Trends in Genetics

CellPress
REVIEWS

Review

Opening the Black Box: Interpretable Machine Learning for Geneticists

Christina B. Azodi,[1,2,*] Jiliang Tang,[3] and Shin-Han Shiu,[1,4,*]

Because of its ability to find complex patterns in high dimensional and heterogeneous data, machine learning (ML) has emerged as a critical tool for making sense of the growing amount of genetic and genomic data available. While the complexity of ML models is what makes them powerful, it also makes them difficult to interpret. Fortunately, efforts to develop approaches that make the inner workings of ML models understandable to humans have improved our ability to make novel biological insights. Here, we discuss the importance of interpretable ML, different strategies for interpreting ML models, and examples of how these strategies have been applied. Finally, we identify challenges and promising future directions for interpretable ML in genetics and genomics.

Importance of Interpretable Machine Learning (ML) and Overview of Strategies

Biological big data [1,2] has driven progresses in fields ranging from population genetics [3] to precision medicine [4]. Much of this progress is possible because of advances in ML (Box 1) [5–10], '[a] field of study that gives computers the ability to learn without being explicitly programmed' [11]. ML works by identifying patterns in data in the form of a **model** (see Glossary) that can be used to make predictions about new data. While powerful, a common criticism is that the ML models are 'black boxes', meaning their internal logic cannot be easily understood by a human [12]. Luckily, strategies to demystify the inner working of ML models are already available and ever improving.

There are three major reasons, troubleshooting, novel insights, and trust, why **interpretable** ML model, or the ability to understand what logic is driving a model's prediction, is important (Figure 1A, Key Figure). First, ML models rarely perform well without tweaking or troubleshooting. Understanding how predictions are made is essential for identifying mistakes or biases in the input data and issues with how the model is **trained**. Second, an ML model with impressive performance may have identified biologically novel patterns. However, such insights will only be available if the model can be interpreted. Third, we are unlikely to trust a prediction if we do not understand why it was made. For example, a doctor may not trust an ML-based diagnosis with no supporting justification out of concern that the model may be capturing artifacts or have unknown biases or limitations [13].

A wide range of strategies for interpretable ML have been developed [14–16] and applied to problems in genetics and genomics. These strategies can be classified based on if they are applicable to all ML **algorithms** (i.e., model-agnostic) or only to one or a subset of algorithms (i.e., model-specific). They can also be classified based on whether they provide **global** or **local interpretations**. Global interpretations involve explaining the overall relationship between **features** and **labels** based on the entire model. While local interpretations focus on explaining the prediction of an **instance** or a subset of instances. For example, imagine you train an ML model to predict if a gene (an instance) is upregulated after some treatment (the label) based on the presence or absence of a set of regulatory sequences (the features). A global interpretation strategy will tell you how important regulatory sequence X is for predicting upregulation across all

Highlights

Machine learning (ML) has emerged as a powerful tool for harnessing big biological data. The complex structure underlying ML models can potentially provide insights into the problems they are used to solve.

Because of model complexity, their inner logic is not readily intelligible to a human, hence the common critique of ML models as black boxes.

However, advances in the field of interpretable ML have made it possible to identify important patterns and features underlying an ML model using various strategies.

These interpretation strategies have been applied in genetics and genomics to derive novel biological insights from ML models.

This area of research is becoming increasingly important as more complex and difficult-to-interpret ML approaches (i.e., deep learning) are being adopted by biologists.

[1]Department of Plant Biology, Michigan State University, East Lansing, MI, USA
[2]Bioinformatics and Cellular Genomics, St. Vincent's Institute of Medical Research, Fitzroy, Victoria, Australia
[3]Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA
[4]Department of Computational Mathematics, Science, and Engineering, Michigan State University, East Lansing, MI, USA

*Correspondence:
cazodi@svi.edu.au (C.B. Azodi) and
shius@msu.edu (S.-H. Shiu).

Box 1. A Crash Course in Machine Learning

ML is when a computer uses data to learn a model for predicting a value, where the relationship between the data and the value is not explicitly provided. The data is composed of instances (i.e., samples) and feature (i.e., independent variables) that describe those instances. For example, if our instances are genes, features describing those genes could be the GC content, the presence or absence of a specific functional domain, or its level of conservation across species. If the values being predicted are not known *a priori* for any instance, then unsupervised ML approaches (e.g., clustering) can be applied to extract previously unknown patterns. If the values being predicted are known for some of the instances, these values are referred to as labels and one can learn from these labels using a supervised ML approach. In this review, we focus on supervised ML. Finally, if the known labels are categorical (e.g., is the gene upregulated or downregulated), it is a classification problem, while if the labels are continuous (e.g., gene expression levels), it is a regression problem.

A common supervised ML workflow involves four steps: training, applying, scoring, and interpretation (Figure I). First, input data made up of features and labels for many instances are divided into a training set and a testing set. The features and labels from the training set are then used to train the ML model. During training, the ML model learns the combination of internal parameters that minimize the error in the predictions of the labels. Second, the trained ML model is applied to the testing set features to generate predicted labels. A trained ML model can also be applied to unlabeled instances to make predictions. Third, the performance of the ML models is scored by comparing the predicted labels with the known labels from the test set. Many different performance metrics are used in the ML field, where the best metric depends on the type of ML problem and the nature of the question being asked. A performance metric not only informs the quality of a model, but also provides a quantitative measure of how much we known about the biological phenomenon in question given the features used. Finally, the ML model is interpreted to provide a better, quantitative understanding on how the input features contribute to the predictions.

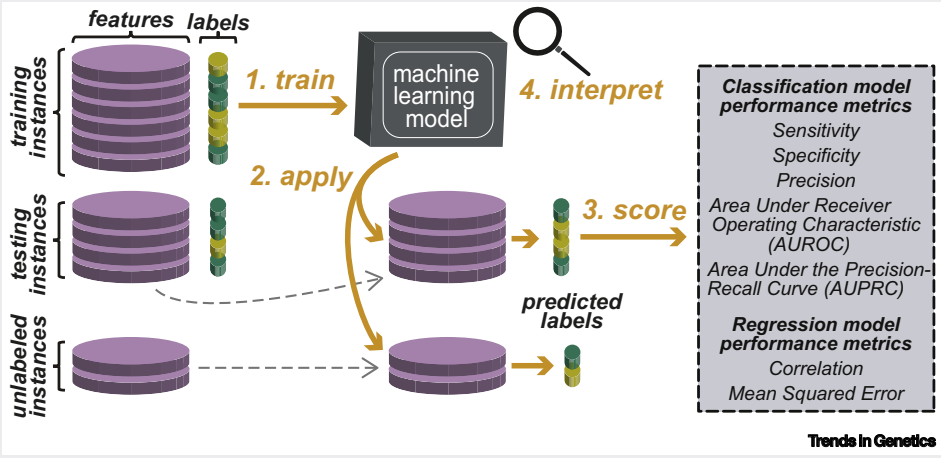


Figure I. A Supervised Machine Learning Workflow.

