Machine learning applications in genetics and genomics

Maxwell W. Libbrecht[1] *and William Stafford Noble*[1,2]

Abstract | The field of machine learning, which aims to develop computer algorithms that improve with experience, holds promise to enable computers to assist humans in the analysis of large, complex data sets. Here, we provide an overview of machine learning applications for the analysis of genome sequencing data sets, including the annotation of sequence elements and epigenetic, proteomic or metabolomic data. We present considerations and recurrent challenges in the application of supervised, semi-supervised and unsupervised machine learning methods, as well as of generative and discriminative modelling approaches. We provide general guidelines to assist in the selection of these machine learning methods and their practical application for the analysis of genetic and genomic data sets.

Machine learning
A field concerned with the development and application of computer algorithms that improve with experience.

[1]*Department of Computer Science and Engineering, University of Washington, 185 Stevens Way, Seattle, Washington 98195–2350, USA.*
[2]*Department of Genome Sciences, University of Washington, 3720 15th Ave NE Seattle, Washington 98195–5065, USA.*
Correspondence to W.S.N.
e-mail: william-noble@uw.edu
doi:10.1038/nrg3920
Published online 7 May 2015

The field of machine learning is concerned with the development and application of computer algorithms that improve with experience[1]. Machine learning methods have been applied to a broad range of areas within genetics and genomics. Machine learning is perhaps most useful for the interpretation of large genomic data sets and has been used to annotate a wide variety of genomic sequence elements. For example, machine learning methods can be used to 'learn' how to recognize the locations of transcription start sites (TSSs) in a genome sequence[2]. Algorithms can similarly be trained to identify splice sites[3], promoters[4], enhancers[5] or positioned nucleosomes[6]. In general, if one can compile a list of sequence elements of a given type, then a machine learning method can probably be trained to recognize those elements. Furthermore, models that each recognize an individual type of genomic element can be combined, along with (learned) logic about their relative locations, to build machine learning systems that are capable of annotating genes — including their untranslated regions (UTRs), introns and exons — along entire eukaryotic chromosomes[7].

As well as learning to recognize patterns in DNA sequences, machine learning algorithms can use input data generated by other genomic assays — for example, microarray or RNA sequencing (RNA-seq) expression data; data from chromatin accessibility assays such as DNase I hypersensitive site sequencing (DNase-seq), micrococcal nuclease digestion followed by sequencing (MNase–seq) and formaldehyde-assisted isolation of regulatory elements followed by sequencing (FAIRE–seq); or chromatin immunoprecipitation followed by sequencing (ChIP–seq) data of histone modification or transcription factor binding. Gene expression data can be used to learn to distinguish between different disease phenotypes and, in the process, to identify potentially valuable disease biomarkers. Chromatin data can be used, for example, to annotate the genome in an unsupervised manner, thereby potentially enabling the identification of new classes of functional elements.

Machine learning applications have also been extensively used to assign functional annotations to genes. Such annotations most frequently take the form of Gene Ontology term assignments[8]. Input of predictive algorithms can be any one or more of a wide variety of data types, including the genomic sequence; gene expression profiles across various experimental conditions or phenotypes; protein–protein interaction data; synthetic lethality data; open chromatin data; and ChIP–seq data of histone modification or transcription factor binding. As an alternative to Gene Ontology term prediction, some predictors instead identify co-functional relationships, in which the machine learning method outputs a network in which genes are represented as nodes and an edge between two genes indicates that they have a common function[9].

Finally, a wide variety of machine learning methods have been developed to help to understand the mechanisms underlying gene expression. Some techniques aim to predict the expression of a gene on the basis of

Artificial intelligence
A field concerned with the development of computer algorithms that replicate human skills, including learning, visual perception and natural language understanding.

Heterogeneous data sets
A collection of data sets from multiple sources or experimental methodologies. Artefactual differences between data sets can confound analysis.

Likelihood
The probability of a data set given a particular model.

Label
The target of a prediction task. In classification, the label is discrete (for example, 'expressed' or 'not expressed'); in regression, the label is of real value (for example, a gene expression value).

Examples
Data instances used in a machine learning task.

Supervised learning
Machine learning based on an algorithm that is trained on labelled examples and used to predict the label of unlabelled examples.

the DNA sequence alone[10], whereas others take into account ChIP–seq profiles of histone modification[11] or transcription factor binding[12] at the gene promoter region. More sophisticated methods attempt to jointly model the expression of all of the genes in a cell by training a network model[13]. Like a co-functional network, each node in a gene expression network denotes a gene; however, edges in this case represent, for example, the regulatory relationships between transcription factors and their targets.

Many of the problems listed above can also be solved using techniques from the field of statistics. Indeed, the line between machine learning and statistics is at best blurry, and some prefer the term statistical learning over machine learning[14]. Historically, the field of machine learning grew out of the artificial intelligence community, in which the term 'machine learning' became popular in the late 1990s. In general, machine learning researchers have tended to focus on a subset of problems within statistics, emphasizing in particular the analysis of large heterogeneous data sets. Accordingly, many core statistical concepts — such as the calibration of likelihood estimates, statistical confidence estimations and power calculations — are essentially absent from the machine learning literature.

Here, we provide an overview of machine learning applications in genetics and genomics. We discuss the main categories of machine learning methods and the key considerations that must be made when applying these methods to genomics. We do not attempt to catalogue all machine learning methods or all reported applications of machine learning to genomics, nor do we discuss any particular method in great detail. Instead, we begin by explaining several key distinctions in the main types of machine learning and then outlining some of the major challenges in applying machine learning methods to practical problems in genomics. We provide an overview of the type of research questions for which machine learning approaches are appropriate and advise on how to select the type of methods that are likely to be effective. A more detailed discussion of machine learning applied to particular subfields of genetics and genomics is available elsewhere[15–18].

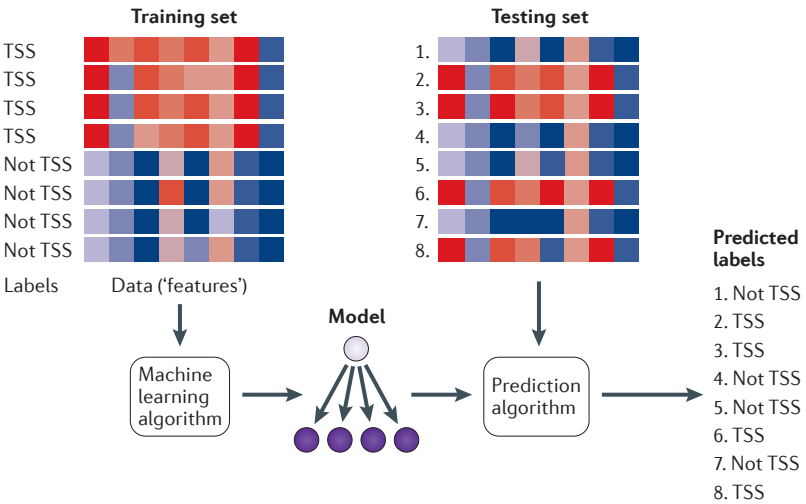

Figure 1 | **A canonical example of a machine learning application.** A training set of DNA sequences is provided as input to a learning procedure, along with binary labels indicating whether each sequence is centred on a transcription start site (TSS) or not. The learning algorithm produces a model that can then be subsequently used, in conjunction with a prediction algorithm, to assign predicted labels (such as 'TSS' or 'not TSS') to unlabelled test sequences. In the figure, the red–blue gradient might represent, for example, the scores of various motif models (one per column) against the DNA sequence.

