Briefings in Bioinformatics, 17(5), 2016, 831–840

doi: 10.1093/bib/bbv082
Advance Access Publication Date: 26 September 2015
Paper

Correct machine learning on protein sequences: a peer-reviewing perspective

Ian Walsh, Gianluca Pollastri and Silvio C. E. Tosatto

Corresponding author: Silvio C. E. Tosatto, Dept. of Biomedical Sciences, University of Padua, viale G. Colombo 3, 35131 Padova, Italy. Tel.: +39 049 827 6269; Fax: +39 049 827 6260; E-mail: silvio.tosatto@unipd.it

Abstract

Machine learning methods are becoming increasingly popular to predict protein features from sequences. Machine learning in bioinformatics can be powerful but carries also the risk of introducing unexpected biases, which may lead to an overestimation of the performance. This article espouses a set of guidelines to allow both peer reviewers and authors to avoid common machine learning pitfalls. Understanding biology is necessary to produce useful data sets, which have to be large and diverse. Separating the training and test process is imperative to avoid over-selling method performance, which is also dependent on several hidden parameters. A novel predictor has always to be compared with several existing methods, including simple baseline strategies. Using the presented guidelines will help nonspecialists to appreciate the critical issues in machine learning.

Key words: machine learning; protein sequence; posttranslational modification; predictor; training; evaluation

Introduction

Recently, next-generation sequencing technologies have released a wealth of protein sequences [1] to the extent that biocuration is increasingly lagging behind [2]. Machine learning (ML) based on small subsets of protein data with experimental annotation has become a vital tool in expanding our proteomic understanding [3–5]. Thus, ML will have an increasingly important role in understanding genomic data in the future.

ML sequence techniques, in their 'supervised' form, involve learning a mapping from primary amino-acid sequences to some important molecular properties (for a recent review, see [6]). The attractiveness of this paradigm is epitomized by entire scientific communities coming together to address it. The Critical Assessment of techniques for protein Structure Prediction (CASP) [7] has mainly tried to predict three-dimensional structure from sequence, but also included subcategories such as secondary structure prediction, intrinsically disordered regions and residue–residue contacts. Recently, the Critical Assessment of Function Annotation (CAFA) experiment [8] tried to assign function to sequences on a large scale. Both CAFA and CASP have shown considerable success assessing various methods (many ML), partly because the molecular properties they try to map are evolutionarily conserved.

In this article, we describe common problems that arise when learning molecular properties at the protein primary structure level. A set of common mistakes seen in the literature and experienced in the peer review process is provided. ML developers can use the common pitfalls as a checklist in the initial design phase and implementation of their algorithm. They should address most of the points raised to maximize the chance of publication. The main focus of this work however is to help the peer reviewer because ML is not always inherently intelligible to nonexperts. In general, learning molecular properties, describing the algorithm and reporting high accuracy is not enough to guarantee a correct algorithm.

Ian Walsh is a PostDoc at the BioComputing UP laboratory, University of Padua. His main research interest is to develop novel prediction methods for structural bioinformatics using machine learning.
Gianluca Pollastri is a principal investigator in the Complex and Adaptive Systems Laboratory at University College Dublin. His research focuses on machine learning for structured data, mainly applied to bioinformatics and chemoinformatics.
Silvio Tosatto is principal investigator of the BioComputing UP laboratory at the Department of Biomedical Sciences, University of Padua. He is also an editor for the journals Amino Acids, Biochimie, PeerJ and PLoS One. His research focuses on structural bioinformatics and its application to biomedical problems.
Submitted: 6 May 2015; **Received (in revised form):** 19 August 2015

Experience in the peer review process has taught us that often ML is applied incorrectly in the molecular biology domain. Poorly designed experiments produce true prediction performances that are worse than claimed or even random, and can slip through the peer review processing net. As an example, a recent review showed that 9 of 11 lysine posttranslational modification (PTM) prediction tools truly behave no better than random [9]. The goal of this article is to provide a set of guidelines for the development of usable and publishable ML methods on protein sequences. By usable we mean that the biological community can execute them with ease to annotate protein sequences with little current experimental information. Future peer reviewers can use this article to filter poorly designed algorithms from publication, in turn whittling down ML methods to a few quality predictors. The core of the article discusses outstanding concerns in the design of any type of ML predictor for biological sequences, with the section on 'Machine Learning on Sequences' only briefly describing ML for context.