Glossary

Algorithm: the procedure taken to solve a problem/build a model.
Decision tree: a model made up of a hierarchical series of true/false questions.
Deep learning: a subset of ML algorithms inspired by the structure of the brain that can find complex, nonlinear patterns in data.
Feature: an explanatory (i.e., independent) variable used to build a model.
Global interpretation: a type of ML interpretation that explains the overall relationship between the features and the label for all instances.
Instance: a single example from which the model will learn from or be applied to.
Interpretable: capable of being understood by a human.
Label: the variable to be predicted (i.e., the dependent variable).
Local interpretation: an ML interpretation that explains the relationship between the features and the label for one or a subset of instances.
Model: the set of patterns learned for a specific problem, where given input (i.e., instances and their features) the model will generate an output (i.e., prediction).
Model performance: a quantitative evaluation of the model's ability to correctly predict labels.
Parameters: variables in an ML model whose values are optimized during training.
Perturbing strategies: a family of interpretation strategies that measure how changes in the input data impact model predictions or performance.
Probing strategies: a family of interpretation strategies that involve inspecting the structure and parameters in a trained model.
Surrogate strategies: a family of interpretation strategies that involve training an inherently interpretable model (e.g., a linear model) using the same data as a black box model to approximate the predictions of the black box model.
Training: the process of identifying the best parameters to make up a model: the learning part in ML.

genes in your dataset. While a local interpretation strategy will tell you how important regulatory sequence X is for predicting gene Y as upregulated. This means that the type of interpretation strategy you select will dictate what you will learn from your ML model, with different strategies possibly telling different stories. We should emphasize that most ML models identify association through correlation (with exceptions, see [17]). Thus, ML interpretation strategies mostly do not identify causal relationships between input features and labels. Instead, interpretations should be used to generate hypotheses of cause-effect relations that can be tested experimentally. We will review three general ML interpretation strategies: **probing**, **perturbing**, and **surrogate strategies** (Figure 1B–D) [14,16].

Probing Strategies Dissect the Inner Structure of ML Models

Training an ML model involves identifying the set of **parameters** best able to predict the label of an instance (e.g., gene Y is upregulated). After training, these parameters can be 'probed' (i.e., inspected) to better understand what the model learned (Figure 1B). Probing strategies

Key Figure

Overview of Machine Learning (ML) Model Interpretation Strategies

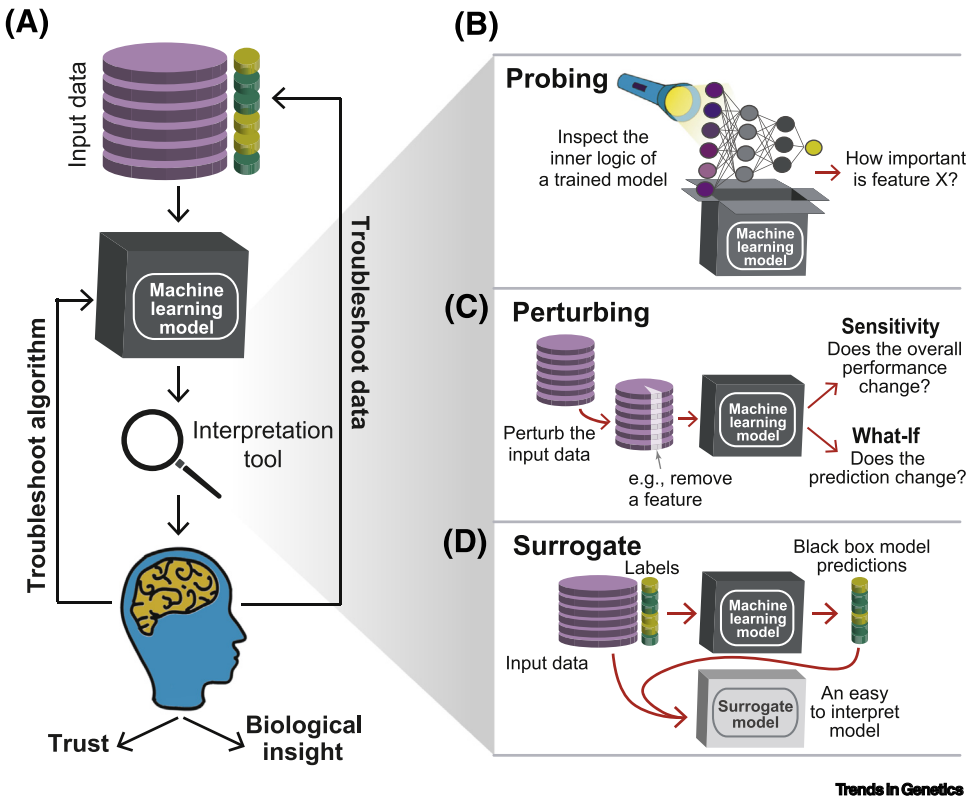
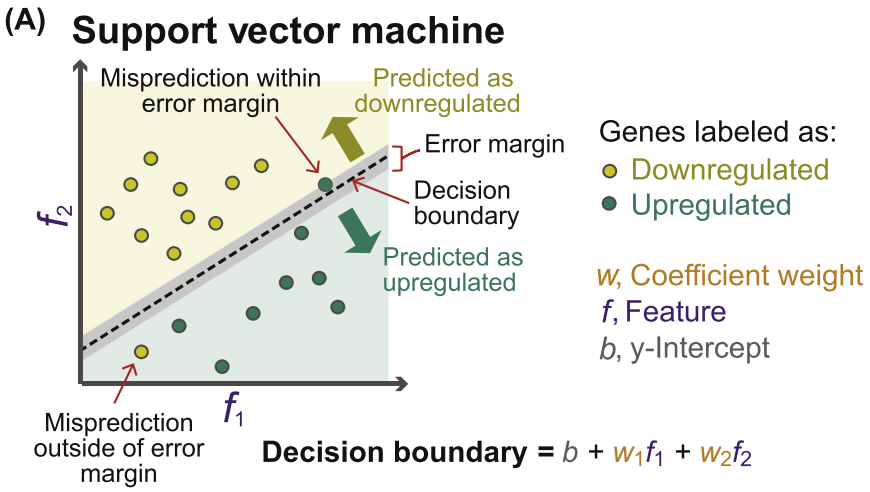


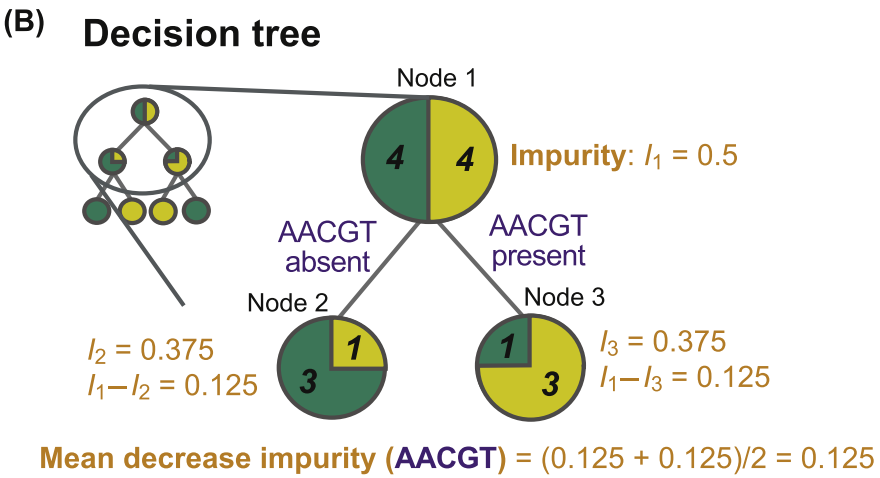
Figure 1. (A) Understanding the inner logic of an ML model (i.e., model interpretability) is important for troubleshooting during model training, generating biological insights, and instilling trust in the predictions made. (B) There are three general strategies for interpreting an ML model: probing, perturbing, and surrogates. Probing strategies involve inspecting the structure and parameters learned by a trained ML model (e.g., a deep learning model pictured here) in order to better understand what features or combination of features are important for driving the model's predictions. Perturbing strategies involve changing values of one or more input features (e.g., setting all values to zero) and measuring the change in model performance (sensitivity analysis) or on the predicted label of a specific instance (what-if analysis). Finally, an easily interpretable model (e.g., linear regression or decision tree) can be trained to predict the predictions from ML models, acting as a surrogate.