Stages of machine learning

Machine learning methods proceed through three stages (FIG. 1). As an example, we consider an application to identify the locations of TSSs within a whole-genome sequence[2]. First, a machine learning researcher develops an algorithm that he or she believes will lead to successful learning. Second, the algorithm is provided with a large collection of TSS sequences as well as, optionally, a list of sequences that are known not to be TSSs. The annotation indicating whether a sequence is a TSS is known as the label. The algorithm processes these labelled sequences and stores a model. Third, new unlabelled sequences are given to the algorithm, and it uses the model to predict labels (in this case, 'TSS' or 'not TSS') for each sequence. If the learning was successful, then all or most of the predicted labels will be correct. If the labels associated with test set examples are known — that is, if these examples were excluded from the training set because they were intended to be used to test the performance of the learning system — then the performance of the machine learning algorithm can be assessed immediately. Otherwise, in a prospective validation setting, the TSS predictions produced by the machine learning system must be tested independently in the laboratory. Note that this is an example of a subtype of machine learning called supervised learning, which is described in more detail in the next section. This process of algorithm design, learning and testing is simultaneously analogous to the scientific method on two different levels. First, the design–learn–test process provides a principled way to test a hypothesis about machine learning: for example, algorithm X can successfully learn to recognize TSSs. Second, the algorithm itself can be used to generate hypotheses: for example, sequence Y is a TSS. In the latter setting, the resulting scientific theory is instantiated in the model produced by the learning algorithm. In this case, a key question, which we return to below, is whether and how easily a human can interpret this model.

Supervised versus unsupervised learning

Machine learning methods can usefully be segregated into two primary categories: supervised or unsupervised learning methods. Supervised methods are trained on labelled examples and then used to make predictions about unlabelled examples, whereas unsupervised methods find structure in a data set without using labels. To illustrate the difference, consider again gene-finding algorithms, which use the DNA sequence of a chromosome as input to predict the locations and detailed intron–exon structure of all of

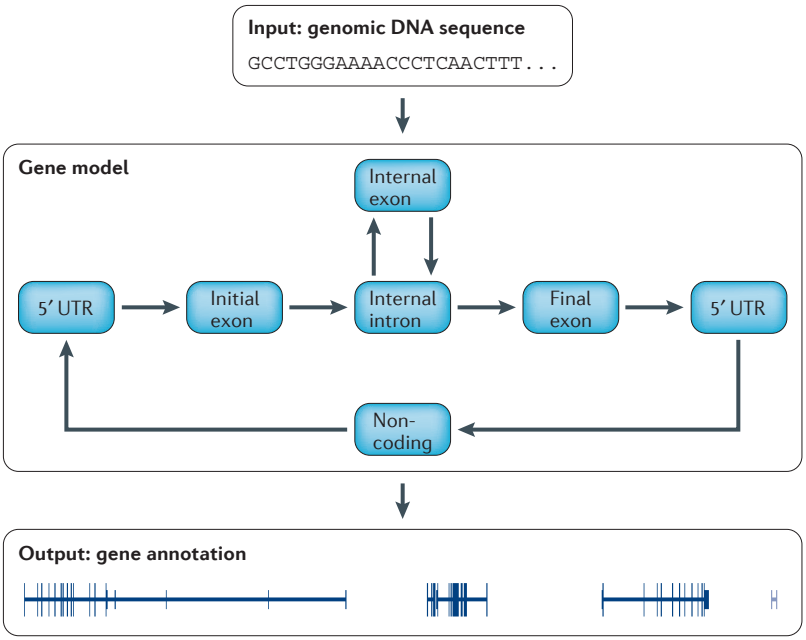

Figure 2 | **A gene-finding model.** A simplified gene-finding model that captures the basic properties of a protein-coding gene is shown. The model takes the DNA sequence of a chromosome, or a portion thereof, as input and produces detailed gene annotations as output. Note that this simplified model is incapable of identifying overlapping genes or multiple isoforms of the same gene. UTR, untranslated region.

the protein-coding genes on the chromosome (FIG. 2). The most straightforward way to do so is to use what is already known about the genome to help to build a predictive model. In particular, a supervised learning algorithm for gene finding requires as input a training set of labelled DNA sequences specifying the locations of the start and end of the gene (that is, the TSS and the transcription termination site, respectively), as well as all of the splice sites in between these sites. The model then uses this training data to learn the general properties of genes, such as the DNA sequence patterns that typically occur near a donor or an acceptor splice site; the fact that in-frame stop codons should not occur within coding exons; and the expected length distributions of 5′ and 3′ UTRs and of initial, internal and final introns. The trained model can then use these learned properties to identify additional genes that resemble the genes in the training set.

When a labelled training set is not available, unsupervised learning is required. For example, consider the interpretation of a heterogeneous collection of epigenomic data sets, such as those generated by the Encyclopedia of DNA Elements (ENCODE) Consortium and the Roadmap Epigenomics Project. A priori, we expect that the patterns of chromatin accessibility, histone modifications and transcription factor binding along the genome should be able to provide a detailed picture of the biochemical and functional activity of the genome. We may also expect that these activities could be accurately summarized using a fairly small set of labels. If we are interested in discovering what types of label best explain the data, rather than imposing a pre-determined set of labels on the data, then we must use unsupervised rather than supervised learning. In this type of approach, the machine learning algorithm uses only the unlabelled data and the desired number of different labels to assign as input[19–21]; it then automatically partitions the genome into segments and assigns a label to each segment, with the goal of assigning the same label to segments that have similar data. The unsupervised approach requires an additional step in which semantics must be manually assigned to each label, but it provides the benefits of enabling training when labelled examples are unavailable and has the ability to identify potentially novel types of genomic elements.

Semi-supervised learning. The intermediate between supervised and unsupervised learning is semi-supervised learning[22]. In supervised learning, the algorithm receives as input a collection of data points, each with an associated label, whereas in unsupervised learning the algorithm receives the data but no labels. The semi-supervised setting is a mixture of these two approaches: the algorithm receives a collection of data points, but only a subset of these data points have associated labels. In practice, gene-finding systems are often trained using a semi-supervised approach, in which the input is a collection of annotated genes and an unlabelled whole-genome sequence. The learning procedure begins by constructing an initial gene-finding model on the basis of the labelled subset of the training data alone. Next, the model is used to scan the genome, and tentative labels are assigned throughout the genome. These tentative labels can then be used to improve the learned model, and the procedure iterates until no new genes are found. The semi-supervised approach can work much better than a fully supervised approach because the model is able to learn from a much larger set of genes — all of the genes in the genome — rather than only the subset of genes that have been identified with high confidence.

Which type of method to use. When faced with a new machine learning task, the first question to consider is often whether to use a supervised, unsupervised or semi-supervised approach. In some cases, the choice is limited; for example, if no labels are available, then only unsupervised learning is possible. However, when labels are available, a supervised approach is not always the best choice because every supervised learning method rests on the implicit assumption that the distribution responsible for generating the training data set is the same as the distribution responsible for generating the test data set. This assumption will be respected if, for example, one takes a single labelled data set and randomly subdivides it into a training set and a testing set. However, it is often the case that the plan is to train an algorithm on a training set that is generated differently from the testing data to which the trained model will eventually be applied. A gene finder trained using a set of human genes will probably not perform very well at finding genes in the mouse genome. Often, the divergence between training and testing is less

Unsupervised learning
Machine learning based on an algorithm that does not require labels, such as a clustering algorithm.

Semi-supervised learning
A machine-learning method that requires labels but that also makes use of unlabelled examples.

obvious. For example, a TSS data set generated by cap analysis of gene expression (CAGE) will not contain non-polyadenylated genes[23]. If such genes also exhibit differences around the TSSs, then the resulting TSS predictor will be biased. In general, supervised learning should be used only when the training set and test set are expected to exhibit similar statistical properties.

When supervised learning is feasible and additional unlabelled data points are easy to obtain, one may consider whether to use a supervised or semi-supervised approach. Semi-supervised learning requires making certain assumptions about the data set[22] and, in practice, assessing these assumptions can often be very difficult. Therefore, a good rule of thumb is to use semi-supervised learning only when there are a small amount of labelled data and a large amount of unlabelled data.

Generative versus discriminative modelling

Applications of machine learning methods generally have one of two goals: prediction or interpretation. Consider the problem of predicting, on the basis of a ChIP–seq experiment, the locations at which a given transcription factor will bind to genomic DNA. This task is analogous to the TSS prediction task (FIG. 1), except that the labels are derived from ChIP–seq peaks. A researcher applying a machine learning method to this problem may either want to understand what properties of a sequence are the most important for determining whether a transcription factor will bind (that is, interpretation), or simply want to predict the locations of transcription factor binding as accurately as possible (that is, prediction). There are trade-offs between accomplishing these two goals — methods that optimize prediction accuracy often do so at the cost of interpretability.

