Machine Learning for Medical Imaging[1]

Bradley J. Erickson, MD, PhD
Panagiotis Korfiatis, PhD
Zeynettin Akkus, PhD
Timothy L. Kline, PhD

Abbreviations: CNN = convolutional neural network, ROI = region of interest

RadioGraphics 2017; 37:505–515

Published online 10.1148/rg.2017160130

Content Codes: ED IN RS

[1]From the Department of Radiology, Mayo Clinic, 200 First St SW, Rochester, MN 55905.
Presented as an education exhibit at the 2016 RSNA Annual Meeting. Received May 5, 2016; revision requested July 27 and received October 28; accepted November 4. For this journal-based SA-CME activity, the authors, editor, and reviewers have disclosed no relevant relationships. **Address correspondence to** B.J.E. (e-mail: *bje@mayo.edu*).

Supported by the National Cancer Institute (CA160045, DK90728).

T.L.K. supported by the PKD Foundation (206g16a).

SA-CME LEARNING OBJECTIVES

After completing this journal-based SA-CME activity, participants will be able to:

- List the basic types of machine learning algorithms and examples of each type.
- Discuss the typical problems encountered with machine learning approaches.
- Compute image features and choose methods to select the best features.

See www.rsna.org/education/search/RG.

Machine learning is a technique for recognizing patterns that can be applied to medical images. Although it is a powerful tool that can help in rendering medical diagnoses, it can be misapplied. Machine learning typically begins with the machine learning algorithm system computing the image features that are believed to be of importance in making the prediction or diagnosis of interest. The machine learning algorithm system then identifies the best combination of these image features for classifying the image or computing some metric for the given image region. There are several methods that can be used, each with different strengths and weaknesses. There are open-source versions of most of these machine learning methods that make them easy to try and apply to images. Several metrics for measuring the performance of an algorithm exist; however, one must be aware of the possible associated pitfalls that can result in misleading metrics. More recently, deep learning has started to be used; this method has the benefit that it does not require image feature identification and calculation as a first step; rather, features are identified as part of the learning process. Machine learning has been used in medical imaging and will have a greater influence in the future. Those working in medical imaging must be aware of how machine learning works.

©RSNA, 2017 • *radiographics.rsna.org*

Introduction

Machine learning is an exciting field of research in computer science and engineering. It is considered a branch of artificial intelligence because it enables the extraction of meaningful patterns from examples, which is a component of human intelligence. The appeal of having a computer that performs repetitive and well-defined tasks is clear: computers will perform a given task consistently and tirelessly; however, this is less true for humans. More recently, machines have demonstrated the capability to learn and even master tasks that were thought to be too complex for machines, showing that machine learning algorithms are potentially useful components of computer-aided diagnosis and decision support systems. Even more exciting is the finding that in some cases, computers seem to be able to "see" patterns that are beyond human perception. This discovery has led to substantial and increased interest in the field of machine learning—specifically, how it might be applied to medical images.

Because commercial products are proprietary, it is hard to determine how many U.S. Food and Drug Administration–cleared products use machine-learning algorithms, but market analysis results indicate that this is an important growth area (1). Computer-aided detection and diagnosis performed by using machine learning algorithms can help physicians interpret medical imaging findings and reduce interpretation times (2). These algorithms have been used for several challenging tasks, such as pulmonary embolism

TEACHING POINTS

- Although cross validation is a good method for estimating accuracy, an important limitation is that each set of training and testing iterations results in a different model, so there is no single model that can be used at the end.
- Decision trees offer the substantial advantage that they produce human-readable rules regarding how to classify a given example.
- The naive Bayes algorithm is different from most machine learning algorithms in that one calculation is used to define the relationship between an input feature set and the output. As such, this method does not involve the same iterative training process that most other machine learning methods involve.
- An important benefit of CNN deep learning algorithms, as compared with traditional machine learning methods, is that there is no need to compute features as a first step.
- Today's machine learning approaches are extremely robust to real-world conditions, and the systems actually benefit from the forced dropout of some data in the learning process.

segmentation with computed tomographic (CT) angiography (3,4), polyp detection with virtual colonoscopy or CT in the setting of colon cancer (5,6), breast cancer detection and diagnosis with mammography (7), brain tumor segmentation with magnetic resonance (MR) imaging (8), and detection of the cognitive state of the brain with functional MR imaging to diagnose neurologic disease (eg, Alzheimer disease) (9–11).

What Is Machine Learning?

Although all readers of this article probably have great familiarity with medical images, many may not know what machine learning means and/or how it can be used in medical image analysis and interpretation tasks (12–14). The following is one broadly accepted definition of machine learning: If a machine learning algorithm is applied to a set of data (in our example, tumor images) and to some knowledge about these data (in our example, benign or malignant tumors), then the algorithm system can learn from the training data and apply what it has learned to make a prediction (in our example, whether a different image is depicting benign or malignant tumor tissue) (Fig 1). If the algorithm system optimizes its parameters such that its performance improves—that is, more test cases are diagnosed correctly—then it is considered to be learning that task.

Machine learning is now being applied in many areas outside of medicine, having a central role in such tasks as speech recognition and translation between languages, autonomous navigation of vehicles, and product recommendations. Some of these tasks were not feasible previously; recent advances in machine learning have made them possible.

In the past, machine learning required structured input, and some techniques would not enable successful learning if any single point of data was missing. Newer algorithms can gracefully accommodate omissions in data, and in some cases, the system can purposefully create omissions in data during the learning phase to make the algorithm more robust. The new algorithms, combined with substantial increases in computational performance and data, have led to a renewed interest in machine learning.

Definitions

There are several terms commonly used in the machine learning community that may not be familiar to radiologists. The following list of key terms may help in understanding how machine learning works.

Classification: The assigning of a class or label to a group of pixels, such as those labeled as tumor with use of a segmentation algorithm. For instance, if segmentation has been used to mark some part of an image as "abnormal brain," the classifier might then try to determine whether the marked part represents benign or malignant tissue.

Model: The set of weights or decision points learned by a machine learning system. Once learned, the model can be assigned to an unknown example to predict which class that example belongs to.

Algorithm: The series of steps taken to create the model that will be used to most accurately predict classes from the features of the training examples.