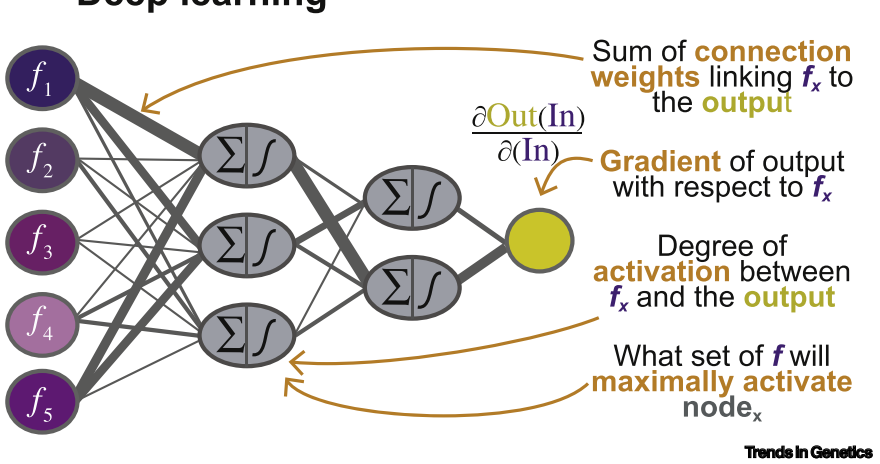
provide global interpretations with some exceptions (see later). Because the type of parameters (e.g., coefficient weight, decision nodes) and how those parameters relate to each other (e.g., linear combination, hierarchical network) varies by algorithm, probing strategies are model-specific. Probing strategies for many classical ML algorithms [e.g., support vector machine (SVM) and **decision tree**-based algorithms] have been around for decades and are relatively straightforward to understand and implement. Probing strategies for more complex ML algorithms (e.g., **deep learning**), while already available, are less well established, with new strategies published each year and no consensus currently on a best approach.

Probing SVM models

SVM is an algorithm that finds the hyperplane that best separates instances by their label when they are plotted in n-dimensional space (n = number of features; n = 2 in Figure 2A). Training







(See figure legend at the bottom of the next page.)

an SVM model to predict gene upregulation using regulatory sequences as features means learning the combination of coefficient weights to apply to regulatory sequences in order to make the best hyperplane. SVM models can be trained to learn either linear or nonlinear relationships between features and labels.

A linear SVM model is probed by extracting the coefficient weights that define the hyperplane (Figure 2A), where features assigned a higher absolute weight have a stronger relationship with the label and thus are more important for driving the prediction. For example, a linear SVM model was trained to classify simulated populations as being under positive or negative selection using genetic markers as features [18]. Coefficient weights from the trained SVM model were used to quantify the importance, where genetic markers with large, positive coefficient weights were considered positively selected. Validating this approach, they found that the SVM-based coefficient weights were highly similar to weights derived from a classical population genetics statistical test (Tajima's *D*). Importantly, the above linear SVM probing strategy, like other strategies discussed in this review, can provide an incomplete picture of feature importance. For example, two highly correlated features will split the weight between them, reducing their perceived importance. Therefore, understanding the limitations of any ML interpretation strategy is critical.

In addition to the issue with correlated feature, a feature with a strong, nonlinear, relationship with the label may not have large coefficient weight in a linear SVM model. Fortunately, SVM can also be used to learn nonlinear relationships with what is called the 'kernel trick', where nonlinearly separable data is projected into a higher-dimensional space where it can be separated by a linear hyperplane [19]. Because different kernels project the data in different ways, probing strategies for nonlinear SVM models tend to be kernel specific [20]. For example, string kernels (e.g., [21–23]) are a family of kernels frequently used in genetics because they can project DNA or protein sequences into a space where each dimension represents a subsequence of a defined length (i.e., a motif) and the values in that dimension represent the number of occurrences of that motif in each instance. Because each dimension represents a sequence motif, the trained coefficient weights describe the relationship between sequence motifs and the label, where motifs assigned large absolute coefficient weights are interpreted as more important for the prediction. For example, Sonnenburg *et al*. trained an SVM model using a variant of the string kernel that also considers positional information [22] to predict splice sites in *Caenorhabditis elegans* [24]. By visualizing the trained coefficient weights, they confirmed the well-established acceptor splice site motif and discovered a novel donor splice site and motif ~43 base pairs upstream of the acceptor splice site. They also demonstrated the utility of perturbation-based interpretation strategies, which we describe later.

Figure 2. Probing a Trained Machine Learning (ML) Model. An ML model that classifies upregulated (green) from downregulated (yellow) genes using regulatory sequences (purple) as features can be probed to find what regulatory sequences are most important for predicting differential expression. (A) A support vector machine model learns the combination of coefficient weights (*w*; orange) forming the decision boundary (dotted line) best able to separate upregulated from downregulated genes, where the features assigned the higher *w* are more important. The decision boundary is a hyperplane represented by the equation shown. (B) A decision tree-based model learns the most predictive series of true/false questions about the features. We zoom in on a node where the regulatory sequence 'AACGT' is used as the feature. How well AACGT separates upregulated from downregulated genes is quantified by calculating the mean decrease in node impurity after AACGT is used at a branch point. (C) Deep learning models train to learn what combinations of connection weights (gray lines) across all nodes and layers result in the best network to classify upregulated from downregulated genes. A trained deep learning model can be probed by inspecting the size of the connection weights (gray line thickness), measuring the gradient of the output with respect to the input [i.e. ∂Out(in)/∂ (in)], and quantifying the extent to which different features cause a node to activate (represented by the light switch).

Probing Decision Tree-Based Models

A decision tree is a set of true/false questions nested in a hierarchical structure (Figure 2B). They are inherently interpretable because the content and order of each question can be directly observed. How well a true/false question separates instances by their label can also be quantified using metrics such as the mean decrease in node impurity, where a node containing instances with all the same labels has an impurity equal to zero and nodes containing instances with a mixture of labels have an impurity greater than zero. For the example in Figure 2B, using the presence/absence of regulatory sequence 'AACGT' to separate up- from downregulated genes results in a decrease in the mean node impurity, meaning this feature was informative in classifying up and downregulated genes. Because single decision trees tend to perform poorly at predicting complex patterns, ensemble approaches (e.g., random forest [25], gradient tree boosting [26,27]), where many decision trees are combined to generate one prediction, are often used. Ensemble decision-tree models can be probed by calculating the mean decrease in node impurity for each feature across all trees in the ensemble. This approach has been used broadly in genetics applications using random forest-based models, including for identification of important gene regulators from gene regulatory networks (e.g., integrative Random forest for Gene Regulatory Network inference, iRafNet [28]) and DNA motifs for predicting differential gene expression [29].