The distinction between generative models and discriminative models plays a large part in the trade-off between interpretability and performance. The generative approach builds a full model of the distribution of features in each of the two classes and then compares how the two distributions differ from one another. By contrast, the discriminative approach focuses on accurately modelling only the boundary between the two classes. From a probabilistic perspective, the discriminative approach involves modelling only the conditional distribution of the label given the input feature data sets, as opposed to the joint distribution of the labels and features. Schematically, if we were to separate two groups of points in a 2D space (FIG. 3a), the generative approach builds a full model of each class, whereas the discriminative approach focuses only on separating the two classes.

A researcher applying a generative approach to the transcription factor binding problem begins by considering what procedure could be used to generate the observed data. A widely used generative model of transcription factor binding uses a position-specific frequency matrix (PSFM) (FIG. 3b), in which a collection of aligned binding sites of width w are summarized in a $4 \times w$ matrix (M) of frequencies, where the entry at position i,j represents the empirical frequency of observing the i^{th} DNA base at position j. We can generate a random bound sequence according to this PSFM model by drawing w random numbers, each in the range $[0,1\rangle$. For the j^{th} random number, we select the corresponding DNA base according to the frequencies in the j^{th} column of the matrix. Conversely, scoring a candidate binding site using the model corresponds to computing the product of the corresponding frequencies from the PSFM. This value is called the likelihood. Training a PSFM is simple — the empirical frequency of each nucleotide at each position simply needs to be computed.

A simple example of a discriminative algorithm is the support vector machine (SVM)[24,25] (FIG. 3c), the goal of which is to learn to output a value of 1 whenever it is given a positive training example and a value of –1 whenever it is given a negative training example. In the transcription factor binding prediction problem, the input sequence of length w is encoded as a binary string of length $4w$, and each bit corresponds to the presence or absence of a particular nucleotide at a particular position.

This generative modelling approach offers several compelling benefits. First, the generative description of the data implies that the model parameters have well-defined semantics relative to the generative process. Accordingly, as shown in the example above, the model not only predicts the locations to which a given transcription factor binds but also explains why the transcription factor binds there. If we compare two different potential binding sites, we can see that the model prefers one site over another and also that the reason is, for example, the preference for an adenine rather than a thymine at position 7 of the motif. Second, generative models are frequently stated in terms of probabilities, and the probabilistic framework provides a principled way to handle problems like missing data. For example, it is still possible for a PSFM to make a prediction for a binding site where one or more of the bound residues is unknown. This is accomplished by probabilistically averaging over the missing bases. The output of the probabilistic framework has well-defined, probabilistic semantics, and this can be helpful when making downstream decisions about how much to trust a given prediction.

In many cases, including the example of transcription factor binding, the training data set contains a mixture of positive and negative examples. In a generative setting, these two groups of examples are modelled separately, and each has its own generative process. For instance, for the PSFM model, the negative (or background) model is often a single set (B) of nucleotide frequencies that represents the overall mean frequency of each nucleotide in the negative training examples. To generate a sequence of length w according to this model, we again generate w random numbers, but now each base is selected according to the frequencies in B. To use the foreground PSFM model together with the background model B, we compute a likelihood ratio that is simply the ratio of the likelihoods computed with respect to the PSFM and with respect to B.

Prediction accuracy
The fraction of predictions that are correct. It is calculated by dividing the number of correct predictions by the total number of predictions.

Generative models
Machine learning models that build a full model of the distribution of features.

Discriminative models
Machine learning approaches that model only the distribution of a label when given the features.

Features
Single measurements or descriptors of examples used in a machine learning task.

Probabilistic framework
A machine learning approach based on a probability distribution over the labels and features.

Missing data
An experimental condition in which some features are available for some, but not all, examples.

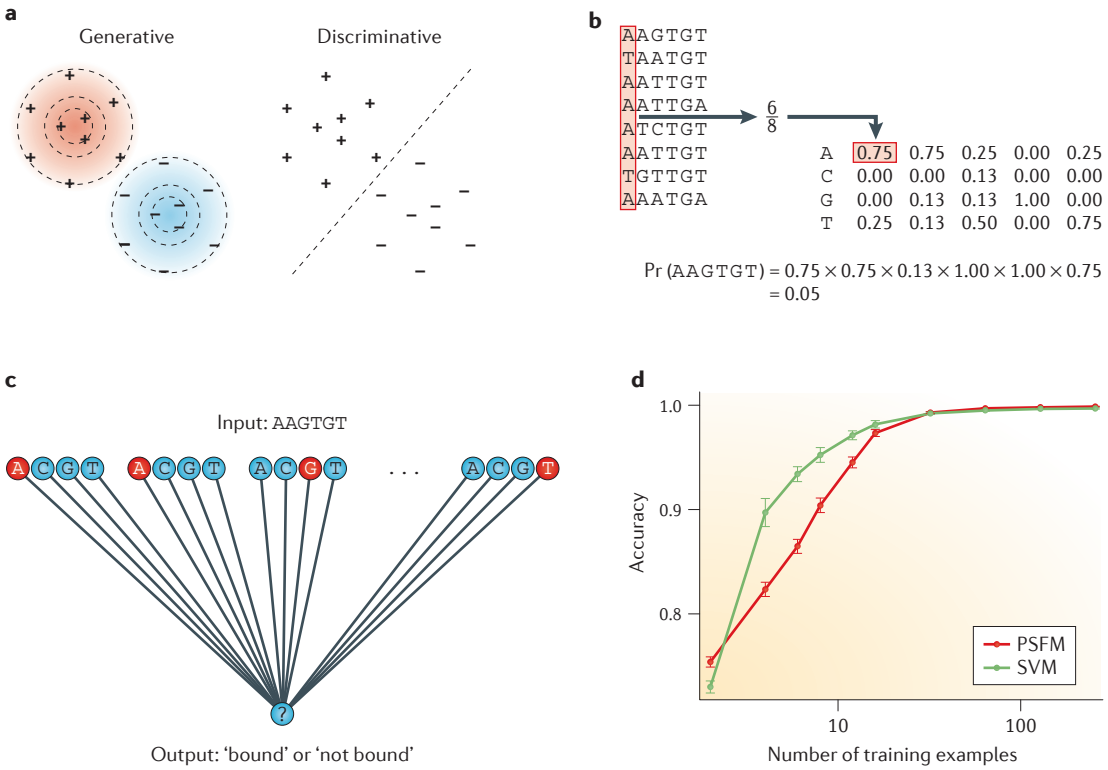

Figure 3 | **Two models of transcription factor binding.** **a** | Generative and discriminative models are different in their interpretability and prediction accuracy. If we were to separate two groups of points, the generative model characterizes both classes completely, whereas the discriminative model focuses on the boundary between the classes. **b** | In a position-specific frequency matrix (PSFM) model, the entry in row *i* and column *j* represents the frequency of the i^{th} base occurring at position *j* in the training set. Assuming independence, the probability of the entire sequence is the product of the probabilities associated with each base. **c** | In a linear support vector machine (SVM) model of transcription factor binding, labelled positive and negative training examples (red and blue, respectively) are provided as input, and a learning procedure adjusts the weights on the edges to predict the given label. **d** | The graph plots the mean accuracy (±95% confidence intervals) of PSFM and SVM, on a set of 500 simulated test sets, of predicting transcription factor binding as a function of the number of training examples.

The primary benefit of the discriminative modelling approach is that it probably achieves better performance than the generative modelling approach with infinite training data[26,27]. In practice, analogous generative and discriminative approaches often converge to the same solution, and generative approaches can sometimes perform better with limited training data. However, when the amount of labelled training data is reasonably large, the discriminative approach will tend to find a better solution, in the sense that it will predict the desired outcome more accurately when tested on previously unseen data (assuming that the data are from the same underlying distribution as the training data). To illustrate this phenomenon, we simulated data according to a simple PSFM model and trained a PSFM and an SVM, varying the number of training examples. To train a model with a width of 19 nucleotides to discriminate between bound and unbound sites with 90% accuracy requires 8 training examples for a PSFM model and only 4 examples for an SVM model (FIG. 3d). This improvement in performance is achieved because, by not attempting to accurately characterize the simpler parts of the 2D space, the discriminative model performs better at solving the discrimination task at hand. Thus, empirically, the discriminative approach will tend to give predictions that are more accurate.

