validated risk factors but only few well-validated predictive systems.

Being aware of this limitation, medical researchers and societies have strived to create effective predictive algorithms [3]. A widely employed method for clinical prediction is the development of a risk scoring system. The researcher cleverly puts together previously known risk factors to yield a single output score. The Child–Pugh score is a well-known example. Based on five clinical measures of total bilirubin, serum albumin, prothrombin time, ascites, and hepatic encephalopathy, a score ranging from 5 to 15 is calculated. Ranges of 5 to 6, 7 to 9, and 10 to 15 points are equivalent to the Child–Pugh classes of A, B, and C, respectively. This system was first proposed in 1964 by the surgeon Charles

Gardner Child [4], and modified by Pugh et al. in 1972.

A scoring system is a big step forward toward implementing an actual prediction algorithm based on multiple factors. However, the methods of constructing such a system have been rather haphazard and based on trial and error. ML has much to offer in this regard.

Much of the recent hype about artificial intelligence (AI) and ML mainly centers around a specific ML algorithm called *deep learning*. To a newcomer in this field, it might seem that AI and deep learning are equivalent terms. However, AI, as an academic field, has existed for a long time and deep learning (a.k.a. multi-layer perceptron, the old-fashioned term) is one of its many research topics. Most of its important theoretical development already took place in the mid-twentieth century but its enormous potential was unrecognized for a long time.

The landscape of ML can be envisioned as fundamentally consisting of three large categories: *supervised learning, unsupervised learning*, and *reinforcement learning* (Fig. 1). Algorithms that constitute each of these categories are diverse, and deep learning is merely one of them. Deep learning is special in certain aspects and supersedes all other algorithms when dealing with tasks related to images, videos, sounds, and machine translation. However, deep learning is occasionally overstated in its capability and misconceived as a magic wand. It is necessary to gain a clear overview of the entire field of ML before digging deeper into any specific algorithm.

This article will primarily deal with data science and ML as applied to anesthesiology using electronic health records (EHRs). Owing to their retrospective character, the main limitation of EHR-based studies is the difficulty of establishing causal relationships. However, the low cost and rich information content provide great potential to uncover hitherto unknown correlations. Such correlations can generate hypotheses to be tested through prospective clinical trials.

Basic concepts of ML

ML is a field of AI concerned with developing algorithms and models to perform prediction tasks in the absence of explicit instructions. ML algorithms build a predictive model based on some *training data* that consist of *features*. A good understanding of these terms is crucial in acquiring a firm grasp of ML. Hereafter, important terms will be highlighted in *italics*.

Data

The data D in a ML task is expressed using a matrix $\begin{bmatrix} x_{11} \cdots x_{1m} \\ \cdots \\ x_{n1} \cdots x_{nm} \end{bmatrix}$. The i^{th} row of X is called an ith *instance* of D. The j^{th} column of X, on the other hand, is called a j^{th} *feature* of D. Throughout this review, we will use a well-known dataset in the ML community (the Pima Indian diabetes dataset), originally from the National Institute of Diabetes and Digestive and Kidney Diseases (https://www.niddk.nih.gov/), which consists of 768 subjects and 8 features. These features are as follows: number of prior pregnancies, plasma glucose concentration (glucose tolerance test), diastolic blood pressure (mmHg), triceps skin fold thickness (mm), 2-h serum insulin (mu U/ml), body mass index (BMI), diabetes pedigree function, and age (Table 1). The objective is to predict whether the patient is diabetic, based on the values of these features.

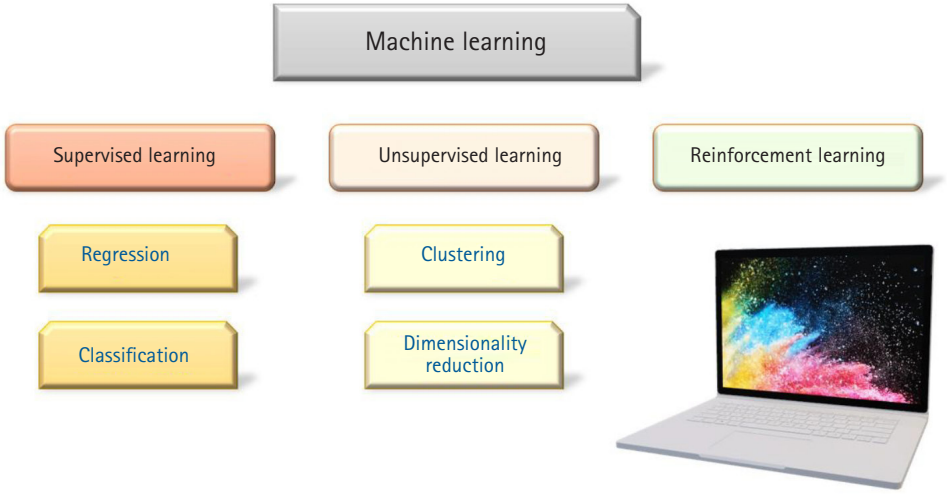

Fig. 1. Machine learning (ML) Landscape. ML algorithms can roughly be divided into three categories – Supervised learning, unsupervised learning, and reinforcement learning. Our focus will primarily be on supervised ML algorithms.

https://doi.org/10.4097/kja.20124

Task

A task is a problem we want to solve—the most common ones being *classification* and *regression*.