The hierarchical structure of decision tree-based models means that interactions between features can also be readily probed. For example, using a tool for finding stable feature interactions in random forest models [30], interactions between genomic, transcriptomic, and epigenomic features were identified that were predictive of deleterious genetic variants [31]. Specifically, an interaction between the local GC content and the distance to the nearest expression quantitative trait loci was important for predicting deleterious variants.

As with coefficient weights from SVM models, mean decrease impurity scores can be misleading when features are highly correlated. This score also tends to inflate continuous features (e.g., expression level values) over categorical features (e.g., high, median, or low expression levels), categorical features with a larger number of categories, and continuous features with a larger numeric range and should therefore be interpreted with caution when the feature space is not uniform [32].

Probing Deep Learning Networks

While the classical ML algorithms described above are useful and readily interpretable, deep learning (Box 2) algorithms are being applied more and more in the ML community because they frequently outperform classical ML algorithms at modeling complex systems when sufficient training data is available [33–35] and they can learn from raw data (e.g., whole DNA sequence) rather than user defined features (e.g., known regulatory sequences). However, in modeling there is often a tradeoff between predictability and interpretability [36]. Fortunately, there has been a substantial effort to develop new methods to probe these complex models. These methods can be categorized into three types: connection weights-based, gradient-based, and activation level-based approaches (Figure 2C) [15].

Connection weight-based feature importance scores quantify the global relationship between each feature and the output (i.e., the label) by summing the learned weights assigned to connections between each input feature and the output layer [37,38]. Following the path through the example artificial neural network (Figure 2C), the sum of connection weights (represented by line widths) between the feature f_1 and the output layer is larger than the sum of connection weights between the feature f_3 and the output layer, indicating f_1 is more important than f_3 for that model.

Box 2. A Crash Course in Deep Learning (DL)

ML algorithms inspired by the structure of the brain make up a subfield of ML called DL. DL is promising for biology because DL models can: (i) learn highly complex nonlinear patterns; (ii) continue to improve when given more training data ('shallow' ML models tend to plateau); and (iii) they can learn from raw data without user defined features [60]. A DL model is made up of multiple layers of nodes connected by edges of different connection weights (w_x) (Figure IA). The nodes in the input layer contain the feature values (f_x) for an instance. The nodes in the hidden layers (hidden nodes) represent the sum of the nodes from the previous layer multiplied by their associated connection weights ($\sum w_x f_x$). The node value from that summation is then passed through an activation function (represented as a light switch), which determines the extent to which that node gets turned on (i.e., activated). DL models can learn nonlinear relationships when the activation function used is nonlinear (e.g., the sigmoid function). The output node (i.e., the predicted label) is the sum of the nodes from the last hidden layer and can be compared with the true label to calculate the error in the model. A DL model is trained by propagating that error back through the model and updating the learned connection weights (i.e., backpropagation of the error) until that error is minimized.

This type of DL algorithm, often referred to as a fully connected artificial neural network, is widely useful for modeling complex, nonlinear relationships. Other DL algorithms many be useful for addressing specific types of biological questions (Figure IB). For example, convolutional neural networks learn spatial patterns making them ideal for identifying sequence motifs and patterns in images. Recurrent neural networks remember earlier predictions and are therefore ideal for sequential data or sequential sequence analysis. In adversarial learning, two DL models are trained in a sort of arms race, one learning to generate synthetic data and the second learning to discern real from synthetic data, thus making them ideal for synthetic biology.

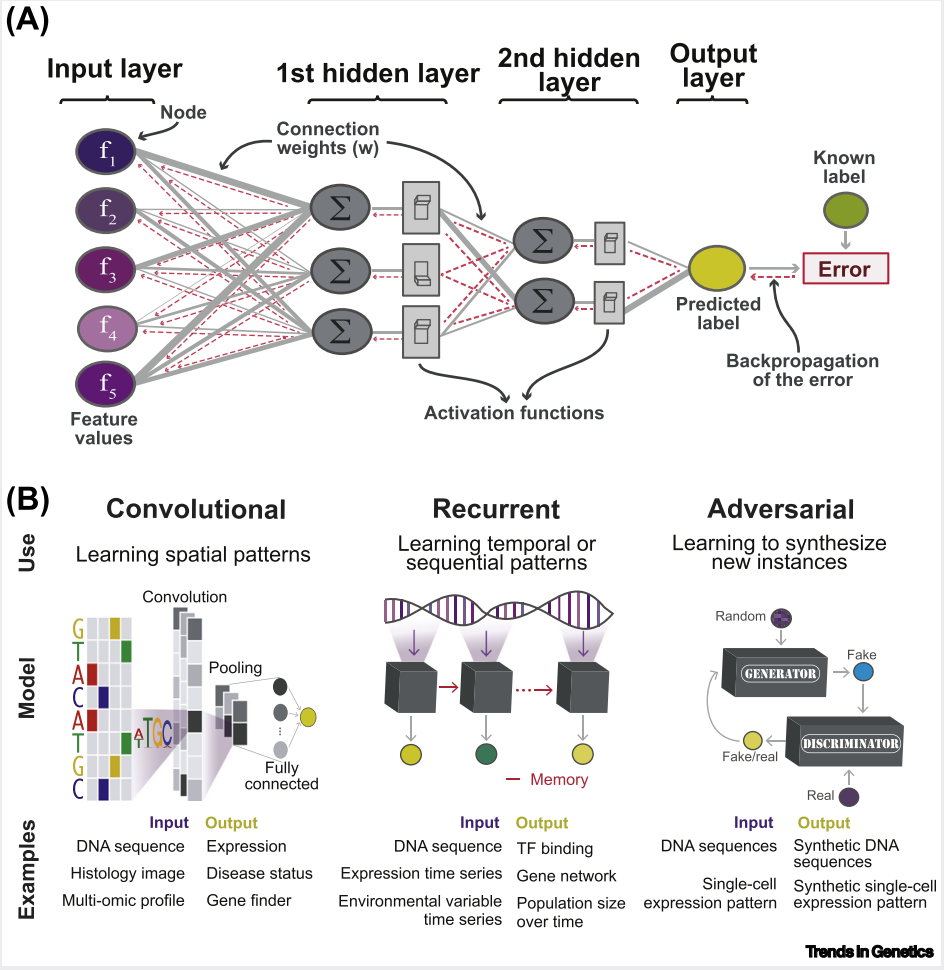


Figure I. Graphical Explanations of Deep Learning Algorithms. (A) An example of a fully connected artificial neural network. (B) Uses, graphical explanations, and example biological applications for three additional deep learning algorithms: convolutional neural networks, recurrent neural networks, and adversarial learning. Abbreviations: TF, Transcription factor.

This approach was used to determine which microRNA features were the most important for predicting the expression level of Smad7, a gene involved in disrupting a signaling process upregulated in patients with breast cancer [39]. Connection weight-based feature importance scores can be misleading when features are on different scales, when positive and negative connection weights cancel each other out, or when a connection has a large weight but is rarely activated (i.e., the node is rarely turned on) [40].