However, the flipside of this accuracy is that by solving a single problem well, the discriminative approach fails to solve other problems at all. Specifically, because the internal parameters of a generatively trained model have well-defined semantics, we can use the model to ask various related questions, for example, not only whether CCCTC-binding factor (CTCF) bind to a particular sequence but also why does it bind to this sequence more tightly than to some other sequence. By contrast, the discriminative model only enables us to answer the single question for which it was designed. Thus, choosing between a generative and discriminative model involves a trade-off between predictive accuracy and interpretability of the model. Although the distinction between generative and discriminative models plays a large part in determining the interpretability of a model, the model's complexity — that is, the number of parameters it has — can be just as important. Models of either type that have a large number

Feature selection
The process of choosing a smaller set of features from a larger set, either before applying a machine learning method or as part of training.

Input space
A set of features chosen to be used as input for a machine learning method.

Uniform prior
A prior distribution for a Bayesian model that assigns equal probabilities to all models.

of parameters tend to be difficult to interpret but can generally achieve higher accuracy than models with few parameters if they are provided with enough data. The complexity of a model can be limited either by choosing a simple model or by using a feature selection strategy to restrict the complexity of the learned model.

Incorporating prior knowledge

In many applications, the success or failure of a machine learning method depends on the extent to which the method accurately encodes various types of prior knowledge about the problem at hand. Indeed, much of the practical application of machine learning involves understanding the details of a particular problem and then selecting an algorithmic approach that enables those details to be accurately encoded. There is no optimal machine learning algorithm that works best for all problems[28], so the selection of an approach that matches the researcher's prior knowledge about the problem is crucial to the success of the analysis.

Often, the encoding of prior knowledge is implicit in the framing of the machine learning problem. As an example, consider the prediction of nucleosome positioning from primary DNA sequence[6]. The labels for this prediction task can be derived, for example, from an MNase–seq assay. In this case, after appropriate processing, each base is assigned an integer count of the number of nucleosome-sized fragments that cover the base. Therefore, it might seem natural to frame the problem as a regression, in which the input is, for example, a sequence of 201 bp and the output is the predicted coverage at the centre of the sequence. However, in practice, we may be particularly interested in identifying nucleosome-free regions. Hence, rather than asking the algorithm to solve the problem of predicting exact coverage at each base, we might instead opt to predict whether each base occurs in a nucleosome-free region. Switching from regression to classification makes the problem easier but, more importantly, this switch also encodes the prior knowledge that regions of high MNase accessibility are of particular biological interest.

Implicit prior knowledge. In other cases, prior knowledge is encoded implicitly in the choice of data that we provide as input to the machine learning algorithm. For example, Yip *et al.*[29] trained a collection of machine learning methods to distinguish among various types of genomic elements. Using chromatin data — DNase I accessibility and ChIP–seq profiles of histone modifications and transcription factor binding — one classifier distinguished regulatory regions that are close to a gene from regulatory regions that are far away from any gene. In this context, design of the input space (that is, the features that are provided as input to the classifier) is of crucial importance. In the work carried out by Yip *et al.*[29], the features were computed by averaging over a 100-bp window. The choice of a 100-bp window is likely to reflect the prior knowledge that, at least for histone modification data, the data are arguably only meaningful at approximately the scale of a single nucleosome (147 bp) or larger. Moreover, replacing a single averaged feature with 100 separate features may be problematic. By contrast, for DNase accessibility data, it is plausible that averaging over 100 bp may remove some useful signal. Alternatively, prior knowledge may be implicitly encoded in the learning algorithm itself, in which some types of solutions are preferred over others[30]. Therefore, in general, the choice of input data sets, their representations and any pre-processing must be guided by prior knowledge about data and application.

Probabilistic priors. In a probabilistic framework, some forms of prior knowledge can be represented explicitly by specifying a prior distribution over the data. A common prior distribution is the uniform prior (also known as 'uninformative' prior) which, despite the name, can be useful in some contexts. Consider, for example, a scenario in which we have gathered a collection of ten validated binding sites for a particular transcription factor (FIG. 4), and we observe that the sequences cluster around a clear consensus motif. If we represent these sequences using a pure PSFM then, because our data set is fairly small, a substantial number of the entries in the PSFM will be zero. Consequently, the model will assign any sequence that contains one of these zero entries an overall probability of zero, even if the sequence otherwise exactly matches the motif. This is counter-intuitive. The solution to this problem is to encode the prior knowledge that every possible DNA sequence has the potential to be bound by a given transcription factor. The result is that even sequences containing nucleotides that we

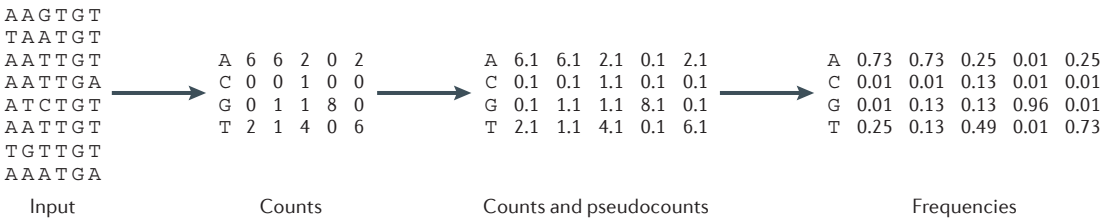

Figure 4 | **Incorporating a probabilistic prior into a position-specific frequency matrix.** A simple, principled method for putting a probabilistic prior on a position-specific frequency matrix involves augmenting the observed nucleotide counts with pseudocounts and then computing frequencies with respect to the sum. The magnitude of the pseudocount corresponds to the weight assigned to the prior.

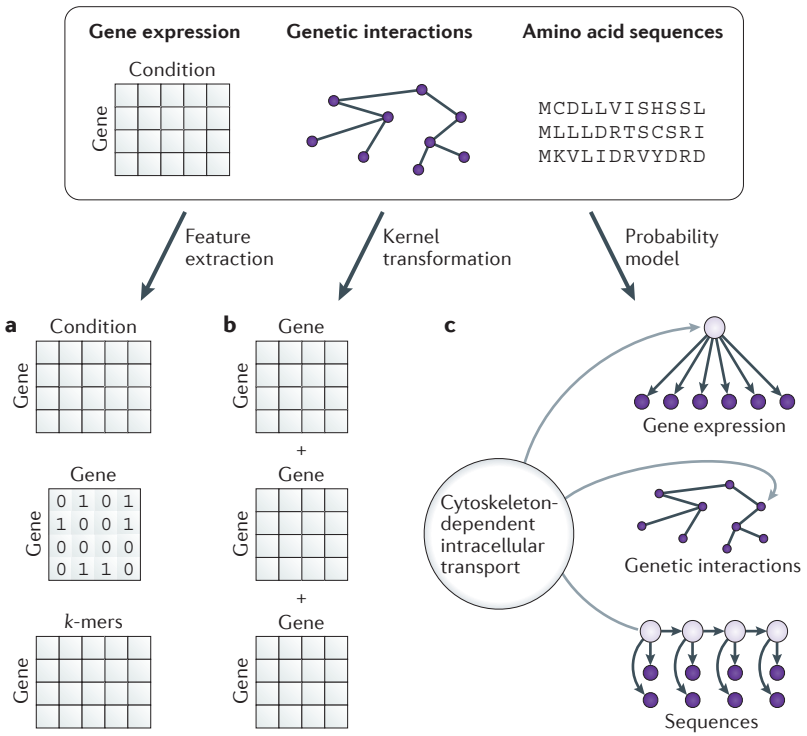

Figure 5 | **Three ways to accommodate heterogeneous data in machine learning.** The task of predicting gene function labels requires methods that take as input data such as gene expression profiles, genetic interaction networks and amino acid sequences. These diverse data types can be encoded into fixed-length features, represented using pairwise similarities (that is, kernels) or directly accommodated by a probability model.

have never observed in a given position can still be assigned a non-zero probability by the model.