In classification, the *target* variable (or *label*) is an element of a countable set S = $\{C_1, \ldots, C_k\}$. In our running example, S has two elements {diabetes mellitus (DM) absent, DM present}. For notational convenience, we assign 0 and 1 to each element, respectively. The role of the ML algorithms is to discover an appropriate mapping from the features to {0, 1}. Such a mapping is called a *model*.

In regression, the target variable is a real number (often restricted to be positive). The task, therefore, is to construct a model that generates a real-numbered prediction that is as similar to the target variable as possible.

Model and parameters

A model takes in features as inputs, performs some logical or mathematical operations, and generates an output. For example, a model can consist of a set of if-then rules such as:

If age > 60 yr and BMI > 30 then output = 1 else output = 0

The cutoff values, 60 yr and BMI of 30, are called *parameters* of the model. Given the model, optimal parameter values must be found such that the outputs are as close to the actual target values. For various reasons, ML algorithms instead try to find parameters that yield the least dissimilar outputs to the target variables. The metric of dissimilarity is called a *loss function*. Depending on the specific task, different loss functions are chosen. For our classification task, we could imagine assigning a value of +1 if there is a mismatch between the output and the target variable, and 0 otherwise. The sum of the assigned values of all instances divided by the number of instances is then defined as the *expected loss*. The parameters that yield the smallest magnitude of this value become the solution of this task.

The above-mentioned set of if-then rules can be combined into a tree, yielding a tree model (Fig. 2). The *decision* tree algorithm is a popular ML sequence that provides an intuitive framework to make classifications and decisions. Often, multiple trees are generated on random subsets of the original data. The decisions of the individual trees are then combined to generate the final prediction. This algorithm is known as the *random forest* algorithm [5].

The practical use of ML involves choosing the right model to apply to data. There is no single model that works best for every

Table 1. Pima Indians Diabetes Dataset with 768 Subjects and 8 Features

Pregnancies (number)	Glucose (mg/dl)	Blood pressure (mmHg)	Skin thickness (mm)	Insulin (mu U/dl)	BMI (kg/m^2)	Diabetes pedigree function	Age (yr)	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0

The ninth column is the target to be predicted. BMI: body mass index. Available from https://www.kaggle.com.

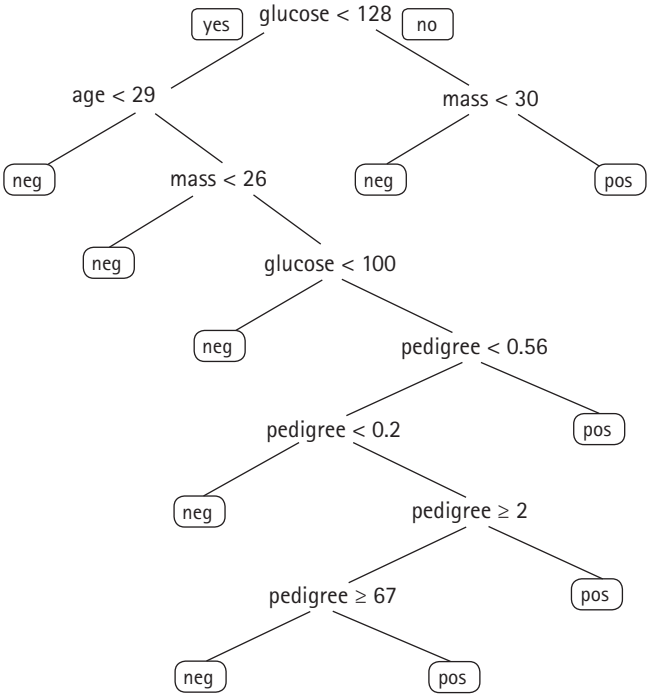

Fig. 2. A decision tree to predict the presence of diabetes based on the Pima Indians Diabetes dataset (glucose: serum glucose [mg/dl], age [yr], mass: body mass index [kg/m^2], pedigree: diabetes pedigree function). neg: negative, pos: positive.

task, a fact that is referred to as the *no free lunch theorem* of ML [6]. It is usually required to try multiple models and find one that works best for a task. This trial and error process lies at the heart of ML artistry. The list of commonly used ML algorithms is as follows:

- Regularized linear regression
- Logistic regression
- Discriminant analysis
- Support vector machine
- Naïve Bayes
- K-Nearest Neighbor
- Decision tree
- Ensemble method
- Neural network

Researchers generally fit different ML algorithms to data and compare the predictive performances. The one that yields the highest performance is generally chosen as the final predictive model.

A specific approach called the *ensemble method* is noteworthy. As the term implies, this method combines several ML algorithms into a single model, and can be approximately divided into two types: *sequential* and *parallel* methods. In the former, the base learner is trained sequentially, each time assigning more weights to previous errors. Prototypical algorithms are *AdaBoost* and *Gradient* Boost [7,8]. Parallel methods, on the other hand, combine learners that have been trained independently. The above-mentioned random forest algorithm is the prototype.

Loss functions

The choice of the loss function is intricately related to the task and the model type.