Labeled data: The set of examples (eg, images), each with the correct "answer." For some tasks, this answer might be the correct boundary of a tumor, and in other cases, it might be whether cancer is present or the type of cancer the lesion represents.

Training: The phase during which the machine learning algorithm system is given labeled example data with the answers (ie, labels)—for example, the tumor type or correct boundary of a lesion. The set of weights or decision points for the model is updated until no substantial improvement in performance is achieved.

Validation set: The set of examples used during training. This is also referred to as the training set.

Testing: In some cases, a third set of examples is used for "real-world" testing. Because the algorithm system iterates to improve performance with the validation set, it may learn unique features of the training set. Good performance with an "unseen" test set can increase confidence that the algorithm will yield correct answers in the real world. Note that different groups sometimes use validation for testing and vice versa. This tends to reflect the engineering versus statistical

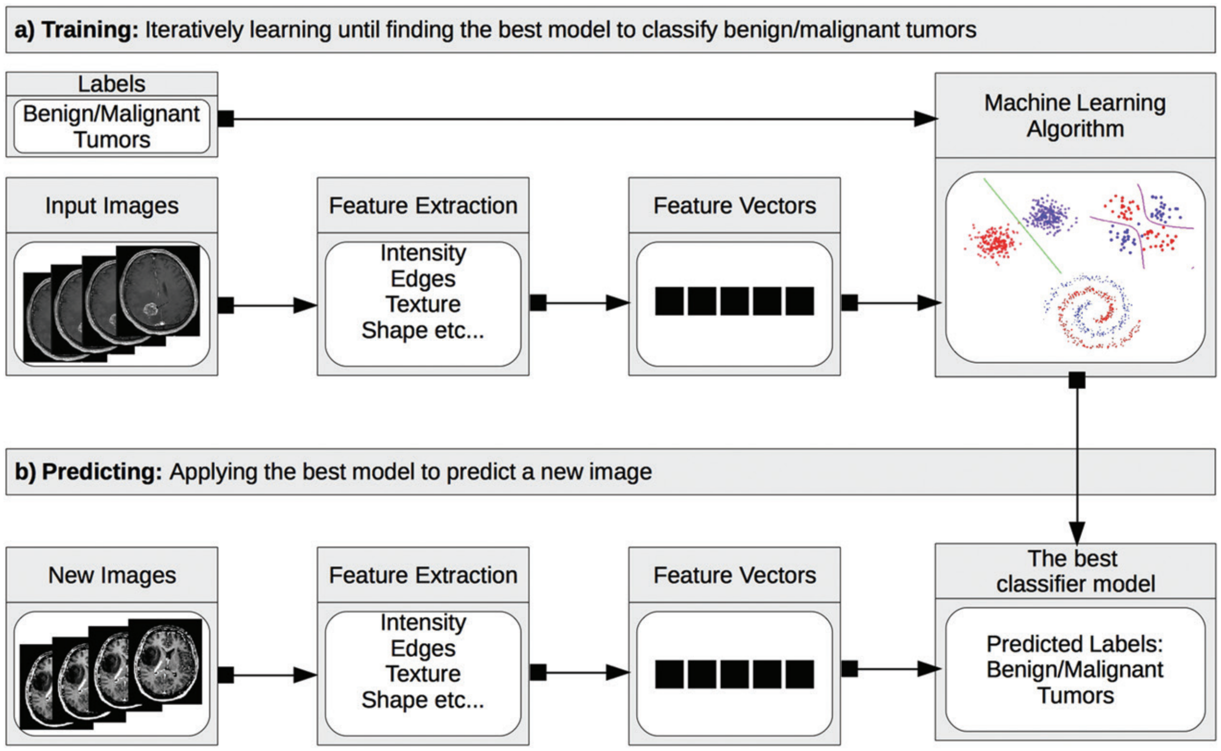
a) Training: Iteratively learning until finding the best model to classify benign/malignant tumors
Labels
Benign/Malignant Tumors
Machine Learning Algorithm
Input Images
Feature Extraction
Intensity Edges Texture Shape etc...
Feature Vectors
b) Predicting: Applying the best model to predict a new image
New Images
Feature Extraction
Intensity Edges Texture Shape etc...
Feature Vectors
The best classifier model
Predicted Labels: Benign/Malignant Tumors

Figure 1. Machine learning model development and application model for medical image classification tasks. For training, the machine learning algorithm system uses a set of input images to identify the image properties that, when used, will result in the correct classification of the image—that is, depicting benign or malignant tumor—as compared with the supplied labels for these input images. **(b)** For predicting, once the system has learned how to classify images, the learned model is applied to new images to assist radiologists in identifying the tumor type.

background. Therefore, it is important to clarify how these terms are used.

Node: A part of a neural network that involves two or more inputs and an activation function. The activation function typically sums the inputs and then uses some type of function and threshold to produce an output.

Layer: A collection of nodes that computes outputs (the next layer unless this is the output layer) from one or more inputs (the previous layer unless this is the input layer).

Weights: Each input feature is multiplied by some value, or weight; this is referred to as weighting the input feature. During training, the weights are updated until the best model is found. Machine learning algorithms can be classified on the basis of training styles: supervised, unsupervised, and reinforcement learning (15). To explain these training styles, consider the task of separating the regions on a brain image into tumor (malignant or benign) versus normal (nondiseased) tissue.

In our example, supervised learning involves gaining experience by using images of brain tumor examples that contain important information—specifically, “benign” and “malignant” labels—and applying the gained expertise to predict benign and malignant neoplasia on unseen new brain tumor images (test data). In this example case, the algorithm system would be given several brain tumor images on which the tumors were labeled as benign or malignant. Later, the system would be tested by having it try to assign benign and malignant labels to findings on the new images, which would be the test dataset. Examples of supervised learning algorithms include support vector machine (16), decision tree (17), linear regression (18), logistic regression (19), naive Bayes (19,20), *k*-nearest neighbor (21), random forest (22), AdaBoost, and neural network methods (23).