In many probability models, much more sophisticated priors have been developed to capture more-complex prior knowledge. A particularly successful example is the use of Dirichlet mixture priors in protein modelling[31]. In this example, the idea is that if we are examining an alignment of protein sequences and if we see many aligned leucines in a particular column, then, because we know that leucine and valine are biochemically similar, we may want to assign a high probability to sequences that contain a valine in that same column, even if we have never seen a valine there in our training data. The name Dirichlet mixture refers to the fact that the prior distribution is represented as a Dirichlet distribution, and that the distribution is a mixture in which each component corresponds to a group of biochemically similar amino acids. Such priors lead to substantially improved performance both in the modelling of evolutionarily related families of proteins[31] and in the discovery of protein motifs[32].

Prior information in non-probabilistic models. By contrast, the incorporation of prior knowledge into non-probabilistic methods can be more challenging. For example, discriminative classifiers, such as artificial neural networks (ANNs) or random forests, do not provide any explicit mechanism for representing prior knowledge. The topology of a multilayer ANN can represent information about dependencies between input features in the input space, but more general priors cannot be represented.

One class of discriminative methods does provide a more general mechanism for representing prior knowledge, if that knowledge can be encoded into a generalized notion of similarity. Kernel methods are algorithms that use a general class of mathematical functions called kernels in place of a simple similarity function (specifically, the cosine of the angle between two input vectors)[33]. The 'flagship' kernel method is the SVM classifier, which has been widely used in many fields, including biological applications ranging from DNA and protein sequence classification to mass spectrometry analysis[25]. Other methods that can use kernels include support vector regression as well as classical algorithms such as *k*-means clustering, principal component analysis and hierarchical clustering. Prior knowledge can be provided to a kernel method by selecting or designing an appropriate kernel function. For example, a wide variety of kernels can be defined between pairs of DNA sequences, in which the similarity can be based on shared *k*-mers irrespective of their positions within the sequences[34], on nucleotides occurring at particular positions[35] or on a mixture of the two[36]. A DNA sequence can even encode a simple model of molecular evolution, using algorithms that are similar to the Smith–Waterman alignment algorithm[37]. Furthermore, the Fisher kernel provides a general framework for deriving a kernel function from any probability model[38]. In this way, formal probabilistic priors can be used in conjunction with any kernel method. Kernel methods have a rich literature, which is reviewed in more detail in REF. 39.

Handling heterogeneous data

Another common challenge in learning from real biological data is that the data themselves are heterogeneous. For example, consider the problem of learning to assign Gene Ontology terms to genes. For a given term, such as 'cytoskeleton-dependent intracellular transport', a wide variety of data types might be relevant, including the amino acid sequence of the protein encoded by the gene; the inferred evolutionary relationships of that protein to other proteins across various species; the microarray or RNA-seq expression profile of the gene across a variety of phenotypic or environmental conditions; and the number and identity of neighbouring proteins identified using yeast two-hybrid or tandem affinity purification tagging experiments, or GFP-tagged microscopy images (FIG. 5). Such data sets are difficult to analyse jointly because of their heterogeneity: an expression profile is a fixed-length vector of real values; protein–protein interaction information is a binary network; and protein sequences are of variable lengths and are made up of a discrete 'alphabet'. Many statistical and machine learning methods for classification assume that all of the data can be represented as fixed-length vectors of real numbers. Such methods cannot be directly applied to heterogeneous data.

Dirichlet mixture priors
Prior distributions for a Bayesian model over the relative frequencies of, for example, amino acids.

Kernel methods
A class of machine learning methods (for example, support vector machine) that use a type of similarity measure (called a kernel) between feature vectors.

The most straightforward way to solve this problem is to transform each type of data into vector format before processing (FIG. 5a). For example, this was the approach taken by Peña-Castillo *et al.*[40] in an assessment of methods for predicting gene function in mice. Participating research groups were provided with separate matrices, each representing a single type of data: gene expression, sequence patterns, protein–protein interactions, phenotypes, conservation profiles or disease associations. All matrices shared a common set of rows (one per gene), but the number and meanings of the columns differed from one matrix to the next. For example, gene expression data were represented directly, whereas protein sequence data were represented indirectly through annotations from repositories such as Pfam[41] and InterPro[42], and protein–protein interactions were represented as square matrices in which entries were the shortest-path lengths between genes.

Alternatively, each type of data can be encoded using a kernel function (FIG. 5b), with one kernel for each data type. Mathematically, the use of kernels is formally equivalent to transforming each data type into a fixed-length vector; however, in the kernel approach the vectors themselves are not always represented explicitly. Instead, the similarity between two data elements — such as amino acid sequences or nodes in a protein–protein interaction network — is encoded in the kernel function. The kernel framework enables more-complex kernels to be defined from combinations of simple kernels. For example, a simple summation of all the kernels for a given pair of genes is itself a kernel function. Furthermore, the kernels themselves can encode prior knowledge by, for example, allowing for statistical dependencies within a given data type but not between data types[43]. Kernels also provide a general framework for automatically learning the relative importance of each type of data relative to a given classification task[44].

Finally, probability models provide a very different method for handling heterogeneous data. Rather than forcing all the data into a vector representation or requiring that all data be represented using pairwise similarities, a probability model explicitly represents diverse data types in the model itself (FIG. 5c). An early example of this type of approach assigned gene functional labels to yeast genes on the basis of gene expression profiles; physical associations from affinity purification, two-hybrid and direct binding measurements; and genetic associations from synthetic lethality experiments[45]. The authors used a Bayesian network, which is a formal graphical language for representing the joint probability distribution over a set of random variables[46]. Querying the network with respect to the variable representing a particular Gene Ontology label yields the probability that a given gene is assigned that label. In principle, the probabilistic framework enables a Bayesian network to represent any arbitrary data type and carry out joint inference over heterogeneous data types using a single model.

In practice, such inference can be challenging because a joint probability model over heterogeneous data may contain a very large number of trainable parameters. Therefore, an alternative method for handling heterogeneous data in a probability model is to make use of the general probabilistic mechanism for handling prior knowledge by treating one type of data before another. For example, as discussed above, predicting the location on a genome sequence to which a particular transcription factor will bind can be framed as a classification problem and solved using a PSFM. However, *in vivo*, the binding of a transcription factor depends not only on the native affinity of the transcription factor for a given DNA sequence but also on the competitive binding of other molecules. Accordingly, measurements of chromatin accessibility, as provided by assays such as DNase-seq[47], can offer valuable information about the overall accessibility of a given genomic locus to transcription factor binding. A joint probability model can take this accessibility into account[48,49], but training such a model can be challenging. The alternative approach uses the DNase-seq data to create a probabilistic prior and applies this prior during the scanning of the PSFM[50].

Feature selection

In any application of machine learning methods, the researcher must decide what data to provide as input to the algorithm. As noted above, this choice provides a method for incorporating prior knowledge into the procedure because the researcher can decide which features of the data are likely to be relevant or irrelevant. For example, consider the problem of training a multiclass classifier to distinguish, on the basis of gene expression measurements, among different types of cancers[51]. Such a classifier could be valuable in two ways. First, the classifier itself could help to establish accurate diagnoses in cases of atypical presentation or histopathology. Second, the model produced during the learning phase could perform feature selection, thus identifying subsets of genes with expression patterns that contribute specifically to different types of cancer. In general, feature selection can be carried out within any supervised learning algorithm, in which the algorithm is given a large set of features (or input variables) and then automatically makes a decision to ignore some or all of the features, focusing on the subset of features that are most relevant to the task at hand.