The gradient-based feature importance scores (a.k.a. saliency) also quantify the global relationship between a feature and the output, but do so by calculating the gradient, or the change in the predicted output (e.g., the likelihood a gene is upregulated) as small changes are made to the input feature (e.g., the frequency of regulatory sequence X). The gradient is calculated using the partial derivative [41]. This approach was used to identify putative distal regulatory sequences in genomic regions where positive and negative gradient-based importance score peaks represented enhancer and silencer regions, respectively [42]. This approach, however, is not useful when small changes in the feature value do not change the output prediction [40].

The third type of approach probes the activation levels. The activation level refers to the output value from a node after it has passed through the activation function (Figure 2C, Box 2). Activation level-based feature importance scores provide a local interpretation for an instance of interest by comparing how much the value of a feature from that instance activates nodes in the trained network compared with the feature values from a reference instance. A reference instance for an image classification model could be one that is solid white, while a reference for a model using a DNA sequence as instances could be an instance with the background nucleotide frequency at every site. This approach, called DeepLIFT [40], has been used in multiple biological studies [43–45]. For example, Zuallaert *et al.* used DeepLIFT to find nucleotide sequences important for predicting splice sites [44]. Because DeepLIFT probes activation levels rather than connection weights, it avoids the pitfall of the connection weight-based approach. Further, because it compares a specific instance with a reference, it also avoids the pitfalls of the gradient-based approach.

In addition to these three approaches for determining feature importance, another way to probe deep learning models is to learn what pattern each node in the network learned to identify. This can be done by finding real or simulated instances that maximally activate that node, then the properties of those real or simulated instances can be used to interpret that node (Figure 2C). For example, if the ten DNA sequences that cause node X to have the maximum possible output value after passing through the activation function all contain the motif ACGG TC, one could interpret that node X is trained to find the ACGGTC motif. Because probing every node in every layer may produce results that are still too complex to interpret, dimensionality reduction techniques can be used to ease interpretation. For example, Esteva *et al.* trained a deep convolutional neural network [46] (Box 2) to diagnose different types of skin cancer from photos [47]. Because of the high level of complexity in their model (it contained 97 convolutional layers), visualizing the activation level at every node would be impossible. Therefore, they used a dimensionality reduction technique to project, for each instance, the output from the last hidden layer (containing 2048 nodes) into two dimensions. This allowed them to visualize how their convolutional neural network learned to separate different types of carcinomas.

Perturbing Strategies for Interpreting ML Models

Perturbing strategies involve modifying the input data and observing some change in the model output (Figure 1C). Because modifications to the input data can be made regardless of the ML

algorithm used, perturbing strategies are generally model agnostic. We discuss two general perturbation-based strategies: sensitivity analysis and what-if methods (Figure 3).

Sensitivity Analysis

Sensitivity analysis involves modifying an input feature and measuring the impact on **model performance** (Figure 3A). Feature modification typically means removing (i.e., leave-one-feature-out) or permuting (e.g., set all values to the mean) one feature at a time. The decrease in model performance after a feature is removed or permuted is an intuitive score for each feature indicating its contribution to the predictions (Figure 3A). Because perturbing a feature not only impacts that feature but also other features that interact with it, sensitivity analysis also captures interaction effects for each feature. However, sensitivity analysis can miss important features if correlation exists in the feature set. For example, if features X and Y are highly correlated, feature Y could compensate when X is removed or permuted, masking its potential importance.

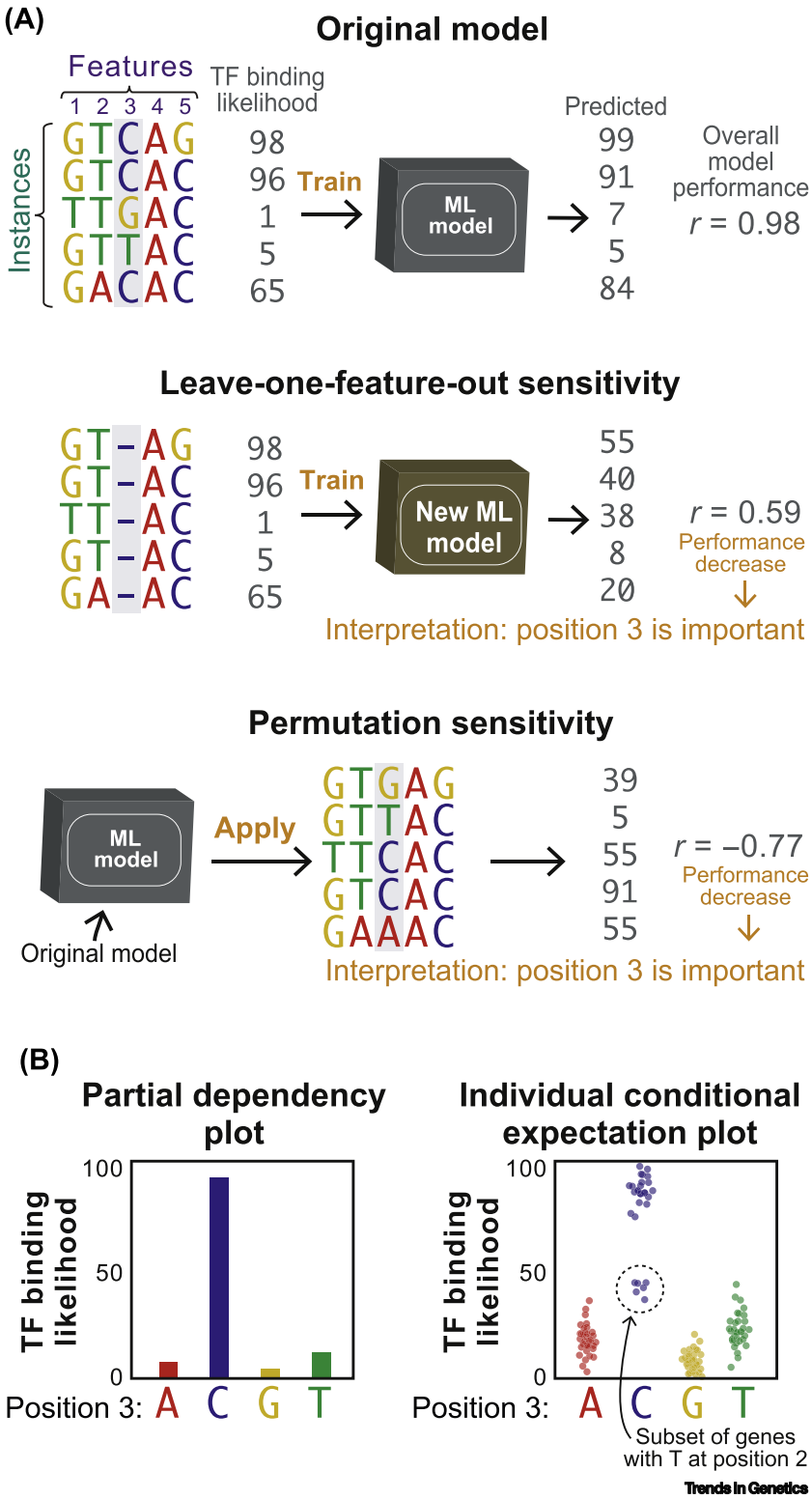
Che *et al.* used the leave-one-feature-out approach on their ensemble decision tree models to find that genomic region length was the most important feature for identifying genomic regions that contain clusters of genes acquired by horizontal gene transfer [48]. While the leave-one-feature-out approach provides an intuitive measure of importance, it is computationally expensive because it typically requires training a new model for every perturbed dataset. Therefore, it is not often used to interpret deep learning model (which are already computing intensive) except when there are few input features. For example, leave-one-feature-out was used to determine that, of five histone marks, removing H3K4me3 resulted in the largest decrease in a deep learning model's ability to predict transcription factor (TF) binding sites [49].