With unsupervised learning, data (eg, brain tumor images) are processed with a goal of separating the images into groups—for example, those depicting benign tumors and those depicting malignant tumors. The key difference is that this is done without the algorithm system being provided with information regarding what the groups are. The algorithm system determines how many groups there are and how to separate them. Examples of unsupervised learning algorithm systems include K-means (24), mean shift (24,25), affinity propagation (26), hierarchical clustering (26,27), DBSCAN (density-based spatial clustering of applications with noise) (28), Gaussian mixture modeling (28,29), Markov random fields (30), ISODATA (iterative self-organizing data) (31), and fuzzy C-means systems (32).

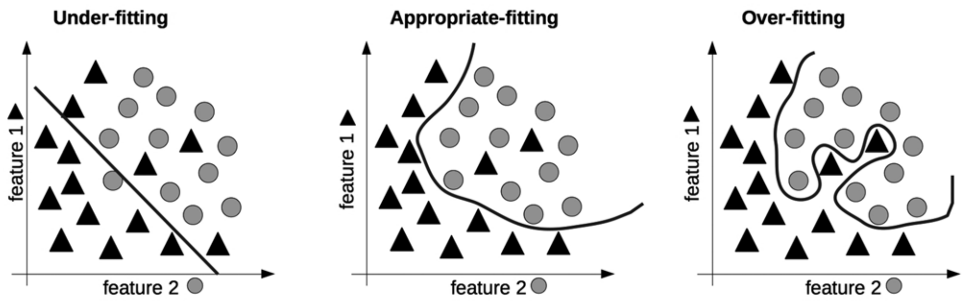

Figure 2. Diagrams illustrate under- and overfitting. Underfitting occurs when the fit is too simple to explain the variance in the data and does not capture the pattern. An appropriate fit captures the pattern but is not too inflexible or flexible to fit data. Overfitting occurs when the fit is too good to be true and there is possibly fitting to the noise in the data. The axes are generically labeled *feature 1* and *feature 2* to reflect the first two elements of the feature vector.

Like supervised learning, reinforcement learning begins with a classifier that was built by using labeled data. However, the system is then given unlabeled data, and it tries to further improve the classification by better characterizing these data—similar to how it behaves with unsupervised learning. Examples of reinforcement learning algorithm systems include Maja (33) and Teaching-Box (34) systems. In this article, we focus on supervised learning, since it is the most common training style applied to medical images.

Segmentation: The splitting of the image into parts. For instance, with tumor segmentation, this is the process of defining where the tumor starts and stops. However, this does not necessarily include deciding that what is included is tumor. The goal in this step is to determine where something starts and stops. This technique is usually used with a classifier that determines that a segment of an image is depicting enhancing tumor and another segment is depicting nonenhancing tumor.

Overfitting: When a classifier that is too specific to the training set is not useful because it is familiar with only those examples, this is known as overfitting (Fig 2). In general, the training set needs to contain many more examples above the number of coefficients or variables used by the machine learning algorithm.

Features: The numeric values that represent the example. In the case of medical images, features can be the actual pixel values, edge strengths, variation in pixel values in a region, or other values. One can also use nonimage features such as the age of the patient and whether a laboratory test has positive or negative results. When all of these features are combined for an example, this is referred to as a feature vector, or input vector.

Feature Computation and Selection

Feature Computation.—The first step in machine learning is to extract the features that contain the information that is used to make decisions. A review of the ways in which features are computed is beyond the scope of this article; thus, we refer readers to the many books that have been written about feature extraction (33,34). Humans learn important features visually, such as during radiology residencies; however, it can be challenging to compute or represent a feature—to assign a numeric value to ground-glass texture, for example. Image features should be robust against variations in noise, intensity, and rotation angles, as these are some of the most common variations observed when working with medical imaging data.

Feature Selection.—Although it is possible to compute many features from an image, having too many features can lead to overfitting rather than learning the true basis of a decision (35). The process of selecting the subset of features that should be used to make the best predictions is known as feature selection (36,37). One feature selection technique is to look for correlations between features: having large numbers of correlated features probably means that some features and the number of features can be reduced without information being lost. However, in some cases, a more complex relationship exists and evaluating a feature in isolation is dangerous. Suppose, for instance, that you are given a list of weights with binary classifications of whether each weight indicates or does not indicate obesity. One could make some guesses, but adding heights would improve the accuracy: a rather high weight value in conjunction with a low height value is more likely to reflect obesity than is a high weight value in conjunction with a high height value.

Training and Testing: The Learning Process

Supervised machine learning is so named because examples of each type of thing to be learned are required. An important question to ask is “How many examples of each class of the thing do I need to learn it well?" It is easy to see that having too few examples will prevent a computer—or a person, for that matter—from recognizing those features of an object that allow one to distinguish between the different classes of that object (35). The exact number of examples in each class that is required depends heavily on how distinctive the classes are. For instance, if you wish to create an algorithm to separate cars and trucks and you provide a learning algorithm system with an image of a red car labeled "class A" and an image of a black truck labeled "class B," then using an image of a red truck to test the learning algorithm system may or may not be successful. If you provide examples of "class A" that include red, green, and black trucks, as well as examples of "class B" that include red, yellow, green, and black cars, then the algorithm system is more likely to separate trucks from cars because the shape features override the color features. Of course, if the person who computed the features used in training did not provide color as an input, then color would not be mistaken as a feature for separating trucks and cars.

One popular way to estimate the accuracy of a machine learning system when there is a limited dataset is to use the cross-validation technique (38,39). With cross validation, one first selects a subset of examples for training and designates the remaining examples to be used for testing. Training proceeds, and the learned state is tested. This process is then repeated, but with a different set of training and testing examples selected from the full set of training examples. In the extreme case, one may remove just one example for testing and use all of the others for each round of training; this technique is referred to as leave-one-out cross validation (40). Although cross validation is a good method for estimating accuracy, an important limitation is that each set of training and testing iterations results in a different model, so there is no single model that can be used at the end.

Example of Machine Learning with Use of Cross Validation.—Having provided the preceding background information, we now describe a concrete though simple example of machine learning. Imagine that we wish to separate brain tumor from normal brain tissue and that we have CT images that were obtained without and those that were obtained with contrast material. We have 10 subjects, and 10 regions of interest (ROIs) in normal white matter and 10 ROIs in tumor tissue have been drawn on the CT images obtained in each of these subjects. This means that we have 100 input vectors from white matter and 100 input vectors from tumor, and we will sequence the vectors such that the first value is the mean CT attenuation of the ROI on the non–contrast material–enhanced image, and the second value is the mean attenuation of the ROI on the contrast material–enhanced image.