In practice, it is important to distinguish among three distinct motivations for carrying out feature selection. First, in some cases, we want to identify a very small set of features that yield the best possible classifier. For example, we may want to produce an inexpensive way to identify a disease phenotype on the basis of the measured expression levels of a handful of genes. Such a classifier, if it is accurate enough, might form the basis of an inexpensive clinical assay. Second, we may want to use the classifier to understand the underlying biology[52–54]. In this case, we want the feature selection procedure to identify only the genes with expression levels that are actually relevant to the task at hand in the hope that the corresponding functional annotations or biological pathways might provide

Bayesian network
A representation of a probability distribution that specifies the structure of dependencies between variables as a network.

insights into the aetiology of disease. Third, we often simply want to train the most accurate possible classifier[55]. In this case, we hope that the feature selection enables the classifier to identify and eliminate noisy or redundant features. Researchers are often disappointed to find that feature selection cannot optimally perform more than one of these three tasks simultaneously.

Feature selection is especially important in the third case because the analysis of high-dimensional data sets, including genomic, epigenomic, proteomic or metabolomic data sets, suffers from the curse of dimensionality[56] — the general observation that many types of analysis become more difficult as the number of input dimensions (that is, data measurements) grows very large. For example, as the number of data features that are provided as input to a machine learning classifier grows, it is increasingly likely that, by chance, one feature perfectly separates the training examples into positive and negative classes. This phenomenon leads to good performance on the training data but poor generalization to data that were not used in training owing to overfitting of the model to the training data. Feature selection methods and dimensionality reduction techniques, such as principal component analysis or multidimensional scaling, aim to solve this problem by projecting the data from higher to lower dimensions.

Imbalanced class sizes

A common stumbling block in many applications of machine learning to genomics is the large imbalance (or label skew) in the relative sizes of the groups being classified. For example, suppose one is trying to use a discriminative machine learning method to predict the locations of enhancers in the genome. Starting with a set of 641 known enhancers, the genome can be broken up into 1,000-bp segments and each segment assigned a label ('enhancer' or 'not enhancer') on the basis of whether it overlaps with a known enhancer. This procedure produces 1,711 positive examples and around 3,000,000 negative examples — 2,000 times as many negative examples as positive examples. Unfortunately, most software cannot handle 3,000,000 examples.

The most straightforward solution to this problem is to select a random, smaller subset of the data. However, in the case of enhancer prediction, selecting 50,000 examples at random results in 49,971 negative examples and only 28 positive examples. This number of positive examples is far too small to train an accurate classifier. To demonstrate this problem, we simulated 93 noisy ChIP–seq assays using a Gaussian model for enhancers and background positions based on data produced by the ENCODE Consortium[57]. We trained a logistic regression classifier to distinguish between enhancers and non-enhancers on the basis of these data. The overall accuracy of the predictions (that is, the percentage of predictions that were correct) was 99.9%. Although this seems good, accuracy is not an appropriate measure with which to evaluate performance in this setting because a null classifier that simply predicts everything to be non-enhancers achieves nearly the same accuracy.

In this context, it is more appropriate to separately evaluate sensitivity (that is, the fraction of enhancers detected) and precision (that is, the percentage of predicted enhancers that are truly enhancers). The balanced classifier described above has a high precision (>99.9%) but a very low sensitivity of 0.5%. The behaviour of the classifier can be improved by using all of the enhancers for training and then picking a random set of 49,000 non-enhancer positions as negative training examples. However, balancing the classes in this way results in the classifier learning to reproduce this artificially balanced ratio. The resulting classifier achieves much higher sensitivity (81%) but very poor precision (40%); thus, this classifier is not useful for finding enhancers that can be validated experimentally.

It is possible to trade off sensitivity and precision while retaining the training power of a balanced training set by placing weights on the training examples. In the case of enhancer prediction, we used the balanced training set, but during training we weighted each negative example 36 times more than a positive example. Doing so results in an excellent sensitivity of 53% with a precision of 95%.

In general, the most appropriate performance measure depends on the intended application of the classifier. For problems such as identifying which tissue a given cell comes from, it may be equally important to identify rare and abundant tissues, and so the overall number of correct predictions may be the most informative measure of performance. In other problems, such as enhancer detection, predictions in one class may be more important than predictions in another. For example, if positive predictions will be published, the most appropriate measure may be the sensitivity among a set of predictions with a predetermined precision (for example, 95%). A wide variety of performance measures are used in practice, including the F_1 measure, the receiver operating characteristic curve and the precision-recall curve[58,59], among others[60]. Machine learning classifiers perform best when they are optimized for a realistic performance measure.

Handling missing data

Machine learning analysis can often be complicated by missing data values. Missing values can come from various sources, such as defective cells in a gene expression microarray, 'unmappable' genome positions in a functional genomic assay or measurements that are unreliable because they saturate the detection limits of an instrument. Missing data values can be divided into two types: values that are missing at random or for reasons that are unrelated to the task at hand (such as defective microarray cells), and values that, when absent, provide information about the task at hand (such as saturated detectors). The presence or absence of values of the latter type is usually best incorporated directly into the model.

The simplest way to deal with data that are missing at random is to impute the missing values[61]. This can be done either with a very simple strategy, such as replacing all of the missing values with zero, or with a

Curse of dimensionality
The observation that analysis can sometimes become more difficult as the number of features increases, particularly because overfitting becomes more likely.

Overfitting
A common pitfall in machine learning analysis that occurs when a complex model is trained on too few data points and becomes specific to the training data, resulting in poor performance on other data.

Label skew
A phenomenon in which two labels in a supervised learning problem are present at different frequencies.

Sensitivity
(Also known as recall). The fraction of positive examples identified; it is given by the number of positive predictions that are correct divided by the total number of positive examples.

Precision
The fraction of positive predictions that are correct; it is given by the number of positive predictions that are correct divided by the total number of positive predictions.

Precision-recall curve
For a binary classifier applied to a given data set, a curve that plots precision (*y* axis) versus recall (*x* axis) for a variety of classification thresholds.

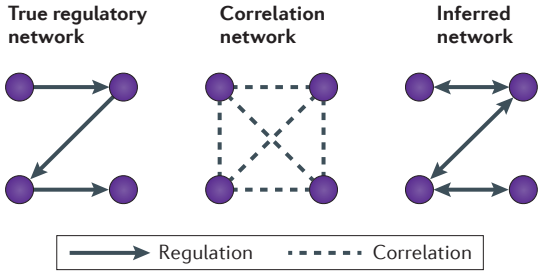

Figure 6 | **Inferring network structure.** Methods that infer each relationship in a network separately, such as by computing the correlation between each pair, can be confounded by indirect relationships. Methods that infer the network as a whole can identify only direct relationships. Inferring the direction of causality inherent in networks is generally more challenging than inferring the network structure[68]; as a result, many network inference methods, such as Gaussian graphical model learning, infer only the network.

more sophisticated strategy. For example, Troyanskaya *et al.*[62] used the correlations between data values to impute missing microarray values. For each target gene expression profile, the authors found the 10 expression profiles showing the greatest similarity to the target profile, and they replaced each missing value in the target profile with an average of the values in the similar profiles. Imputing data points this way can be used as a pre-processing step for any other analysis, but downstream analyses are 'blind' to this information and cannot make use of either the fact that a data point is missing or the added uncertainty that results from missing data values.

Another method for dealing with missing data is to include in the model information about the 'missingness' of each data point. For example, Kircher *et al.*[63] aimed to predict the deleteriousness of mutations based on functional genomic data. The functional genomic data provided a feature vector associated with each mutation, but some of these features were missing. Therefore, for each feature, the authors added a Boolean feature that indicated whether the corresponding feature value was present. The missing values themselves were then replaced with zeroes. A sufficiently sophisticated model will be able to learn the pattern that determines the relationship between the feature and the presence or absence indicator. An advantage of this approach to handling missing data is that it is applicable regardless of whether the absence of a data point is significant — if it is not, the model will learn to ignore the absence indicator.

Finally, probability models can explicitly model missing data by considering all the potential missing values. For example, this approach was used by Hoffman *et al.*[21] to analyse functional genomic data, which contain missing values due to 'mappability' issues from short-read sequencing data. The probability model provides an annotation by assigning a label to each position across the genome and modelling the probability of observing a certain value given this label. Missing data points are handled by summing over all possibilities for that random variable in the model. This approach, called marginalization, represents the case in which a particular variable is unobserved. However, marginalization is only appropriate when data points are missing for reasons that are unrelated to the task at hand. When the presence or absence of a data point is likely to be correlated with the values themselves, incorporating presence or absence explicitly into the model is more appropriate.