Permutation-based approaches determine feature importance scores by measuring how the performance of an ML model changes when different features are randomly permuted. They are more computationally efficient than leave-one-feature-out strategies because only one model needs to be trained. This makes permutation a useful strategy when feature importance scores need to be calculated iteratively. For example, a strategy called 'gene shaving' can be used to identify minimum sets of marker genes needed to predict a label (e.g., a disease state) by iteratively removing the least important genes from the model until performance suffers. Deng *et al.* trained gene shaving random forest models using permutation-based importance scores to identify a minimum set of lung cancer marker genes [50]. Permutation-based importance scores are also particularly intriguing for genetic studies because the logic is similar to DNA mutagenesis experiments. It was demonstrated that *in silico* mutagenesis (i.e., computationally permuting DNA sequences) could identify which nucleotides impact tissue-specific gene expression the most [51]. Finally, the permutation-based strategy can also be used in image analysis, where it is called occlusion sensitivity. Here, different regions in images are grayed out and the resulting change in performance is measured. For example, occlusion of regions of blood smear images confirmed that a malaria classification model performed worst when parasitized regions were grayed out [52].

What-if Analysis

The what-if approach (also referred to as counterfactuals [53]) measures how the prediction of a particular instance changes (rather than the overall model performance) when the input value for one or more features is changed. Thus, what-if analysis provides local interpretations while sensitivity analysis provides global interpretations. Here, we focus on two what-if methods: partial dependency plots (PDPs) and individual conditional expectation (ICE) plots (Figure 3B [16]).

PDPs show how a prediction changes when the input value for a feature of interest is changed, marginalizing (i.e., ignoring) the effects of all other features [27]. Imagine we trained an ML



(See figure legend at the bottom of the next page.)

Table 1. Platforms and Software Available for Interpretable Machine Learning

Name	Strategy	Use	Scope	Description	Platform	Refs
CamurWeb	Probing	Decision tree-based models	Global	Interpret decision rules from classifier with alternative and multiple rule models	Web-based tool	[61]
DeepTRIAGE	Probing	Attention-based deep learning	Local	Deep learning for the tractable individualized analysis of gene expression	Python package	[62]
iml	Probing, perturbing	Model agnostic	Global, local	Interpretable machine learning: toolbox for implementing multiple interpretation methods	R package	[16]
iNNvestigate	Probing	Deep learning	Global, local	Toolbox for implementing multiple interpretation methods for neural network (NN) models	Keras	[63]
iRF	Probing	Random forest	Global	Iterative random forest: decision tree-based method to identify significant feature interactions	R package	[30]
LIME	Surrogate	Model agnostic	Local	Local interpretable model-agnostic explanations: a tool to generate local surrogates for black box models	Python package	[58]
Lucid	Probing	Deep learning	Global, local	Toolbox of methods for visualizing and interpreting neural networks	Tensorflow	[i]
NeuralNetTools	Probing, perturbing	Deep learning	Global, local	Toolbox for implementing multiple interpretation methods	R package	[64]
poim2motif	Perturbing	Nonlinear SVM	Global	Positional oligomer importance matrices. A framework for extracting important sequence motifs from weighted degree kernel SVM models	Python package	[65]
SpliceRover	Probing	Deep learning	Local	Tool to interpret which nucleotides contribute most predicting splice sites using DeepLIFT	Web-based tool	[44]
The What-If Tool	Probing, perturbing	Model agnostic	Global, local	Code free toolbox for assessing, comparing, and interpreting Tensorflow/python-based ML models	TensorBoard, Jupyter, Colaboratory notebooks	[i]

model that predicts the likelihood that a sequence will be bound by a TF. A PDP could show, for example, how the TF-binding likelihood of the sequence in question would change if the nucleotide at the position of interest is changed from C to A, G, or T (left panel, Figure 3B). For example, this approach was used to demonstrate the impact of RNA sequence features (e.g., *k*-mer abundance) on the predicted likelihood of ribosome recruitment to an mRNA [54]. However, PDPs can miss important features when there are interactions between features. Imagine if a C at position 3 increased TF binding affinity when position 2 contained a T, but decreased binding affinity if position 2 contained an A. Because position 2 is marginalized in the PDP of position 3, the interaction may mask the importance of position 3.

ICE plots were proposed to address this limitation of PDPs [55]. ICE plots are essentially PDPs generated for every individual instance in the dataset. For example, an ICE plot for position 3

Figure 3. Perturbing the Input to a Machine Learning (ML) Model. An example ML model predicting if a transcription factor (TF) may bind (i.e., the label) to a specific sequence (i.e., the features) can be interpreted with perturbing strategies. (A) Sensitivity analysis. Leave-one-feature-out means a new ML model is trained on the same input data with one feature (e.g., position 3) removed. Then the overall performance of the original model and the new model are compared. Permutation means the original model is applied to input data with the values permuted (e.g., shuffled) for one feature at a time. The performance of the model applied to the original and the shuffled data are compared. Both sensitivity analyses on position 3 shown here resulted in a decrease in performance, leading to the interpretation that position 3 is important for TF binding. (B) What-if analysis. The partial dependency plot (left) shows the TF binding likelihood if position 3 was an A, C, G, or T, ignoring the effects of nucleotides at other positions. This plot shows that a C at position 3 increases the likelihood of TF binding. The individual conditional expectation plot (right) shows the TF binding likelihood score for every instance (dot) in the dataset when position 3 is A, C, G, or T. This plot shows when position 3 is C, the binding likelihoods have a bimodal distribution, which is due to interaction with position 2 in this hypothetical example.

would show that the presence of a C at position 3 only increases the TF binding likelihood in a subset of sequences, which, with further investigation, are the sequences with a T in position 2 (right panel, Figure 3B). Because they show how changing a feature value changes the prediction of each instance, ICE plots can uncover interactions or group-specific effects that may be of biological importance. Because this strategy does not require model retraining, it is well suited for interpreting deep learning models. For example, ICE plots were used to better understand what patterns of gene expression an adversarial deep learning model (see Figure IB in Box 2) learned were characteristic of single cell data (referred to as sensitivity plots [56]). By varying the expression level of individual genes (the feature) within the single cell (the instance), they found the genes with the biggest impact on the prediction (real or simulated) were genes known to be markers for particular cell-type states.

What-if analyses can provide highly detailed and intuitive interpretations of ML models, including the magnitude, direction, and nonlinearities in the relationships between features and the output label. A limitation is that PDP and ICE plots can only be visualized for one or two features at a time, so they are typically only generated for models with few features or for a subset of features deemed important by another interpretation strategy or from domain knowledge [57].