With CT of brain tumors, the attenuation values on the nonenhanced images will be similar, though perhaps lower on average for normal brain tissue than for tumors. Enhancing tumor will have higher attenuation on the contrast-enhanced images. However, other tissues in the brain, such as vessels, also will enhance. It is also possible that parts of the tumor will not enhance. In addition, although much of the tumor may be darker on the nonenhanced images, areas of hemorrhage or calcification can make the lesion brighter. On the basis of the latter observation, we will also calculate the variance in attenuation and use this value as the third feature in the vector. To help eliminate vessels, we will calculate the tubularity of the voxels with an attenuation higher than 300 HU and store this value as the fourth feature. One can imagine many more values, such as location of the tumor in the head, that might be useful for some tasks, but we will stick with these four features.

We will take 70 of the normal brain tissue ROIs and 70 tumor ROIs and send them to the machine learning algorithm system. The algorithm system will start with random weights for each of the four features and in this simple model add the four products. If the sum is greater than 0, the algorithm system will designate the ROI as tumor; otherwise, the ROI will be designated as normal brain tissue. The algorithm system will do this for all 140 examples. It will then try to adjust one of the weights to see whether this reduces the number of wrong interpretations. The system will keep adjusting weights until no more improvement in accuracy is seen. It will then take the remaining 30 examples of each normal brain tissue ROI and each tumor ROI and evaluate the prediction accuracy; in this example case, let us say that it will designate 50 of these 60 ROIs correctly.

We will now take a different group of 70 tumor ROIs and 70 normal tissue ROIs and train in a new network to see how accurate the algorithm system is in interpreting the remaining 30 tumor cases and 30 normal cases. We will repeat this process several times to derive a mean accuracy for this algorithm and dataset. This would be an example of 70/30 cross validation.

Types of Machine Learning Algorithms

There are many algorithms for selecting the best weights for features. These algorithms are based on different methods for adjusting the feature weights and assumptions about the data. Some of the common techniques—specifically, those

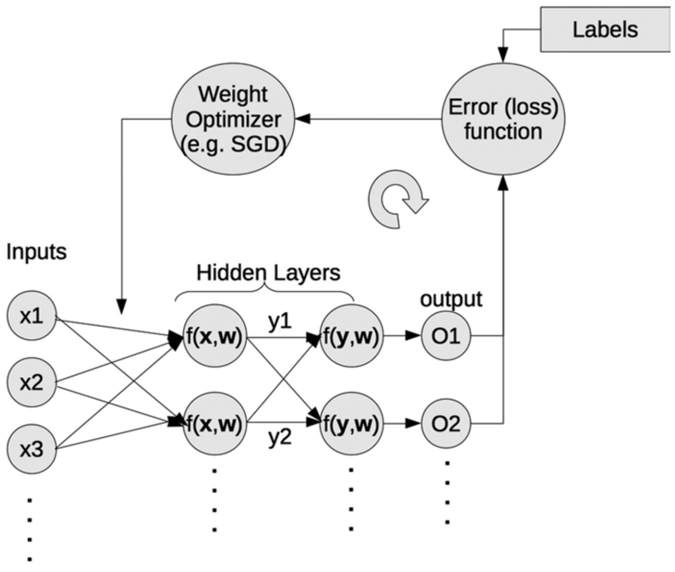

Figure 3. Example of a neural network. In this case, the input values *(×1, ×2, ×3)* are multiplied by a weight *(w)* and passed to the next layer of nodes. Although we show just a single weight, each such connection weight has a different numeric value, and it is these values that are updated as part of the learning process. Each node has an activation function *(f)* that computes its output *(y)* by using *x* and *w* as inputs. The last layer is the output layer. Those outputs are compared with the expected values (the training sample labels), and an error is calculated. The weight optimizer determines how to adjust the various weights in the network in order to achieve a lower error in the next iteration. Stochastic gradient descent *(SGD)* is one common way of updating the weights of the network. The network is considered to have completed learning when there is no substantial improvement in the error over prior iterations.

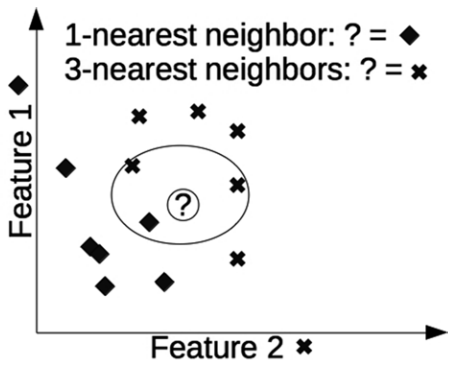

Figure 4. Example of the *k*-nearest neighbors algorithm. The unknown object *(?)* would be assigned to the ◆ class on the basis of the nearest neighbor (*k* = 1), but it would be assigned to the × class if *k* were equal to 3, because two of the three closest neighbors are × class objects. Values plotted on the x and y axes are those for the two-element feature vector describing the example objects.

involving neural networks, *k*-nearest neighbors, support vector machines, decision trees, the naive Bayes algorithm, and deep learning—are described in the following sections.

Neural Networks

Learning with neural networks is the archetypal machine learning method. The following three functions are parts of the learning schema for this method (Fig 3): *(a)* the error function measures how good or bad an output is for a given set of inputs, *(b)* the search function defines the direction and magnitude of change required to reduce the error function, and *(c)* the update function defines how the weights of the network are updated on the basis of the search function values.

The example provided in Figure 3 would be a neural network with several input nodes (referred to as ×1 to ×*n*), two hidden layers, and an output layer with several output nodes. The output nodes are summed and compared with the desired output by the error (loss) function, which then uses the weight optimizer to update the weights in the neural network. As described earlier, during the training phase, examples are presented to the neural network system, the error for each example is computed, and the total error is computed. On the basis of the error, the search function determines the overall direction to change, and the update function then uses this change metric to adjust the weights. This is an iterative process, and one typically continues to adjust the weights until there is little improvement in the error. Real-world examples typically have one or more hidden layers and more complex functions at each node.

k-Nearest Neighbors

With *k*-nearest neighbors (41), one classifies an input vector—that is, a collection of features for one unknown example object—by assigning the object to the most similar class or classes (Fig 4). The number of neighbors, or known objects that are closest to the example object, that "vote" on the classes that the example object may belong to is *k*. If *k* is equal to 1, then the unknown object is simply assigned to the class of that single nearest neighbor. The similarity function, which determines how close one example object is to another, can be the Euclidean distance between the values of the input vector versus the values of the vector for the other examples. However, it is critical that the normalization of the values in the feature vectors be performed correctly.