In regression, the loss function is formulated so that it reflects the *expected error*. The most commonly used loss function is the *mean squared error* (MSE), defined as $\frac{1}{n}\sum(y_i\text{-}f(X_i))^2$, where, y_i and $f(X_i)$ are the target variable and the model output of the i^{th} instance, respectively. *Root-mean-square error* (RMSE), which is defined simply as $\sqrt{MSE}$, is also widely used. The latter has an advantage that the scale of the error is the same as that of the target variable. It is possible to state that the model generates outputs with an error margin of ε (%), where, $\varepsilon = 100(\%)\times\sqrt{MSE}/(mean\ prediction)$.

In classification tasks, the model output is a number between 0 and 1, often interpreted as the probability of the target y_i being positive. *Binary cross entropy*, defined as

$$-[y_i log p_i + (1\text{-}y_i)\log(1\text{-}p_i)]$$

where, p_i is the model output for the ith instance, is widely used as the loss function.

Optimization algorithms are used to search the parameter space of a given model so that the loss function is minimized. There are various algorithms, the most common one being based on calculating the *gradient* of the loss function. It is an iterative search algorithm where the parameter values are updated every time based on the calculated gradient of the current parameter estimates. A gradient is a multivariate extension of a *derivative*. Since the loss function, L, is a function of its parameters p, changing the values of p results in changes in L. Intuitively, changing p in the direction that most sensitively affects L is likely to be an effective search strategy. To illustrate this idea more vividly, imagine a landscape where the value of L is plotted along the vertical axis and the parameters (restricted to two-dimensional for demonstrative purposes) on the horizontal plane (Fig. 3). Supposing that a certain agent is riding along the ridges of the loss function surface, searching for its deepest valley, starting from any arbitrary point on the surface, a plausible search strategy would be to look around and descend down the path with the steepest slope. A tangent with the steepest slope, the derivative in a univariate problem, is the gradient. Repeating such a search often results in the agent ending up in one of the valleys, which is a potential solution to the search task. Unfortunately, the gradient-based search strategy

https://doi.org/10.4097/kja.20124

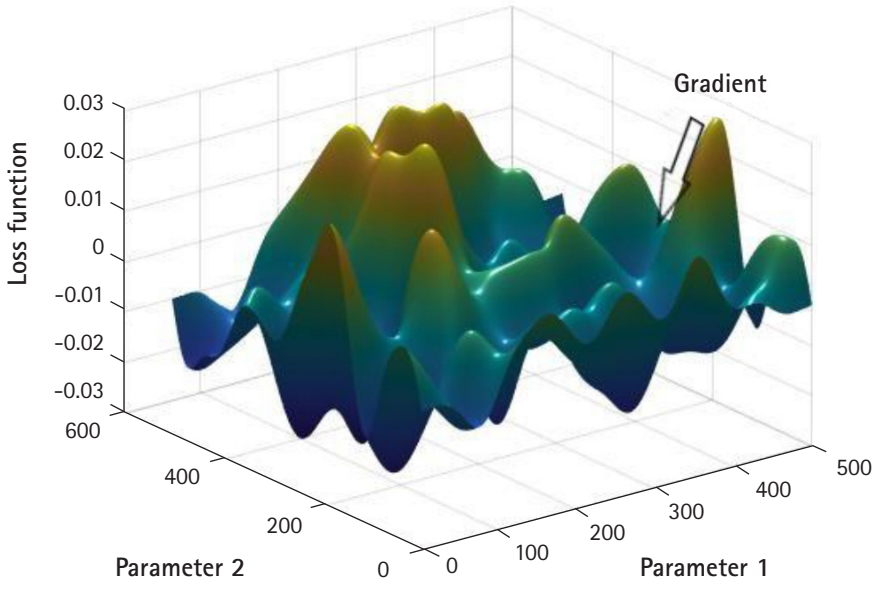

Fig. 3. A hypothetical landscape illustrating the relationship between the loss function and the parameters.

does not guarantee a globally optimal solution when applied to a loss function surface with multiple peaks and valleys, as shown in Fig. 3. Methods to circumvent this limitation do exist (such as introducing a momentum, starting the search from multiple initial parameter values, adopting a stochastic component, and so on) but none of them definitely solve the problem (other than the brute-force strategy of searching all possible parameter combinations). For more information, refer to [9].

Overfitting and underfitting

An important limitation associated with minimizing a loss function given a set of features, target variables, and a model is that this does not necessarily minimize the difference between f(X) and the true signal $\tilde{Y}$. The optimized prediction $\hat{Y} = f(X)$ may be closest to $Y = \tilde{Y} + \varepsilon$ (where, ε is a random error) but not to $\tilde{Y}$. This phenomenon is given an infamous name called *overfitting* in ML literature.

To avoid overfitting, one often keeps a separate dataset with an identical signal $\tilde{Y}$ but different random error ε'. Overfitting can be detected by comparing the loss function based on the original dataset with that calculated using the separate dataset. A significant difference between the two loss functions indicates that the parameters are fitting the random error ε.

Training, validation, and test datasets

In ML nomenclature, the original dataset used to optimize the parameters is called the *training dataset*, and a separate dataset used for detecting overfitting is called the *test dataset*. Hereafter, training and test datasets will be denoted as Tr and Te, respectively. It is customary in ML practice to randomly split D into Tr and Te. The split ratio is often chosen such that the size of Tr is greater than Te.