ML on sequences

ML algorithms learn behaviors from predefined data, but their output behavior is not explicitly programmed [10]. Learning molecular properties of protein sequences is a common task in bioinformatics, with a wide range of applications including pinpointing functional residues [11] and determining protein–protein interactions [12] to name just two. In their 'supervised' form, given model parameters θ, ML algorithms take as input a sequence, s_i, and produce a prediction output, y_i. Strictly speaking s_i is not the actual protein sequence but a feature representation of the sequence. The model parameters θ are optimized to learn some experimentally annotated molecular behavior on a data set of protein sequences. In the supervised case, target labels, t_i, are available as annotations to their corresponding sequence s_i. Figure 1A shows the general flow for supervised techniques highlighting the main players, namely θ, s_i, y_i and t_i. In supervised methods, θ is optimized to maximize the closeness of the predicted properties to their true labeled target (e.g. minimizing $\Sigma_i|t_i\text{-}y_i|$, where the sum is extended over all *i* data points). With the increase in quantity and quality of experimental annotations *in vivo*/*in vitro* (i.e. the availability of many high quality t_i), supervised learning techniques have become prevalent in the literature, including neural networks (NNs), support vector machines (SVMs) and random forests (RFs) to name just a few. The majority of these techniques are classification based (assignment of categorical labels) and to a lesser extent regression (assignment of real number measurements). One of the earliest examples of supervised classification was secondary structure prediction, classifying each amino acid of the protein sequence into α-helix, β-strand or coil [13–15]. Also common are unsupervised techniques and are usually used to group data when target labels are missing [see Figure 1B]. Although supervised and unsupervised methods are clearly the most common in the literature, other ML techniques such as semi-supervised and reinforcement learning may gain popularity in the future. While most of the topics discussed in this article could be applied to all, supervised and unsupervised methods are our main concern. For a comprehensive review of ML techniques in bioinformatics, see [16]. Although many protein ML papers contain flaws, especially on submission to a journal, a list of examples showing good practice is shown in Supplementary Table S1. This table should give the reader a feeling for good practices before we highlight common problems in the next sections.

Pitfalls

In this section and the following ones, we present some common problems when creating ML methods for protein sequence-based problems. These are summarized as a checklist for peer review concerns in Supplementary Table S2, as quality requirements for publication are often ambiguous in the field of protein sequence prediction. Moreover, the points raised should be considered early during algorithm design.

Data quality and representativeness

The most important concern in any ML approach is the design of large and diverse, high-quality learning sets. In protein sequence-based prediction, this often involves extracting structural or functional information from databases such as the Protein Data Bank [17] or UniProt [18]. In many cases, data sets are constructed without any filters to guarantee data quality, and this may introduce noise, which can be learned alongside signal (see 'Over-fitting and under-fitting'). When reviewing, data design concerns should arise when quality is not mentioned. The best criterion to determine data quality is by expert domain knowledge, but sometimes a peer reviewer may lack this expertise. To help, the quality of the data is often given in the source databases, and at the least, the reviewer should check each source. For example, when predicting a PTM such as phosphorylation, mining data with the 'phosphorylation' keyword in UniProt is too basic. Data should be filtered to only include the PTM sites with UniProt-flagged experimental evidence (see Supplementary Table S1 for another example). Algorithms learned with noisy low-quality data sets are often of limited use for biological inference. While there are examples in the literature of effective use of medium- to low-quality data to enhance predictive performances [19], at the least, the inclusion of low-quality data in the training and testing sets should be assessed experimentally.

A common term for large and diverse sets in the literature is representativeness [20]. For example, in supervised classification, all target labels should be of sufficient quantity, and in supervised regression, all target values should be equally spread. Ideally, a high-quality representative set can be used to learn the often more important, complex nonlinear relationships in the data. This can however lead to over-fitting on small data samples. Reduction of data size to a small amount in favor of high-quality filters is acceptable if caution is taken to test for and prevent over-fitting, e.g. by limiting the learning capabilities to linear SVMs or Perceptron algorithms. We will discuss the issue of over-fitting and the type of relationships an algorithm can learn in 'Over-fitting and under-fitting'. Table 1 gives a list of the allowed combinations with respect to representativeness, quality and the ability of an algorithm to learn complex relationships. To have low quality and representativeness usually means the *in vivo*/*in vitro* annotations have not reached enough maturity to perform ML. This should be particularly examined when the paper is claiming a novel prediction problem.