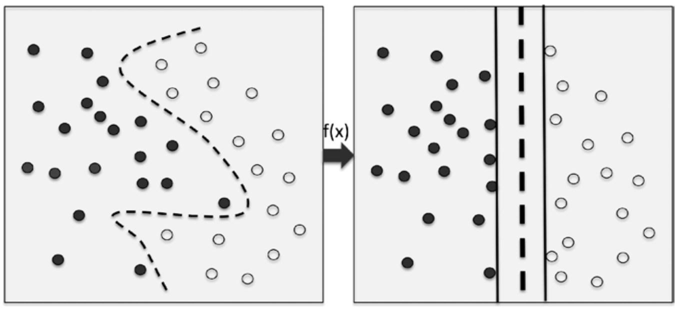

Figure 5. Example shows two classes (●, ○) that cannot be separated by using a linear function (left diagram). However, by applying a nonlinear function *f(x),* one can map the classes to a space where a plane can separate them (right diagram). This example is two dimensional, but support vector machines can have any dimensionality required. These machines generally are "well behaved," meaning that for new examples that are similar, the classifier usually yields reasonable results. When the machine learning algorithm is successful, the two classes will be perfectly separated by the plane. In the real world, perfect separation is not possible, but the optimal plane that minimizes misclassifications can be found.

Support Vector Machines

Support vector machines are so named because they transform input data in a way that produces the widest plane, or support vector, of separation between the two classes. Support vector machines allow flexible selection of the degree to which one wishes to have a wide plane of separation versus the number of points that are wrong owing to the wide plane. These learning machines were invented some time ago (42), and the reason for their recent greater popularity is the addition of basis functions that can map points to other dimensions by using nonlinear relationships (43,44) and thus classify examples that are not linearly separable. This capability gives support vector machine algorithms a big advantage over many other machine learning methods. A simple example of how a nonlinear function can be used to map data from an original space (the way the feature was collected—eg, the CT attenuation) to a hyperspace (the new way the feature is represented—eg, the cosine of the CT attenuation) where a hyperplane (a plane that exists in that hyperspace, the idea being to have the plane positioned to optimally separate the classes) can separate the classes is illustrated in Figure 5.

Decision Trees

All of the machine learning methods described up to this point have one important disadvantage: the values used in the weights and the activation functions usually cannot be extracted to gain some form of information that can be interpreted by humans. Decision trees offer the substantial advantage that they produce human-readable rules regarding how to classify a given example (45). Decision trees are familiar to most people and typically take the form of yes or no questions—for example, whether a numeric value is higher than a certain value.

The aspect of decision trees that applies to machine learning is the rapid search for the many possible combinations of decision points to find the points that, when used, will result in the simplest tree with the most accurate results. When the algorithm is run, one sets the maximal depth (ie, maximal number of decision points) and the maximal breadth that is to be searched and establishes how important it is to have correct results versus more decision points.

In some cases, one can improve accuracy by using an ensemble method whereby more than one decision tree is constructed. Two commonly used ensemble methods are bagging and random forest techniques. By boosting with aggregation, or bagging, one builds multiple decision trees by repeatedly resampling the training data by means of replacement, and voting on the trees to reach a consensus prediction (46). Although a random forest classifier uses a number of decision trees to improve the classification rate and is often high performing, it does not resample the data. In addition, with random forests, only a subset of the total number of features is randomly selected and the best split feature from the subset is used to split each node in a tree—unlike with bagging, whereby all features are considered for splitting a node.

Naive Bayes Algorithm

According to the Bayes theorem, one of the oldest machine learning methods (47), the probability of an event is a function of related events. The Bayes theorem formula is $P(y|x) = [P(y) \times P(x|y)]/P(x)$: the probability (P) of y given x equals the probability of y times the probability of x given y, divided by the probability of x.

In machine learning, where there are multiple input features, one must chain the probabilities of each feature together to compute the final probability of a class, given the array of input features that is provided.

The naive Bayes algorithm is different from most machine learning algorithms in that one calculation is used to define the relationship between an input feature set and the output. As such, this method does not involve the same iterative training process that most other machine learning methods involve. It does require training and testing data, so the issues related to training and testing data still apply.

This algorithm is referred to as the naive Bayes algorithm rather than simply the Bayes algorithm to emphasize the point that all features are assumed to be independent of each other. Because this is usually not the case in real life, using this approach can lead to misleading results. However, this method can be used to acquire useful estimates of performance, even when this assumption is violated (48). In addition, the use of this approach often leads to more robust results when there are fewer examples and when the examples do not include all possibilities.

These considerations also raise the important issue of pretest probabilities and accuracy: if the prevalence of a positive finding were 1%, then one could simply designate all cases as those of negative findings and achieve 99% accuracy. In many cases, 99% accuracy would be good, and this algorithm would also have 100% specificity; however, it would have 0% sensitivity. From this perspective, it is important to recognize that accuracy alone is not sufficient and prior probability is an important piece of information that will affect performance measures.

Deep Learning

Deep learning, also known as deep neural network learning, is a new and popular area of research that is yielding impressive results and growing fast. Early neural networks were typically only a few (<5) layers deep, largely because the computing power was not sufficient for more layers and owing to challenges in updating the weights properly. Deep learning refers to the use of neural networks with many layers—typically more than 20. This has been enabled by tools that leverage the massively parallel computing power of graphics processing units that were created for computer gaming, such as those built by NVidia Corporation (Santa Clara, Calif). Several types of deep learning networks have been devised for various purposes, such as automatic object detection (49) and segmentation (50) on images, automatic speech recognition (51), and genotypic and phenotypic detection and classification of diseases in bioinformatics. Some deep learning algorithm tools are deep neural networks, stacked auto encoders, deep Boltzmann machines, and convolutional neural networks (CNNs). We will focus on CNNs because these are most commonly applied to images (52,53).