While this practice partly safeguards against overfitting, repeated bouts of training can lead to the overfitting of Te as well. Hence, Tr is generally split yet again into (real) training and *validation* (V) *datasets* (Fig. 4).

In analogy, V serves the purpose of practice exams. Optimization on Tr is followed by validation on V. When there is room for improvement, either a different model is chosen, or model hyper-parameters are modified. The procedure of training the new or modified model is repeated until a satisfactory result on V is achieved. The final predictive model is then validated using Te. If the predictive performance on Te is comparable to that on Tr, the model is then accepted and used.

A variant of this procedure, called *k-fold cross-validation*, is also widely used. First, Tr is partitioned into k chunks (often of equal or similar sizes). One of the k chunks is defined as V and the rest as Tr. Then, the predictive performance on V is assessed, and this

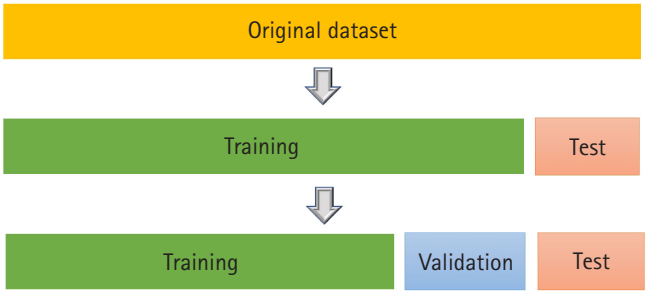

Fig. 4. Splitting of the dataset into training, validation, and datasets. Model comparison is done using the validation dataset. Predictive performance of the final model is assessed using the test dataset.

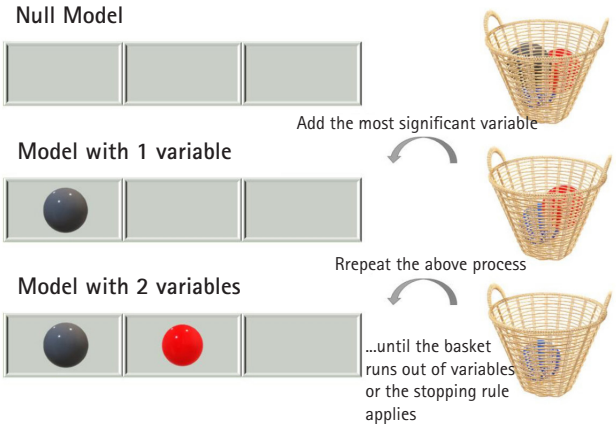

Fig. 5. The stepwise selection procedure. Each time, the most significant variable is incorporated into the prediction model. Such a selection process is repeated over all candidate variables until no more candidates exist or the algorithm hits a predefined stopping rule.

procedure is repeated on all possible allocations of V and Tr. Finally, the k assessment scores from the k validation runs are averaged to yield the *mean performance index*. Models that improve the mean performance index are chosen.

While splitting D into Tr and V, and Te is the standard method to tackle overfitting, one should always try to directly incorporate the *data generating mechanism* if possible. For example, to predict the concentration of an anesthetic drug that is known to be eliminated from the kidney by first order kinetics, it would be a better choice to use a pharmacokinetic model $C(t) = C(0)e^{(-kt)}$ (k: elimination rate constant) than to use a high degree polynomial $C(t) = C(0) + \beta_1 t + + \beta_2 t^2 \ldots + \beta_p t^p$. The latter function can undoubtedly generate outputs that match the observed concentrations given a sufficiently high degree p. However, this model would be prone to overfitting the data.

Feature selection

Not all features are informative in predicting the target. Returning to our running example of the Pima Indians Diabetes dataset, supposing that in addition to the 8 features, we are given the favorite movie genre of the subjects. This additional feature, however, is unlikely to improve predictive performance because movie preference is not related to the development of diabetes in any meaningful way.

A more serious problem occurs when so-called collinear features are present. Suppose the data consist of weights measured in kilograms and pounds as two distinct columns (i.e., features). If a classification model is built using both features, the optimization algorithm would fail to converge. This is because the relative contribution of each of the two features to the model output cannot be uniquely determined.

First, the investigator must inspect the features manually and filter them based on domain knowledge. Here, features, such as patient names (which have no information in predicting the target of our interest) are excluded.

After manual filtering of the features, *feature selection algorithms* can be applied. There are basically three different classes of such algorithms: (1) *stepwise selection* methods that iteratively incorporate the "best" feature at each step, (2) *dimensionality reduction* techniques that extract the most important components to be used as new features, and (3) *regularization* or *shrinkage* methods that penalize large parameter values during the optimization process.

The stepwise selection method is an umbrella term for all approaches that selects the one best feature at a time, evaluates the gain in predictive performance, and then decides whether it should be included in the model (Fig. 5). While stepwise selection algorithms do not always provide the optimal solution, they are easily understandable and simple to implement.

Dimensionality reduction techniques eliminate collinearity by lumping correlated features. For example, given features of height and weight, a new feature of BMI can be calculated from the two. Replacing height and weight with BMI reduces the number of features and at the same time solves the problem of feature collinearity. One of the most popular dimensionality reduction methods is the *principal components analysis*. Given multiple features, the method identifies the principal components that retain as much of the original variance as possible. For more information on this particular method, refer to [10].