Modelling dependence among examples

So far, we have focused on machine learning tasks that involve sets of independent instances of a common pattern. However, in some domains, individual entities, such as genes, are not independent, and the relationships among them are important. By inferring such relationships, it is possible to integrate many examples into a meaningful network. Such a network might represent physical interactions among proteins, regulatory interactions between genes or symbiosis between microorganisms. Networks are useful both for understanding the biological relationships between entities and as input into a downstream analysis that makes use of these relationships.

The most straightforward way to infer the relationships among examples is to consider each pair independently. In this case, the problem of network learning is reduced to a normal machine learning problem, defined on pairs of individuals rather than individual examples. Qiu *et al.*[64] used an SVM classifier to predict, using data such as protein sequences and cellular localization, whether a given pair of proteins physically interact.

A downside of any approach that considers each relationship independently is that such methods cannot take into account the confounding effects of indirect relationships (FIG. 6). For example, in the case of gene regulation, an independent model cannot infer whether a pair of genes directly regulate each other or whether they are both regulated by a third gene. Such spurious inferred relationships, called transitive relationships, can be removed by methods that infer the graph as a whole. For example, Friedman *et al.*[65] inferred a Bayesian network on gene expression data that models which genes regulate each other. Such a network includes only direct effects and models indirect correlations through multiple-hop paths in the network. Therefore, methods that infer a network as a whole are more biologically interpretable because they remove these indirect correlations; a large number of such methods have been described and reviewed elsewhere[66,67].

Conclusions

On the one hand, machine learning methods, which are most effective in the analysis of large, complex data sets, are likely to become ever more important to genomics as more large data sets become available through international collaborative projects, such as the 1000 Genomes Project, the 100,000 Genomes Project, ENCODE, the Roadmap Epigenomics Project and the US National

Marginalization
A method for handling missing data points by summing over all possibilities for that random variable in the model.

Transitive relationships
An observed correlation between two features that is caused by direct relationships between these two features and a third feature.

Institutes of Health's 4D Nucleome Initiative. On the other hand, even in the presence of massive amounts of data, machine learning techniques are not generally useful when applied in an arbitrary manner. In practice, achieving good performance from a machine learning method usually requires theoretical and practical knowledge of both machine learning methodology and the particular research application area. As new technologies for generating large genomic and proteomic data sets emerge — pushing beyond DNA sequencing to mass spectrometry, flow cytometry and high-resolution imaging methods — demand will increase not only for new machine learning methods but also for experts that are capable of applying and adapting them to big data sets. In this sense, both machine learning itself and scientists proficient in these applications are likely to become increasingly important to advancing genetics and genomics.