Surrogate Strategies for Interpreting ML Models

Image you have an ML model that is truly a black box, meaning that it cannot be probed, and perturbation strategies do not provide useful information. In such a case, one can train a more interpretable model to approximate the black box model. Examples of interpretable models include linear models where coefficients reflect feature importance, or decision trees where mean decrease node impurity can be calculated. These inherently interpretable models are referred to as surrogate models. For example, to generate a surrogate model for a black box model that can predict gene upregulation using regulatory elements as features, we would first apply the black box model to a set of genes, *G*, and extract the black box predicted label (i.e., up- or downregulated) for those genes (Figure 1B). Then we would use the same set of genes *G* as the instances and the black box predicted labels as the labels to train a surrogate.

One major limitation of surrogate models is that black box models are often highly complex (e.g., highly nonlinear, many higher order interactions) and, thus, cannot be fully learned by an interpretable surrogate. To overcome this, one approach is to generate a surrogate to learn just a portion of the black box model, known as a local interpretable model-agnostic explanation (LIME) [58]. While the complex logic underlying the whole model may be too much for a surrogate model to learn, the logic for one instance or a group of similar instances (e.g., coexpressed genes), hence local, may be simple enough. For example, LIME was used to better understand why some patients were misclassified by a black box model predicting survival after cardiac arrest [59]. A LIME model for a patient that was mispredicted to survive showed that the black box model was too heavily influenced by certain features (e.g., healthy neurologic status, lack of chronic respiratory illness) and did not place sufficient weight on other features that are also important (e.g., elevated creatinine, advanced age).

Concluding Remarks

Interpretability is critical for applications of ML in genetics and beyond and will therefore see substantial advances in the coming years. Just like there is no one universally best ML algorithm, there will not likely be one ML interpretation strategy that works best on all data or for all questions. Rather, the interpretation strategy should be tailored to what one wants to learn from the ML model and confidence in the interpretation will come when multiple approaches tell the same story. Luckily, many user-friendly tools have already been developed to facilitate

Outstanding Questions

How can we interpret ML models trained on heterogeneous (e.g., multi-omic) and high dimensional (number of features >> number of instances) data? ML algorithms are well suited to take advantage of the large-scale multi-omic data for generating predictive models. However, interpreting ML models trained on high dimensional and heterogenous data remains challenging. These challenges are exasperated when features are highly correlated, dependent, and of different types (e.g., continuous, binary, or categorical).

What ML modeling and interpretation strategies are best for studying complex biological systems? Given the importance of nonlinear effects in biology (e.g., epistatic interactions, feedback loops, community dynamics, synergistic/antagonistic effects), interpretation strategies that can tease out these complex, nonlinear relationships are critical.

How can we benchmark existing and new interpretation strategies for applications in genetics and genomics? Availability of high quality, benchmark dataset will be crucial. In addition, the strategies used to interpret an ML model may identify similar or different underlying logic and characteristics of the model. How can we compare ML interpretation strategies and results? Further, how could we join the findings from multiple strategies into a more complete and coherent interpretation of that model?

How can interpretable ML be accessible for biologists? Using and implementing ML interpretation strategies can require significant computational knowledge. What roles will interdisciplinary training in computational and data science and user-friendly software play in encouraging the interpretation of ML models in genetics and genomics?

How can biologists ensure that model interpretability will continue to be an area of development for folks working in the artificial intelligence (AI) field? As the power and precision of ML models improves, more and more trust will likely be placed in them. What role can biologists play in shaping the future of trustworthy AI, particularly in the context of model interpretability?

interpreting ML models using the strategies described in this review and more (Table 1). Critically, the insights that can be learned from interpreting an ML model are constrained by the content, quality, and quantity of the data used to generate the model. Care should be taken when selecting data and features to avoid introducing technical or biological artifacts into the models and, thus, into the interpretations.

There are still many challenges to interpreting ML models in genetics and genomics (see Outstanding Questions). These challenges, while not necessarily unique to genetics or genomics, represent opportunities for computational biologists to innovate and contribute novel solutions. They also highlight the importance of training the next generation of biologists able to work at the intersection of computer and biological science.

Acknowledgments

We thank Yuying Xie for his insightful feedback on this review. This work was partly supported by the National Science Foundation (NSF) Graduate Research Fellowship (Fellow ID: 2015196719) to C.B.A.; NSF (IIS-1907704, IIS-1845081, CNS-1815636) grants to J.T.; and the US Department of Energy Great Lakes Bioenergy Research Center (BER DE-SC0018409) and NSF (IOS-1546617, DEB-1655386) grants to S-H.S.

Resources

[i] https://pair-code.github.io/what-if-tool/index.html

[ii] https://github.com/tensorflow/lucid

References

1. Marx, V. (2013) Biology: the big challenges of big data. *Nature* 498, 255–260
2. Stephens, Z.D. *et al.* (2015) Big data: astronomical or genomical? *PLoS Biol.* 13, e1002195
3. Schrider, D.R. and Kern, A.D. (2018) Supervised machine learning for population genetics: a new paradigm. *Trends Genet.* 34, 301–312
4. Alyass, A. *et al.* (2015) From big data analysis to personalized medicine for all: challenges and opportunities. *BMC Med. Genet.* 8, 33
5. Angermueller, C. *et al.* (2016) Deep learning for computational biology. *Mol. Syst. Biol.* 12, 878 816
6. Chicco, D. (2017) Ten quick tips for machine learning in computational biology. *BioData Min.* 10, 35
7. Cuperlovic-Culf, M. (2018) Machine learning methods for analysis of metabolic data and metabolic pathway modeling. *Metabolites* 8, 4
8. Libbrecht, M.W. and Noble, W.S. (2015) Machine learning applications in genetics and genomics. *Nat. Rev. Genet.* 16, 321–332
9. Ma, C. *et al.* (2014) Machine learning for big data analytics in plants. *Trends Plant Sci.* 19, 798–808
10. Tarca, A.L. *et al.* (2007) Machine learning and its applications to biology. *PLoS Comput. Biol.* 3, e116
11. Samuel, A.L. (1959) Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.* 3, 210–229
12. Lipton, Z.C. (2018) The mythos of model interpretability. *Comm. ACM* 16, 10
13. Miller, T. (2019) Explanation in artificial intelligence: insights from the social sciences. *Artif. Intell.* 267, 1–38
14. Guidotti, R. *et al.* (2018) A survey of methods for explaining black box models. *ACM Comput. Surv.* 51, 1–42
15. Montavon, G. *et al.* (2018) Methods for interpreting and understanding deep neural networks. *Digit. Signal Process.* 73, 1–15
16. Molnar, C. (2019) *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. (1st edn), leanpub.com
17. Peters, J. *et al.* (2017) *Elements of Causal Inference*, MIT Press
18. Ronen, R. *et al.* (2013) Learning natural selection from the site frequency spectrum. *Genetics* 195, 181–193
19. Ben-Hur, A. *et al.* (2008) Support vector machines and kernels for computational biology. *PLoS Comput. Biol.* 4, e1000173
20. Barakat, N. and Bradley, A.P. (2010) Rule extraction from support vector machines: a review. *Neurocomputing* 74, 178–190
21. Leslie, C. *et al.* (2002) The spectrum kernel: a string kernel for SVM protein classification. In *Pacific Symposium on Biocomputing*, pp. 564–575
22. Schölkopf, B. *et al.* (2004) Accurate splice site detection for *Caenorhabditis elegans*. In *Kernel Methods in Computational Biology*, MIT Press
23. Ghandi, M. *et al.* (2014) Enhanced regulatory sequence prediction using gapped k-mer features. *PLoS Comput. Biol.* 10, e1003711
24. Sonnenburg, S. *et al.* (2008) POIMs: positional oligomer importance matrices—understanding support vector machine-based signal detectors. *Bioinformatics* 24, i6–i14
25. Breiman, L. (2001) Random forests. *Mach. Learn.* 45, 5–32
26. Schapire, R.E. (2003) The boosting approach to machine learning: an overview. In *Nonlinear Estimation and Classification* (Denison, D.D. *et al.*, eds), pp. 149–171, Springer
27. Friedman, J.H. (2001) Greedy function approximation: a gradient boosting machine. *Ann. Stat.* 29, 1189–1232
28. Petralia, F. *et al.* (2015) Integrative random forest for gene regulatory network inference. *Bioinformatics* 31, i197–i205
29. Uygun, S. *et al.* (2019) Cis-regulatory code for predicting plant cell-type transcriptional response to high salinity. *Plant Physiol.* 181, 1739–1751
30. Basu, S. *et al.* (2018) Iterative random forests to discover predictive and stable high-order interactions. *Proc. Natl. Acad. Sci. U. S. A.* 115, 1943–1948
31. Vervier, K. and Michaelson, J.J. (2018) TiSAn: estimating tissue-specific effects of coding and non-coding variants. *Bioinformatics* 34, 3061–3068
32. Strobl, C. *et al.* (2007) Bias in random forest variable importance measures: illustrations, sources and a solution. *BMC Bioinforma.* 8, 25
33. Banerjee, S. *et al.* (2017) Performance of deep learning algorithms vs. shallow models, in extreme conditions - some empirical studies. In *Pattern Recognition and Machine Intelligence*, pp. 565–574, Springer
34. Guo, Y. *et al.* (2016) Deep learning for visual understanding: a review. *Neurocomputing* 187, 27–48