Lastly, regularization controls the magnitude of regression coefficients. This is based on the premise that large magnitude coefficients (in the order of tens to thousands) are unlikely. From a Bayesian point of view, regularization is equivalent to imposing a null hypothesis of non-significance to all features (i.e., zero coefficient). Large coefficients that deviate from zero are thus penalized

https://doi.org/10.4097/kja.20124

in the calculation of the loss function. The three widely used regularization schemes are called *LASSO*, *Ridge*, and *ElasticNet* [11]. A more interested reader is referred to a review article that specifically deals with various feature selection methods [12].

Multilayer neural network algorithms, so-called *deep learning*, can automatically extract useful features from the raw input. This is one of the great advantages of this algorithm and has contributed to its high popularity.

Assessment of predictive performance

Training models is important; however, assessing their performance and selecting the best model is perhaps even more important. This is particularly true for medical researchers since ML engineers can be hired to do the computational work but judging the overall quality of the work remains the responsibility of the principal investigator.

A better model is the one that makes fewer errors. In regression, such a model is the one associated with a lower MSE. In classification, a model with a higher accuracy, defined as the fraction of correctly classified instances, might be used. However, more subtle complications arise when there is an imbalance among the classes. Referring to our running example, suppose that 99% of the subjects were diabetic. In such a case, simply predicting all subjects as diabetic would achieve an accuracy score of 99%.

A *confusion matrix* is a table showing the frequencies of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) (Table 2).

Several important metrics are defined based on the values of the confusion matrix:

(i) *Sensitivity* (a.k.a. recall, true positive rate) = $\frac{TP}{(FN+TP)}$

(ii) *Specificity* (a.k.a. selectivity, true negative rate) = $\frac{TN}{(TN+FP)}$

(iii) *Precision* (a.k.a. positive predictive value) = $\frac{TP}{(FP+TP)}$

(iv) *Negative predictive value* = $\frac{TN}{(TN+FN)}$

The most important concept related to the application of the confusion matrix is the tradeoff between sensitivity and specificity. The most sensitive algorithm is the one that predicts everything positive, i.e., the sensitivity is 100%. Such a simple method guarantees that the algorithm classifies every positive instance correctly (all instances are classified as positive anyway). However, the price paid is that no negative instance is classified correctly, yielding a specificity of 0%. Since the definition of positives and negatives is arbitrary, it is easy to imagine that by flipping their definitions, a 100% specificity can be achieved at the cost of 0% sensitivity.

Table 2. A Confusion Matrix

	Prediction = 0	Prediction = 1
Actual target = 0	TN	FP
Actual target = 1	FN	TP

TN: true negative, FP: false positive, FN: false negative, TP: true positive.

As an analogy, an anticancer drug that kills all cells will certainly eradicate all cancer cells. The sensitivity is 100%. However, since this drug would kill off all healthy cells as well (i.e., specificity 0%), such a drug will never make it to the clinic.

One must find an optimal point where both sensitivity and specificity are acceptable for practical use. Evidently, this is no longer a mathematical problem. If sensitivity is more important (e.g., cancer diagnosis), the cost associated with the lower specificity can be tolerated.

Now, suppose model A achieves 90% sensitivity and 10% specificity while model B achieves 10% sensitivity and 90% specificity. Which of the two models is the better?

The widely adopted solution is to first draw a curve called a *receiver operating characteristic* (ROC) curve, which is created by plotting sensitivity against 1 – specificity, and then calculate the *area under the curve* (AUC). An ideal algorithm that achieves 100% sensitivity and 100% specificity would be associated with an AUC of 100%, which is the maximum score achievable. A random guess, due to the tradeoff between sensitivity and specificity, would satisfy the following equality:

Sensitivity + Specificity = 100%

The ROC curve is then a straight line with slope of unity that passes through the origin, and its AUC is 50% Fig. 6A).

Most predictive models have AUC values that fall between these two extremes: 50% and 100%. Model selection, therefore, is often carried out by comparing the AUC of ROC curves Fig. 6B).

The use of AUC, however, is not without problems. Lobo et al. [13] recommend against the use of AUC for the following reasons: it ignores the predicted probability values, summarizes the test performance over regions of the ROC space in which one would rarely operate, weights omission and commission error equally, does not inform about the distribution of model errors, and most importantly, the total extent to which models are carried out highly influences the rate of well-predicted absences and the AUC scores.

A widely used alternative to the AUC is the F1 score, defined as the harmonic mean of precision and sensitivity:

$$F1 = \frac{2TP}{2TP+FP+FN}$$

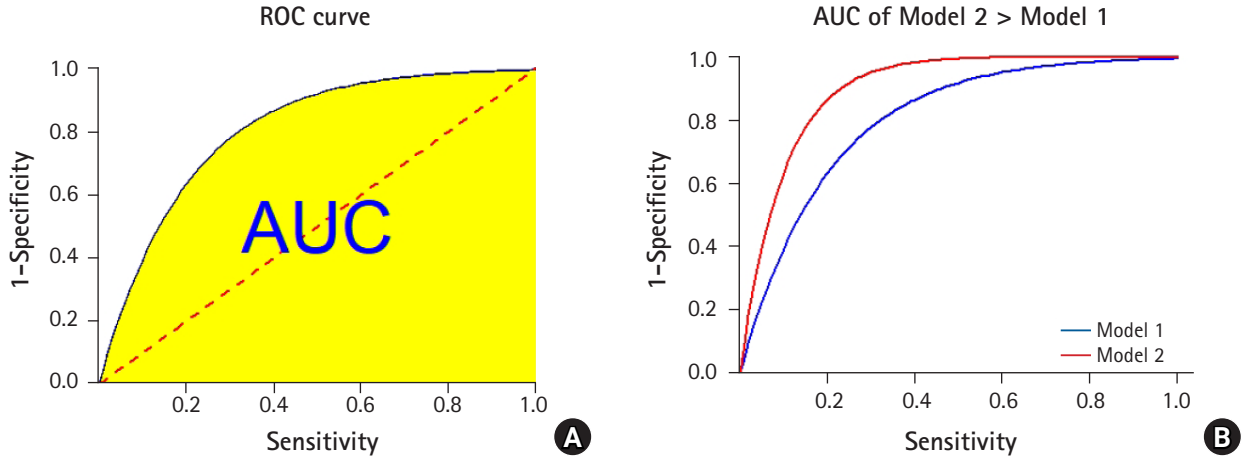

Fig. 6. The receiver operating characteristic (ROC) curve and the area under the curve (AUC). (A) The yellow shaded area corresponds to the AUC. The dotted red line corresponds to the ROC curve of a random guess algorithm. (B) A model associated with a higher AUC is generally taken to be a better model.

Notwithstanding these issues, AUC is currently the standard method widely used for model comparison and assessment of predictive performance.

Summary

Supervised ML requires that the data be expressed as a matrix. Each row corresponds to an instance of the learning material, while each column represents the values of the different features. The matrix is subsequently fed into a suitable model that consists of tunable parameters. The machine carries out the learning process by minimizing a predefined loss function. All loss functions reflect the degree to which the model outputs differ from the true values, playing the role of a *supervisor*.

Different models are trained on Tr and their predictive performances assessed on V. A metric such as the AUC of the ROC curve must be chosen prior to the analysis as the model selection criterion. Models are then compared using this metric, where the model yielding the highest (or lowest) value is selected as the final model. The predictive performance of the final model is judged using Te. If the performance reasonably matches that of Tr and V, one can be assured that overfitting has not occurred to any significant degree. The final model is then deployed as a real-world application.

Tools and software

R

R (https://www.r-project.org) is a free software environment for statistical computing and graphics. It supports Windows, MacOS, and a wide variety of UNIX platforms. The current stable release version is 3.6.3 (Holding the Windsock), but updates occur frequently. Users are recommended to periodically check their software version. Its main advantage is the wide third-party support. Virtually any kind of statistical analysis can be performed by searching and downloading relevant packages. It is light and consumes little computer memory.

Most R users also use RStudio (https://rstudio.com), an integrated development environment (IDE) for R. It significantly facilitates the use of R by providing an interactive console, a syntax-highlighting editor, a command history, an environment of defined variables, and a plotting window. It is available both as free open source and commercial licenses. However, the free open source license is fully functional, and operates on Windows, Mac, and Linux. Furthermore, RStudio Online is available for cloud computing.

Its main drawback is the relatively slow computation time compared to other software. Unless the data to be processed is in the range of gigabytes to terabytes, this is not going to be a critical issue. Some third-party packages solve this problem by using codes that run on faster compiled languages (such as C++).

Python

Python (https://python.org) is another leading software used by ML researchers. The latest version is 3.8.2, but this would soon change as further updates are made. Similar to R, it is developed under an open source license and can be downloaded freely. It is the language of choice for first time programmers owing to its

https://doi.org/10.4097/kja.20124

gentle learning curve, but this does not mean that it is limited in its scope and function. Top IT firms (such as Google) are strong proponents of Python, and Google has developed one of the most popular deep learning application programming interfaces (APIs), *TensorFlow*, on which it runs.

The counterpart of RStudio for Python is Anaconda (https://anaconda.com), a highly popular IDE used by the majority of Python ML researchers. The Individual Edition operates under the open source license and can be freely downloaded. The main advantage of using Anaconda is that most ML-related third-party packages (or *libraries*), such as *Scikit-learn*, *NumPy*, *Pandas*, and *SciPy*, come pre-installed. The need for manually installing the required libraries is thus minimized. Anaconda also includes one of the most revolutionary projects that changed the practice of ML research, i.e., *Jupyter* notebook (https://jupyter.org). Project Jupyter is a non-profit, open-source project that started in 2014. It evolved to support interactive data science and scientific computing, and supports Python, R, and other programming languages. It operates on a web-browser, where codes can be embedded as cells, greatly facilitating the presentation, sharing, and collaboration of ML research. It is currently the primary presentation format used by Kaggle (https://kaggle.com), a popular platform for launching public ML competitions and sharing public data.

Others

MATLAB (https://mathworks.com/products/matlab.html) is the *de facto* standard programming language for engineers. It is a commercial software developed by MathWorks (https://mathworks.com), and offers fast and reliable tools for all areas of scientific computing. Third-party tools can also be shared through MATLAB central (https://mathworks.com/matlabcentral), an open exchange for the MATLAB community.