1. Mitchell, T. *Machine Learning* (McGraw-Hill, 1997). **This book provides a general introduction to machine learning that is suitable for undergraduate or graduate students.**
2. Ohler, W., Liao, C., Niemann, H. & Rubin, G. M. Computational analysis of core promoters in the *Drosophila* genome. *Genome Biol.* **3**, RESEARCH0087 (2002).
3. Degroeve, S., Baets, B. D., de Peer, Y. V. & Rouzé, P. Feature subset selection for splice site prediction. *Bioinformatics* **18**, S75–S83 (2002).
4. Bucher, P. Weight matrix description of four eukaryotic RNA polymerase II promoter elements derived from 502 unrelated promoter sequences. *J. Mol. Biol.* **4**, 563–578 (1990).
5. Heintzman, N. *et al.* Distinct and predictive chromatin signatures of transcriptional promoters and enhancers in the human genome. *Nature Genet.* **39**, 311–318 (2007).
6. Segal, E. *et al.* A genomic code for nucleosome positioning. *Nature* **44**, 772–778 (2006).
7. Picardi, E. & Pesole, G. Computational methods for ab initio and comparative gene finding. *Methods Mol. Biol.* **609**, 269–284 (2010).
8. Ashburner, M. *et al.* Gene ontology: tool for the unification of biology. *Nature Genet.* **25**, 25–29 (2000).
9. Fraser, A. G. & Marcotte, E. M. A probabilistic view of gene function. *Nature Genet.* **36**, 559–564 (2004).
10. Beer, M. A. & Tavazoie, S. Predicting gene expression from sequence. *Cell* **117**, 185–198 (2004).
11. Karlic, R., R. Chung, H., Lasserre, J., Vlahovicek, K. & Vingron, M. Histone modification levels are predictive for gene expression. *Proc. Natl Acad. Sci. USA* **107**, 2926–2931 (2010).
12. Ouyang, Z., Zhou, Q. & Wong, H. W. ChIP–seq of transcription factors predicts absolute and differential gene expression in embryonic stem cells. *Proc. Natl Acad. Sci. USA* **106**, 21521–21526 (2009).
13. Friedman, N. Inferring cellular networks using probabilistic graphical models. *Science* **303**, 799–805 (2004).
14. Hastie, T., Tibshirani, R. & Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference and Prediction* (Springer, 2001). **This book provides an overview of machine learning that is suitable for students with a strong background in statistics.**
15. Hamelryck, T. Probabilistic models and machine learning in structural bioinformatics. *Stat. Methods Med. Res.* **18**, 505–526 (2009).
16. Swan, A. L., Mobasheri, A., Allaway, D., Liddell, S. & Bacardit, J. Application of machine learning to proteomics data: classification and biomarker identification in postgenomics biology. *OMICS* **17**, 595–610 (2013).
17. Upstill-Goddard, R., Eccles, D., Fliege, J. & Collins, A. Machine learning approaches for the discovery of gene–gene interactions in disease data. *Brief. Bioinform.* **14**, 251–260 (2013).
18. Yip, K. Y., Cheng, C. & Gerstein, M. Machine learning and genome annotation: a match meant to be? *Genome Biol.* **14**, 205 (2013).
19. Day, N., Hemmaplardh, A., Thurman, R. E., Stamatoyannopoulos, J. A. & Noble, W. S. Unsupervised segmentation of continuous genomic data. *Bioinformatics* **23**, 1424–1426 (2007).
20. Ernst, J. & Kellis, M. ChromHMM: automating chromatin-state discovery and characterization. *Nature Methods* **9**, 215–216 (2012). **This study applies an unsupervised hidden Markov model algorithm to analyse genomic assays such as ChIP–seq and DNase-seq in order to identify new classes of functional elements and new instances of existing functional element types.**
21. Hoffman, M. M. *et al.* Unsupervised pattern discovery in human chromatin structure through genomic segmentation. *Nature Methods* **9**, 473–476 (2012).
22. Chapelle, O., Schölkopf, B. & Zien, A. (eds) *Semi-supervised Learning* (MIT Press, 2006).
23. Stamatoyannopoulos, J. A. Illuminating eukaryotic transcription start sites. *Nature Methods* **7**, 501–503 (2010).
24. Boser, B. E., Guyon, I. M. & Vapnik, V. N. in *A Training Algorithm for Optimal Margin Classifiers* (ed. Haussler, D.) 144–152 (ACM Press, 1992). **This paper was the first to describe the SVM, a type of discriminative classification algorithm.**
25. Noble, W. S. What is a support vector machine? *Nature Biotech.* **24**, 1565–1567 (2006). **This paper describes a non-mathematical introduction to SVMs and their applications to life science research.**
26. Ng, A. Y. & Jordan, M. I. *Advances in Neural Information Processing Systems* (eds Dietterich, T. *et al.*) (MIT Press, 2002).
27. Jordan, M. I. Why the logistic function? a tutorial discussion on probabilities and neural networks. *Computational Cognitive Science Technical Report 9503* [online], http://www.ics.uci.edu/~dramanan/teaching/ics273a_winter08/homework/jordan_logistic.pdf (1995).
28. Wolpert, D. H. & Macready, W. G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1**, 67–82 (1997). **This paper provides a mathematical proof that no single machine learning method can perform best on all possible learning problems.**
29. Yip, K. Y. *et al.* Classification of human genomic regions based on experimentally determined binding sites of more than 100 transcription-related factors. *Genome Biol.* **13**, R48 (2012).
30. Urbanowicz, R. J., Granizo-Mackenzie, D. & Moore, J. H. in *Proceedings of the Parallel Problem Solving From Nature* 266–275 (Springer, 2012).
31. Brown, M. *et al.* in *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology* (ed. Rawlings, C.) 47–55 (AAAI Press, 1993).
32. Bailey, T. L. & Elkan, C. P. in *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology* (eds Rawlings, C. *et al.*) 21–29 (AAAI Press, 1995).
33. Schölkopf, B. & Smola, A. *Learning with Kernels* (MIT Press, 2002).
34. Leslie, C. *et al.* (eds) *Proceedings of the Pacific Symposium on Biocomputing* (World Scientific, 2002).
35. Rätsch, G. & Sonnenburg, S. in *Kernel Methods in Computational Biology* (eds Schölkopf, B. *et al.*) 277–298 (MIT Press, 2004).
36. Zien, A. *et al.* Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics* **16**, 799–807 (2000).
37. Saigo, H., Vert, J.-P. & Akutsu, T. Optimizing amino acid substitution matrices with a local alignment kernel. *BMC Bioinformatics* **7**, 246 (2006).
38. Jaakkola, T. & Haussler, D. *Advances in Neural Information Processing Systems 11* (Morgan Kauffmann, 1998).
39. Shawe-Taylor, J. & Cristianini, N. *Kernel Methods for Pattern Analysis* (Cambridge Univ. Press, 2004). **This textbook describes kernel methods, including a detailed mathematical treatment that is suitable for quantitatively inclined graduate students.**
40. Peña-Castillo, L. *et al.* A critical assessment of *M. musculus* gene function prediction using integrated genomic evidence. *Genome Biol.* **9**, S2 (2008).
41. Sonnhammer, E., Eddy, S. & Durbin, R. Pfam: a comprehensive database of protein domain families based on seed alignments. *Proteins* **28**, 405–420 (1997).
42. Apweiler, R. *et al.* The InterPro database, an integrated documentation resource for protein families, domains and functional sites. *Nucleic Acids Res.* **29**, 37–40 (2001).
43. Pavlidis, P., Weston, J., Cai, J. & Noble, W. S. Learning gene functional classifications from multiple data types. *J. Computat. Biol.* **9**, 401–411 (2002).
44. Lanckriet, G. R. G., Bie, T. D., Cristianini, N., Jordan, M. I. & Noble, W. S. A statistical framework for genomic data fusion. *Bioinformatics* **20**, 2626–2635 (2004).
45. Troyanskaya, O. G., Dolinski, K., Owen, A. B., Altman, R. B. & Botstein, D. A. Bayesian framework for combining heterogeneous data sources for gene function prediction (in *Saccharomyces cerevisiae*). *Proc. Natl Acad. Sci. USA* **100**, 8348–8353 (2003).
46. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann, 1998). **This textbook on probability models for machine learning is suitable for undergraduates or graduate students.**
47. Song, L. & Crawford, G. E. DNase-seq: a high-resolution technique for mapping active gene regulatory elements across the genome from mammalian cells. *Cold Spring Harbor Protoc.* **2**, pdb.prot5384 (2010).
48. Wasson, T. & Hartemink, A. J. An ensemble model of competitive multi-factor binding of the genome. *Genome Res.* **19**, 2102–2112 (2009).
49. Pique-Regi, R. *et al.* Accurate inference of transcription factor binding from DNA sequence and chromatin accessibility data. *Genome Res.* **21**, 447–455 (2011).
50. Cuellar-Partida, G. *et al.* Epigenetic priors for identifying active transcription factor binding sites. *Bioinformatics* **28**, 56–62 (2011).
51. Ramaswamy, S. *et al.* Multiclass cancer diagnosis using tumor gene expression signatures. *Proc. Natl Acad. Sci. USA* **98**, 15149–15154 (2001).
52. Glaab, E., Bacardit, J., Garibaldi, J. M. & Krasnogor, N. Using rule-based machine learning for candidate disease gene prioritization and sample classification of cancer gene expression data. *PLoS ONE* **7**, e39932 (2012).
53. Tibshirani, R. J. Regression shrinkage and selection via the lasso. *J. R. Statist. Soc. B* **58**, 267–288 (1996). **This paper was the first to describe the technique known as lasso (or L_1 regularization), which performs feature selection in conjunction with learning.**
54. Urbanowicz, R. J., Granizo-Mackenzie, A. & Moore, J. H. An analysis pipeline with statistical and visualization-guided knowledge discovery for Michigan-style learning classifier systems. *IEEE Comput. Intell. Mag.* **7**, 35–45 (2012).
55. Tikhonov, A. N. On the stability of inverse problems. *Dokl. Akad. Nauk SSSR* **39**, 195–198 (1943). **This paper was the first to describe the now-ubiquitous method known as L_2 regularization or ridge regression.**
56. Keogh, E. & Mueen, A. *Encyclopedia of Machine Learning* (Springer, 2011).
57. ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome. *Nature* **489**, 57–74 (2012).
58. Manning, C. D. & Schütze, H. *Foundations of Statistical Natural Language Processing* (MIT Press, 1999).
59. Davis, J. & Goadrich, M. *Proceedings of the International Conference on Machine Learning* (ACM, 2006). **This paper provides a succinct introduction to precision-recall and receiver operating characteristic curves, and details under which scenarios these approaches should be used.**

60. Cohen, J. Weighted κ: nominal scale agreement provision for scaled disagreement or partial credit. *Psychol. Bull.* **70**, 213 (1968).
61. Luengo, J., García, S. & Herrera, F. On the choice of the best imputation methods for missing values considering three groups of classification methods. *Knowl. Inf. Syst.* **32**, 77–108 (2012).
62. Troyanskaya, O. *et al.* Missing value estimation methods for DNA microarrays. *Bioinformatics* **17**, 520–525 (2001).
This study uses an imputation-based approach to handle missing values in microarray data. The method was widely used in subsequent studies to address this common problem.
63. Kircher, M. *et al.* A general framework for estimating the relative pathogenicity of human genetic variants. *Nature Genet.* **46**, 310–315 (2014).
This study uses a machine learning approach to estimate the pathogenicity of genetic variants using a framework that takes advantage of the fact that natural selection removes deleterious variation.
64. Qiu, J. & Noble, W. S. Predicting co-complexed protein pairs from heterogeneous data. *PLoS Comput. Biol.* **4**, e1000054 (2008).
65. Friedman, N., Linial, M., Nachman, I. & Pe'er, D. Using Bayesian networks to analyze expression data. *J. Comput. Biol.* **7**, 601–620 (2000).
66. Bacardit, J. & Llorà, X. Large-scale data mining using genetics-based machine learning. *Wiley Interdiscip. Rev.* **3**, 37–61 (2013).
67. Koski, T. J. & Noble, J. A review of Bayesian networks and structure learning. *Math. Applicanda* **40**, 51–103 (2012).
68. Pearl, J. *Causality: Models, Reasoning and Inference* (Cambridge Univ. Press, 2000).

Competing interests statement
The authors declare no competing interests.

FURTHER INFORMATION

1000 Genomes Project: http://www.1000genomes.org
100,000 Genomes Project: http://www.genomicsengland.co.uk
4D Nucleome: https://commonfund.nih.gov/4Dnucleome/index
ENCODE: http://www.encodeproject.org
InterPro: http://www.ebi.ac.uk/interpro/
Machine learning on Coursera: https://www.coursera.org/course/ml
Machine learning open source software: https://mloss.org/software/
Pfam: http://pfam.xfam.org/
PyML: http://pyml.sourceforge.net/
Roadmap Epigenomics Project: http://www.roadmapepigenomics.org
Weka 3: http://www.cs.waikato.ac.nz/ml/weka/

ALL LINKS ARE ACTIVE IN THE ONLINE PDF