CNNs are similar to regular neural networks. The difference is that CNNs assume that the inputs have a geometric relationship—like the rows and columns of images. The input layer of a CNN has neurons arranged to produce a convolution of a small image (ie, kernel) with the image. This kernel is then moved across the image, and its output at each location as it moves across the input image creates an output value. Although CNNs are so named because of the convolution kernels, there are other important layer types that they share with other deep neural networks. Kernels that detect important features (eg, edges and arcs) will have large outputs that contribute to the final object to be detected.

In deep networks, specialized layers are now used to help amplify the important features of convolutional layers. The layer typically found after a convolution layer is an activation layer. In the past, activation functions were designed to simulate the sigmoidal activation function of a neuron, but current activation layers often have a much simpler function. A common example is the rectified linear unit, or ReLU (54), which has an output of 0 for any negative value and an output equal to the input value for any positive value.

The pooling layer is another type of layer that is important to CNNs. A pooling layer will take the output of something like a convolution kernel and find the maximal value; this is the so-called max-pool function (55). By taking the maximal value of the convolution, the pooling layer is rewarding the convolution function that best extracts the important features of an image.

An important step in training deep networks is regularization, and one popular form of regularization is dropout (56). Regularization refers to rescaling the weights connecting a pair of layers to a more effective range. Somewhat counterintuitively, randomly setting the weights between nodes of layers to 0 has been shown to substantially improve performance because it reduces overfitting. Dropout regularization is typically implemented by having weights (often 50% or more between two layers) set to 0. The specific connections that are set to 0 at a given layer are random and vary with each round of learning. One can imagine that if random connection weights are set to 0 and a group of examples is tested, then those weights that are really important will affect performance, but

Open-Source Traditional and Deep Machine Learning Library Packages Compatible with Various Programming Languages

Programming Language	Traditional Machine Learning Libraries	Deep Neural Network Machine Learning Libraries
Python	Scikit-learn, PyBrain, Nilearn, Pattern, MILK, Mixtend	Pylearn2, Nolearn, Theano, Lasagne, Keras, Chainer, DeePy, TensorFlow
R	Caret, Boruta, GMMBoost, H2O, KlaR, rminer	Darch, DeepNet
C++	Shogun	Caffe, EBLearn, Intel Deep Learning Framework
Lua	SciLua	Torch
Octave MATLAB	...	DeepLearnToolbox
Java	Encog, Spark, Mahout, MALLET, Weka	Deeplearning4j
JavaScript	Clusterfck, LDA, Node-SVM, ml.jis	ConvNetJS

Sources.—References 63–85.

those weights that are not so important and perhaps reflective of a few specific examples will have a much smaller influence on performance. With enough iterations, only the really important connections will be kept.

There are many possible combinations of layers and layer sizes. At present, there is no formula to define the correct number and type of layer for a given problem. Selecting the best architecture for a given problem is still a trial-and-error process. It is interesting that some different neural network architectures have been successful in machine learning competitions such as the ImageNet Challenge (57). Some of these architectures are LeNet (58), GoogleNet (59), AlexNet (60), VGGNet (61), and ResNet (62).

An important benefit of CNN deep learning algorithms, as compared with traditional machine learning methods, is that there is no need to compute features as a first step. The CNN effectively finds the important features as a part of its search process. As a result, the bias of testing only those features that a human believes to be important is eliminated. The task of computing many features and then selecting those that seem to be the most important also is eliminated.

Open-Source Tools

A wide variety of open-source tools for developing and implementing machine learning are available. These tools are compatible with the majority of modern programming languages, including Python, C++, Octave MATLAB, R, and Lua. Furthermore, tools such as Apache Storm, Spark, and H2O libraries have been developed for machine learning tasks and large datasets. Most deep learning tool kits can now leverage graphics processing unit power to accelerate the computations of a deep network. Some of the most commonly used libraries for machine learning are summarized in the Table. Python libraries tend to be the most popular and can be used to implement the most recently available algorithms; however, there are many ways to access the algorithms implemented in one language from another language. In fact, many Python libraries are implemented in C++. Furthermore, some libraries are built on other libraries—for example, the Keras library runs on top of either Theano or TensorFlow (67).

We have set up a GitHub repository that provides simple examples of the machine learning libraries described herein. To access this repository with the sample code and example images, run the following program from a command prompt: *git clone git://github.com/slowvak/MachineLearningForMedicalImages.git*.

You must have the Git software installed on your computer. Then change directory ("cd") to the MachineLearningForMedicalImages directory and follow the instructions in the Readme.md file. If you do not have Git software on your computer, you can download the code as a zip file from the *github.com* website.

Conclusion

There has been tremendous progress in machine learning technology since this algorithm was first imagined 50 years ago. In the beginning, the models were simple and "brittle"—that is, they did not tolerate any deviations from the examples provided during training. Today's machine learning approaches are extremely robust to real-world conditions, and the systems actually benefit from the forced dropout of some data in the learning process. Owing to the rapid pace of technologic advancements, tasks previously thought to be limited to humans will be taken on by machine learning systems. Machine learning is already being applied in the practice of radiology, and these applications will probably grow at a rapid pace in the near future. The use of machine learning in radiology has important implications for the practice of medicine, and it is important that we engage this area of research to ensure that the best care is afforded to patients. Understanding the properties of machine learning tools is critical to ensuring that they are applied in the safest and most effective manner.