The main drawback of using MATLAB, of course, is its significant cost. Cheaper licenses are available for students, but the prices can still be substantial. Add-on libraries are available for ML and deep learning. Some universities, fortunately, offer free access to MATLAB to their students and faculty members based on an annually renewable contract.

If one is already familiar with MATLAB and its syntax, its adoption for ML research might be a fair choice. However, if one is new to ML and has no prior experience with programming languages, the author personally recommends the use of R or Python since they are free and more widely supported by the industry (such as Google).

Traditional compiled languages, such as Java and C++, can also be used, but the learning curves of these languages tend to be steeper. Unless one is a strong developer to begin with, these languages might not be the best choice.

To be fair, the author must state that all Turing-complete languages (i.e., most modern programming languages) are capable of performing ML research. The author's recommendation is primarily based on the ease of learning and the popularity. The second aspect is important because of the greater community support, the larger variety of third-party packages, and the richer documentation available.

Hardware and operating system

For ML research projects based on small- to medium-sized data (which include most traditional medical datasets), the hardware is not of primary concern. An entry-level standard CPU is adequate for most purposes, and an Intel i5 processor is more than sufficient. Any operating system (OS) (e.g., Windows, MacOS, or Linux) may be used. However, if one intends to conduct ML research based on big data or requires deep learning for image, sound, or video classification, a workstation PC that runs on GPU processors is recommended. The OS of preference is Linux (and more specifically, Ubuntu). The cost of acquiring and maintaining a server can be substantial, however. Unless the researcher is fully determined to conduct deep learning-based research, the author recommends the use of Cloud solutions instead. A good starting point is Google Colab (https://colab.research.google.com). This allows the use of Python based on Jupyter notebooks and provides the users with GPU (and TPU) support at no cost.

Examples of ML-assisted risk prediction

Dynamic prediction of postoperative nausea and vomiting

The Apfel simplified score for predicting postoperative nausea and vomiting (PONV) is currently a standard risk stratification system [14]. It consists of four factors, namely, gender, smoking status, history of motion sickness or PONV, and use of postoperative opioids. Each risk factor contributes a score of +1. Apfel scores of 1, 2, 3, and 4 are believed to be associated with 20%, 40%, 60%, and 80% risk of PONV, respectively. This system is limited in that patients given postoperative opioids, now a common clinical practice, already have one of the risk factors and none of the remaining three factors is modifiable. The Apfel system does not discriminate patients undergoing different surgery types and given different doses of opioids. Moreover, it does not consider the decay of PONV risk with time. A recent publication

by the author and co-investigators identified time-varying risk factors of PONV in patients undergoing general anesthesia and given intravenous patient-controlled analgesia [15]. The dataset consisted of more than 20,000 patients with greater than 40 different features collected from EHRs in Severance hospital, Seoul, Korea. ML techniques were applied to rank the most significant factors. Stepwise feature selection, using the highest-ranking features, was embedded into the workflow to develop a predictive model of PONV at different postoperative time intervals. A web application was created to assist in making real-time predictions of PONV risk under actual clinical settings (https://dongy.shinyapps.io/ponv_pred). The predictions were in concordance with the Apfel scores and went a step further to generate actual probabilities of PONV given a multitude of patient covariates, surgery type, and time elapsed after surgery. This work is a good demonstration of how ML can be applied to improve traditional scoring schemes.

Prediction of postoperative pain

Tighe et al. [16] reported a study whereby different ML models were built on data collected from EHRs to predict patients likely to experience severe postoperative pain after anterior cruciate ligament reconstruction. The results showed that ML models outperformed traditional logistic regression models in predictive severe postoperative pain requiring peripheral nerve block. The same group subsequently carried out a similar research where ML classifiers were trained to predict which patients would require a preoperative acute pain service consultation [17]. Training of various classifiers followed by an ensemble of a group of high-performing classifiers did not yield improved performance. Rather, the results showed that dimensional reduction improved computational efficiency while preserving predictive performance. The group recently argued that ML algorithms, when combined with complex and heterogeneous data from electronic medical record systems, can forecast acute postoperative pain outcomes with accuracies similar to methods that rely only on variables specifically collected for pain outcome prediction [18].

Prediction of postoperative kidney injury

Lee et al. [19] used ML techniques to predict acute kidney injury after cardiac surgery based on data collected from EHRs and developed an internet-based risk estimator. Compared with logistic regression analysis, decision tree, random forest, and support vector machine showed similar performances with regards to the AUC. The gradient boosting technique showed the best performance with the highest AUC. The same group published a similar article based on patients undergoing liver transplantation [20].

Empirical evidence from ML researchers and Kaggle competition winners suggest that ensemble methods often outcompete other algorithms when dealing with data in tabular format. Deep learning is generally unbeatable when it comes to classifying images, sounds, videos, and complex time series data.

Summary

While we have looked at a few instances, examples demonstrating the successful application of ML techniques in anesthesiology are too numerous to be listed in a single review. Most ML applications, despite being highly variable in the topics that they deal with, generally follow a common workflow. Becoming familiar with such a workflow is probably the first thing a newcomer to the field would want to achieve.