35. LeCun, Y. *et al.* (2015) Deep learning. *Nature* 521, 436–444
36. Freitas, A.A. (2014) Comprehensible classification models: a position paper. *ACM SIGKDD Explor. Newsl.* 15, 1–10
37. Garson, D.G. (1991) Interpreting neural network connection weights. *AI Expert.* 6, 46–51
38. Olden, J.D. and Jackson, D.A. (2002) Illuminating the "black box": a randomization approach for understanding variable contributions in artificial neural networks. *Ecol. Model.* 154, 135–150
39. Manzanarez-Ozuna, E. *et al.* (2018) Model based on GA and DNN for prediction of mRNA-Smad7 expression regulated by miRNAs in breast cancer. *Theor. Biol. Med. Model.* 15, 24
40. Shrikumar, A. *et al.* (2017) Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning* (70), pp. 3145–3153
41. Simonyan, K. *et al.* (2013) Deep inside convolutional networks: visualising image classification models and saliency maps. *arXiv.* Published online December 20, 2013. http://doi.org/abs/1312.6034
42. Kelley, D.R. *et al.* (2018) Sequential regulatory activity prediction across chromosomes with convolutional neural networks. *Genome Res.* 28, 739–750
43. Washburn, J.D. *et al.* (2019) Evolutionarily informed deep learning methods for predicting relative transcript abundance from DNA sequence. *Proc. Natl. Acad. Sci. U. S. A.* 116, 5542–5549
44. Zuallaert, J. *et al.* (2018) SpliceRover: interpretable convolutional neural networks for improved splice site prediction. *Bioinformatics* 34, 4180–4188
45. Kim, J.-S. *et al.* (2018) RIDDLE: race and ethnicity imputation from disease history with deep learning. *PLoS Comput. Biol.* 14, e1006106
46. Szegedy, C. *et al.* (2016) Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, IEEE
47. Esteva, A. *et al.* (2017) Dermatologist-level classification of skin cancer with deep neural networks. *Nature* 542, 115–118
48. Che, D. *et al.* (2010) Classification of genomic islands using decision trees and their ensemble algorithms. *BMC Genomics* 11, S1
49. Jing, F. *et al.* (2019) An integrative framework for combining sequence and epigenomic data to predict transcription factor binding sites using deep learning. *IEEE/ACM Trans. Comput. Biol. Bioinform.* Published online February 28, 2019. https://doi.org/10.1109/TCBB.2019.2901789
50. Jiang, H. *et al.* (2004) Joint analysis of two microarray gene-expression data sets to select lung adenocarcinoma marker genes. *BMC Bioinforma.* 5, 81
51. Zhou, J. *et al.* (2018) Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk. *Nat. Genet.* 50, 1171–1179
52. Rajaraman, S. *et al.* (2018) Understanding the learned behavior of customized convolutional neural networks toward malaria parasite detection in thin blood smear images. *J. Med. Imaging* 5, 1
53. Wachter, S. *et al.* (2018) Counterfactual explanations without opening the black box: automated decisions and the GDPR. *Harv. J. Law Technol.* 31, 841–887
54. Gritsenko, A.A. *et al.* (2017) Sequence features of viral and human internal ribosome entry sites predictive of their activity. *PLoS Comput. Biol.* 13, e1005734
55. Goldstein, A. *et al.* (2015) Peeking inside the black box: visualizing statistical learning with plots of individual conditional expectation. *J. Comput. Graph. Stat.* 24, 44–65
56. Ghahramani, A. *et al.* (2018) Generative adversarial networks simulate gene expression and predict perturbations in single cells. *bioRxiv.* Published online July 30, 2018. https://doi.org/10.1101/262501
57. Liu, Z. and Yang, J. (2014) Quantifying ecological drivers of ecosystem productivity of the early-successional boreal *Larix gmelinii* forest. *Ecosphere* 5, art84
58. Ribeiro, M.T. *et al.* (2016) "Why should I trust you?": explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pp. 1135–1144
59. Nanayakkara, S. *et al.* (2018) Characterising risk of in-hospital mortality following cardiac arrest using machine learning: a retrospective international registry study. *PLoS Med.* 15, e1002709
60. Eraslan, G. *et al.* (2019) Deep learning: new computational modelling techniques for genomics. *Nat. Rev. Genet.* 20, 389–403
61. Weitschek, E. *et al.* (2018) CamurWeb: a classification software and a large knowledge base for gene expression data of cancer. *BMC Bioinforma.* 19, 354
62. Beykikhoshk, A. *et al.* (2020) DeepTRIAGE: interpretable and individualised biomarker scores using attention mechanism for the classification of breast cancer sub-types. *BMC Med. Genet.* 13, 20
63. Alber, M. *et al.* (2018) iNNvestigate neural networks! *arXiv.* Published online August 13, 2018. http://doi.org/abs/1808.04260
64. Beck, M.W. (2018) NeuralNetTools: visualization and analysis tools for neural networks. *J. Stat. Softw.* 85, 1–20
65. Vidovic, M.M.-C. *et al.* (2015) SVM2Motif—reconstructing overlapping DNA sequence motifs by mimicking an SVM predictor. *PLoS One* 10, e0144782