References

1. From $600 M to $6 billion, artificial intelligence systems poised for dramatic market expansion in healthcare. Frost & Sullivan website. http://ww2.frost.com/news/press-releases/600-m-6-billion-artificial-intelligence-systems-poised-dramatic-market-expansion-healthcare/. Accessed September 2, 2016.
2. Schoepf UJ, Costello P. CT angiography for diagnosis of pulmonary embolism: state of the art. Radiology 2004;230(2):329–337.
3. Schoepf UJ, Schneider AC, Das M, Wood SA, Cheema JI, Costello P. Pulmonary embolism: computer-aided detection at multidetector row spiral computed tomography. J Thorac Imaging 2007;22(4):319–323.
4. Dundar MM, Fung G, Krishnapuram B, Rao RB. Multiple-instance learning algorithms for computer-aided detection. IEEE Trans Biomed Eng 2008;55(3):1015–1021.
5. Summers RM. Improving the accuracy of CTC interpretation: computer-aided detection. Gastrointest Endosc Clin N Am 2010;20(2):245–257.
6. Yoshida H, Näppi J. CAD in CT colonography without and with oral contrast agents: progress and challenges. Comput Med Imaging Graph 2007;31(4-5):267–284.
7. Chan HP, Lo SC, Sahiner B, Lam KL, Helvie MA. Computer-aided detection of mammographic microcalcifications: pattern recognition with an artificial neural network. Med Phys 1995;22(10):1555–1567.
8. Bauer S, Wiest R, Nolte LP, Reyes M. A survey of MRI-based medical image analysis for brain tumor studies. Phys Med Biol 2013;58(13):R97–R129.
9. Mitchell TM, Shinkareva SV, Carlson A, et al. Predicting human brain activity associated with the meanings of nouns. Science 2008;320(5880):1191–1195.
10. Davatzikos C, Fan Y, Wu X, Shen D, Resnick SM. Detection of prodromal Alzheimer's disease via pattern classification of magnetic resonance imaging. Neurobiol Aging 2008;29(4):514–523.
11. Kim D, Burge J, Lane T, Pearlson GD, Kiehl KA, Calhoun VD. Hybrid ICA-Bayesian network approach reveals distinct effective connectivity differences in schizophrenia. Neuroimage 2008;42(4):1560–1568.
12. Suzuki K. Pixel-based machine learning in medical imaging. Int J Biomed Imaging 2012;2012:792079 .
13. Jalalian A, Mashohor SB, Mahmud HR, Saripan MI, Ramli AR, Karasfi B. Computer-aided detection/diagnosis of breast cancer in mammography and ultrasound: a review. Clin Imaging 2013;37(3):420–426.
14. Kononenko I. Machine learning for medical diagnosis: history, state of the art and perspective. Artif Intell Med 2001;23(1):89–109.
15. Flach P. Machine learning: the art and science of algorithms that make sense of data. Cambridge, England: Cambridge University Press, 2012.
16. Cristianini N, Shawe-Taylor J. An introduction to support vector machines and other kernel-based learning methods. Cambridge, England: Cambridge University Press, 2000.
17. Quinlan JR. Induction of decision trees. Mach Learn 1986;1(1):81–106.
18. Seber GAF, Lee AJ. Linear regression analysis. 2nd ed. New York, NY: Wiley, 2012.
19. Hosmer DW, Stanley L. Applied logistic regression. 2nd ed. New York, NY: Wiley, 2000.
20. Lowd D, Daniel L, Pedro D. Naive Bayes models for probability estimation. Proceedings of the 22nd International Conference on Machine Learning - ICML '05. New York, NY: Association for Computing Machinery, 2005.
21. Zhou CY, Chen YQ. Improving nearest neighbor classification with cam weighted distance. Pattern Recognit 2006;39(4):635–645.
22. Breiman L. Random forests. Mach Learn 2001;45(1):5–32.
23. Hornik K, Kurt H, Maxwell S, Halbert W. Multilayer feedforward networks are universal approximators. Neural Netw 1989;2(5):359–366.
24. Krishna K, Narasimha Murty M. Genetic K-means algorithm. IEEE Trans Syst Man Cybern B Cybern 1999;29(3):433–439.
25. Comaniciu D, Meer P. Mean shift: a robust approach toward feature space analysis. IEEE Trans Pattern Anal Mach Intell 2002;24(5):603–619.
26. Dueck D, Frey BJ. Non-metric affinity propagation for unsupervised image categorization. IEEE 11th International Conference on Computer Vision. New York, NY: Institute of Electrical and Electronics Engineers, 2007; 1–8.
27. Johnson SC. Hierarchical clustering schemes. Psychometrika 1967;32(3):241–254.
28. Birant D, Kut A. ST-DBSCAN: an algorithm for clustering spatial-temporal data. Data Knowl Eng 2007;60(1):208–221.
29. Roberts SJ, Husmeier D, Rezek I, Penny W. Bayesian approaches to Gaussian mixture modeling. IEEE Trans Pattern Anal Mach Intell 1998;20(11):1133–1142.
30. Chellappa R, Jain AK. Markov random fields: theory and application. Boston, Mass: Academic Press, 1993.
31. Dunn JC. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. J Cybern 1973;3(3):32–57.
32. Bezdek JC, Ehrlich R, Full W. FCM: the fuzzy C-means clustering algorithm. Comput Geosci 1984;10(2-3):191–203.
33. Nixon M, Nixon MS, Aguado AS. Feature extraction & image processing for computer vision. London, England: Academic Press, 2012.
34. Wang Y, Yuhang W. Spatial feature extraction algorithms (master's thesis). Hanover, NH: Dartmouth College, 2005.
35. Way TW, Sahiner B, Hadjiiski LM, Chan HP. Effect of finite sample size on feature selection and classification: a simulation study. Med Phys 2010;37(2):907–920.
36. Saeys Y, Inza I, Larrañaga P. A review of feature selection techniques in bioinformatics. Bioinformatics 2007;23(19):2507–2517.
37. Kuhn M, Johnson K. An introduction to feature selection. In: Applied predictive modeling. New York, NY: Springer, 2013; 487–519.
38. Kohavi R, Ron K, John GH. Automatic parameter selection by minimizing estimated error. Proceedings of the Twelfth International Conference on machine learning, Tahoe City, Calif. New York, NY: Elsevier, 1995; 304–312.
39. Arlot S, Celisse A. A survey of cross-validation procedures for model selection. Stat Surv 2010;4:40–79.
40. Zhang T, Tong Z. A leave-one-out cross validation bound for kernel methods with applications in learning. Lect Notes Comput Sci 2001;2111:427–443.
41. Cover T, Hart P. Nearest neighbor pattern classification. IEEE Trans Inf Theory 1967;13(1):21–27.
42. Vapnik V, Lerner A. Pattern recognition using generalized portrait method. Autom Remote Control 1963;24(6):774–780.
43. Buhmann MD. Radial basis functions with compact support. In: Radial basis functions: theory and implementations. Cambridge, England: Cambridge University Press, 147–162.
44. Prajapati GL, Patle A. On performing classification using SVM with radial basis and polynomial kernel functions: 2010 3rd International Conference on Emerging Trends in Engineering and Technology. New York, NY: Institute of Electrical and Electronics Engineers, 2010.
45. Rokach L, Maimon O. Data mining with decision trees: theory and applications. 2nd ed. Singapore: World Scientific Publishing Company, 2014.
46. Breiman L, Leo B. Bagging predictors. Mach Learn 1996;24(2):123–140.
47. Duda R, Hart P. Pattern classification and scene analysis. New York, NY: Wiley, 1973; 139–143.