For classification tasks, the reference model is usually the logistic regression model, which is one of the most widely used statistical models in traditional medical research. The easy interpretability of the estimated odds ratios is perhaps one of the reasons for its high popularity. The researcher then tests other candidate ML models and compares the predictive performance (often based on the ROC-AUC) with the reference model. Most ML algorithms outperform the logistic regression model, and researchers use this fact to support the use of their own ML-based model. While this is the general workflow seen in most ML research articles, one must be careful not to put too much emphasis on the predictive performance alone. Sometimes, being able to interpret the results or explicitly identify the risk factors is important. If the gain in predictive performance is not dramatic, the logistic regression analysis could still be favored.

Conclusion

The application of ML to anesthesiology holds great promise for the future. The main advantage of ML is its ability to deal with many features with complex interactions, in addition to its specific focus on maximizing predictive performance.

However, the emphasis on data-driven prediction can sometimes neglect mechanistic understanding. This is particularly true when it comes to black box ML algorithms, such as deep learning. For example, a neural network trained to recognize the fingerprint of a private user does its job well, but it is often difficult to decipher how it does it. AlphaGo successfully beat the world Go champion but even its creator did not know how each move was made. It is somewhat unfortunate that as ML gains increasing

https://doi.org/10.4097/kja.20124

popularity, researchers are paying less attention to the reasoning behind the predictions. We hope that a proper balance between the two will be restored as ML researchers become more mature.

This article mainly focused on supervised ML as applied to EHR data. The basic concepts of ML were introduced, and several examples of the successful applications of ML to anesthesiology were shown. The author hopes that this article serves as a rudimentary roadmap for anesthesiology researchers who are now beginning to apply ML to their fields.

Conflicts of Interest

No potential conflict of interest relevant to this article was reported.

References

1. Jordan MI, Mitchell TM. Machine learning: trends, perspectives, and prospects. Science 2015; 349: 255-60.
2. Rajkomar A, Dean J, Kohane I. Machine learning in medicine. N Engl J Med 2019; 380: 1347-58.
3. Chen JH, Asch SM. Machine learning and prediction in medicine - beyond the peak of inflated expectations. N Engl J Med 2017; 376: 2507-9.
4. Child CG, Turcotte JG. Surgery and portal hypertension. Major Probl Clin Surg 1964; 1: 1-85.
5. Ho TK. The random subspace method for constructing decision forests. IEEE T Pattern Anal 1998; 20: 832-44.
6. Gomez D, Rojas A. An empirical overview of the no free lunch theorem and its effect on real-world machine learning classification. Neural Comput 2016; 28: 216-28.
7. Freund Y. An adaptive version of the boost by majority algorithm. Mach Learn 2001; 43: 293-318.
8. Schapire RE, Freund Y. Boosting: foundations and algorithms. Cambridge, MA: MIT Press. 2012, p 526.
9. Sun S, Cao Z, Zhu H, Zhao J. A survey of optimization methods from a machine learning perspective. IEEE Trans Cybern 2020; 50: 3668-81.
10. Jolliffe IT, Cadima J. Principal component analysis: a review and recent developments. Philos Trans A Math Phys Eng Sci 2016; 374: 20150202.
11. Demir-Kavuk O, Kamada M, Akutsu T, Knapp EW. Prediction using step-wise L1, L2 regularization and feature selection for small data sets with large number of features. BMC Bioinformatics 2011; 12: 412.
12. Remeseiro B, Bolon-Canedo V. A review of feature selection methods in medical applications. Comput Biol Med 2019; 112: 103375.
13. Lobo JM, Jimenez-Valverde A, Real R. AUC: a misleading measure of the performance of predictive distribution models. Global Ecol Biogeogr 2008; 17: 145-51.
14. Weilbach C, Rahe-meyer N, Raymondos K, Weissig A, Scheinichen D, Piepenbrock S. Postoperative nausea and vomiting (PONV): usefulness of the Apfel-score for identification of high risk patients for PONV. Acta Anaesthesiol Belg 2006; 57: 361-3.
15. Chae D, Kim SY, Song Y, Baek W, Shin H, Park K, et al. Dynamic predictive model for postoperative nausea and vomiting for intravenous fentanyl patient-controlled analgesia. Anaesthesia 2020; 75: 218-26.
16. Tighe P, Laduzenski S, Edwards D, Ellis N, Boezaart AP, Aygtug H. Use of machine learning theory to predict the need for femoral nerve block following ACL repair. Pain Med 2011; 12: 1566-75.
17. Tighe PJ, Lucas SD, Edwards DA, Boezaart AP, Aytug H, Bihorac A. Use of machine-learning classifiers to predict requests for preoperative acute pain service consultation. Pain Med 2012; 13: 1347-57.
18. Tighe PJ, Harle CA, Hurley RW, Aytug H, Boezaart AP, Fillingim RB. Teaching a machine to feel postoperative pain: combining high-dimensional clinical data with machine learning algorithms to forecast acute postoperative pain. Pain Med 2015; 16: 1386-401.
19. Lee HC, Yoon HK, Nam K, Cho YJ, Kim TK, Kim WH, et al. Derivation and validation of machine learning approaches to predict acute kidney injury after cardiac surgery. J Clin Med 2018; 7: E322.
20. Lee HC, Yoon SB, Yang SM, Kim WH, Ryu HG, Jung CW, et al. Prediction of acute kidney injury after liver transplantation: machine learning approaches vs. logistic regression model. J Clin Med 2018; 7: E428.