Data splits

Once the data set is finalized, the next stage is to train and measure the performance of the ML models. Obviously, evaluating models on the same data used to optimize the parameters gives no reliable indication of the behavior of the predictor on unseen data ('generalization'). The simplest approach is to split the data into 'training' and 'testing' for parameter optimization and evaluation, respectively. However, the relative sizes of

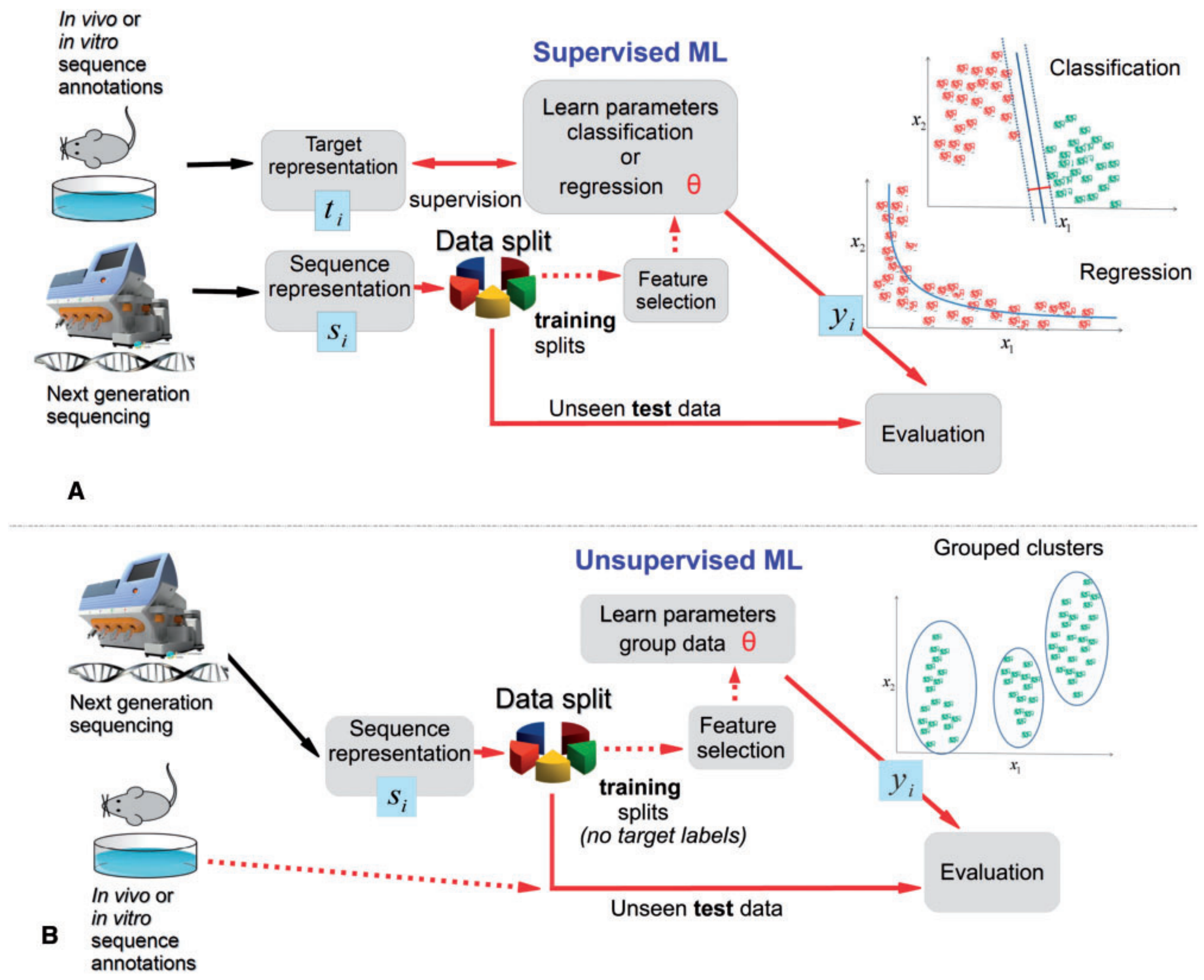

Figure 1. Typical flow of protein sequence ML approaches (dashed lines for optional paths). (**A**) In Supervised learning both input sequence and output target labels are required. Evaluation must be done on unseen data not used to optimize parameters. (**B**) Unsupervised learning only requires input sequence representations and target labels are often missing. Evaluation is a much trickier concept for unsupervised learning but can be done by measuring the consistency of groups on unseen data. Also, unsupervised learning evaluation may use target labels, if some are available.

these sets is a hidden experimental parameter that can greatly influence the final performance, especially when the overall amount of data is small and different data splits yield large deviations. This problem is part of a more general issue (selection of hyper-parameters, or 'magic numbers'), which we discuss in more detail in 'Over-fitting and under-fitting'. While resorting to a single fixed split between training and testing data (e.g. one-fifth of data used as a test set and four-fifth as a training set) is acceptable when data are abundant, its effect on the results should at least be assessed by randomly resampling the subdivision many times and estimating statistical variations.