48. Hand DJ, Keming Y. Idiot's Bayes: not so stupid after all? Int Stat Rev 2001;69(3):385–398.
49. Szegedy C, Toshev A, Erhan D. Deep neural networks for object detection. In: Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ, eds. Advances in neural information processing systems. Red Hook, NY: Curran Associates, 2013; 2553–2561.
50. Brosch T. Efficient deep learning of 3D structural brain MRIs for manifold learning and lesion segmentation with application to multiple sclerosis. The University of British Columbia Library website. https://open.library.ubc.ca/collections/ubctheses/24/items/1.0305854. Published 2016. Accessed October 2016.
51. Abadi M, Agarwal A, Barham P, et al. TensorFlow: large-scale machine learning on heterogeneous distributed systems. Cornell University Library website. http://arxiv.org/abs/1603.04467. Published 2016. Accessed October 2016.
52. Lee H, Hansung L, Yunsu C, Jeongnyeo K, Daihee P. Face image retrieval using sparse representation classifier with Gabor-LBP histogram. Lect Notes Comput Sci 2011;6513: 273–280.
53. Le Cun Y, Jackel LD, Boser B, et al. Handwritten digit recognition: applications of neural net chips and automatic learning. Neurocomputing 1990;68:303–318.
54. Dahl GE, Sainath TN, Hinton GE. Improving deep neural networks for LVCSR using rectified linear units and dropout. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. New York, NY: Institute of Electrical and Electronics Engineers, 2013.
55. Zhou YT, Chellappa R, Vaid A, Jenkins BK. Image restoration using a neural network. IEEE Trans Acoust 1988; 36(7):1141–1151.
56. Srivastava N, Hinton GR, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15(7): 1929–1958.
57. Russakovsky O, Olga R, Jia D, et al. ImageNet large scale visual recognition challenge. Int J Comput Vis 2015;115(3): 211–252.
58. Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE 1998;86(11): 2278–2324.
59. Szegedy C, Christian S, Wei L, et al. Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). New York, NY: Institute of Electrical and Electronics Engineers, 2015.
60. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ, eds. Advances in neural information processing systems. Red Hook, NY: Curran Associates, 2012; 1097–1105.
61. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. Cornell University Library website. http://arxiv.org/abs/1409.1556. Published 2014. Accessed October 2016.
62. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. Cornell University Library website. http://arxiv.org/abs/1512.03385. Published 2015. Accessed October 2016.
63. Scikit-learn: machine learning in Python. Scikit Learn website. http://scikit-learn.org/stable/. Accessed May 2, 2016.
64. Web mining module for Python, with tools for scraping, natural language processing, machine learning, network analysis and visualization. CLiPS Resources website. http://www.clips.ua.ac.be/pages/pattern. Accessed October 2016.
65. Coelho LP. Milk: machine learning toolkit. Luis Predo Coelho website. luispedro.org/software/milk/. Accessed May 2, 2016.
66. Pylearn2. Pylearn2 dev documentation website. http://deeplearning.net/software/pylearn2/. Accessed October 2016.
67. Keras: Deep learning library for Theano and TensorFlow. Keras Documentation website. http://keras.io/. Accessed October 2016.
68. DeePy: a highly extensible deep learning framework. DeePy website. https://github.com/zomux/deepy. Accessed October 2016.
69. The Caret package. Caret website. http://topepo.github.io/caret/index.html. Accessed May 2, 2016.
70. Boruta: wrapper algorithm for all-relevant feature selection. Boruta website. https://cran.r-project.org/web/packages/Boruta/index.html. Accessed May 2, 2016.
71. GMMBoost: likelihood-based boosting for generalized mixed models. GMMBoost website. https://cran.r-project.org/web/packages/GMMBoost/index.html. Accessed October 2016.
72. H2O: R interface for H2O. H2O website. https://cran.r-project.org/web/packages/h2o/index.html. Accessed October 2016.
73. Shogun: unified and efficient machine learning. Shogun website. http://www.shogun-toolbox.org/. Accessed October 2016.
74. EBLearn: open source C++ machine learning library. EBLearn website. http://eblearn.sourceforge.net/. Accessed October 2016.
75. Intel deep learning framework (IDLF). Intel Software Developer Zone website. https://01.org/intel-deep-learning-framework. Accessed October 2016.
76. Torch: a scientific computing framework for Luajit. Torch website. http://torch.ch/. Accessed October 2016.
77. Encog machine learning framework. Heaton Research website. http://www.heatonresearch.com/encog/. Accessed October 2016.
78. Apache Spark: lightning-fast cluster computing. Apache Spark website. http://spark.apache.org/. Accessed October 2016.
79. What is Apache Mahout? Mahout website. http://mahout.apache.org/. Accessed October 2016.
80. MALLET: machine learning for language toolkit. MALLET website. http://mallet.cs.umass.edu/. Accessed October 2016.
81. Deep learning for Java: open-source, distributed, deep learning library for the JVM. Deeplearning4j website. https://deeplearning4j.org/. Accessed October 2016.
82. Clusterfck. GitHub Harthur/clusterfck website. https://github.com/harthur/clusterfck. Accessed October 2016.
83. LDA topic modeling for node.js. GitHub Primaryobjects/lda website https://github.com/primaryobjects/lda. Accessed October 2016.
84. Node-SVM: support vector machine (SVM) library for nodejs. Not Pictured: Mangoes website. https://www.npmjs.com/package/node-svm. Accessed DATE.
85. ConvNetJS: deep learning in your browser. ConvNetJS website. http://cs.stanford.edu/people/karpathy/convnetjs/. Accessed DATE.