'Cross-validation' provides a means to select ML models and perform evaluation in a fair manner. A K-fold cross-validation splits the data in K equal folds, with $K - 1$ folds used for training the model, and the remaining test fold used for measuring generalization. An estimate of generalization on the entire data set is obtained by averaging performance on all test folds. Leave-one-out cross-validation (also known as jack-knife) is the extreme where each test fold contains only one data point. The advantages of cross-validation over single train/test splits are numerous. It is statistically more robust [21, 22], and all the data are used for testing, and potentially nearly all for training, which is particularly important for small data sets. It also allows the calculation of statistical stability on testing folds, indicating if the algorithm is robust to changing data. Learning in cross-validation will result in K trained models, and the final predictor can be a combination (ensemble) of the K trained models, which results in the inclusion of all the data sets in the process. Although there is no objective way to test this ensemble predictor on the original data, a further, independent set can be used for this purpose if available. Concerns should be raised when there is no cross-validation and data size is small. Also, the stability of the algorithm should be measured across the folds or by resampling train/test splits (e.g. performance standard deviations on all folds).

Sequence diversity

Clusters of similar data are problematic for learning. A data set with large quantities of similar data will lead to biased learning. Clusters of similar data may be split across the training data used for learning and the testing data used for performance calculations. This will produce over-optimistic generalization measurements as the ML algorithm learns, but easily recognizes the same or similar data in the test set.

A common technique is to redundancy reduce a set by finding similar sequence pairs above a residue identity threshold and discard one (typically the lower-quality one). Well-established algorithms such as CD-Hit [23], BlastClust [24], PISCES [25] and TESE [26] are effective tools to reduce sets by sequence identity. Because these reduction algorithms often rely on local sequence alignments and a large number of proteins are multidomain, some unique domains may be lost to the reduction. It may be an option to preprocess the sequences into domains using tools such as PFAM [27]. An important consideration when diversifying data is the learning focus. Motif-based learning focuses on part of the sequence where only selected amino acids are relevant and others ignored (e.g. lysine-centered motifs in the ubiquitination PTM). In this case, each data point is a short peptide and not the entire sequence. A common mistake in motif-based learning is to only remove redundant sequences, as this may not remove similar or even identical motifs inside the sequence. However, the balance here is to require enough identity to in fact capture the motif and not enough to have clusters of similar motifs within the same class (see Supplementary Figure S1 for an example). As the length of the motif is the important factor, a plot such as in Supplementary Figure S1 will ease any concerns about similar clusters of motifs. Figure 2 shows an example situation where diversity is applied at the full sequence and the motif level.

While a sufficient level of identity for redundancy reduction is debatable, identity thresholds of 25% for secondary structure and intrinsic disorder [28, 29] have been mandated for full sequences, as above 25% structural properties can often be assigned by similarity [30]. For motifs, the choice is trickier because the sequence stretch is a small fragment. Thresholds at 60% [31] and 30% [32] of the motif length have been proposed for PTM prediction. Lack of filtering or discussion thereof in a manuscript should be considered a major concern and lead to an automatic request for clarifications or outright rejection.

Table 1. Combinations of acceptable data quality, representativeness and algorithm ability (e.g. high variance algorithm that can model complex processes)

Data			
Quality	Representativeness	Algorithm ability	Acceptable
Good	Good	Good	Yes
Good	Good	Poor	Yes
Good	Poor	Poor	Yes
Good	Poor	Good	Improve[a]
Poor	Good	Good	Improve[b]
Poor	Good	Poor	Improve[b]
Poor	Poor	Good	Reject
Poor	Poor	Poor	Reject

[a]Possible over-fitting certainly needs over-fitting checks.

[b]Bad signal-to-noise ratio, performance needs to be evaluated on high-quality and representative test set but could be used for training.

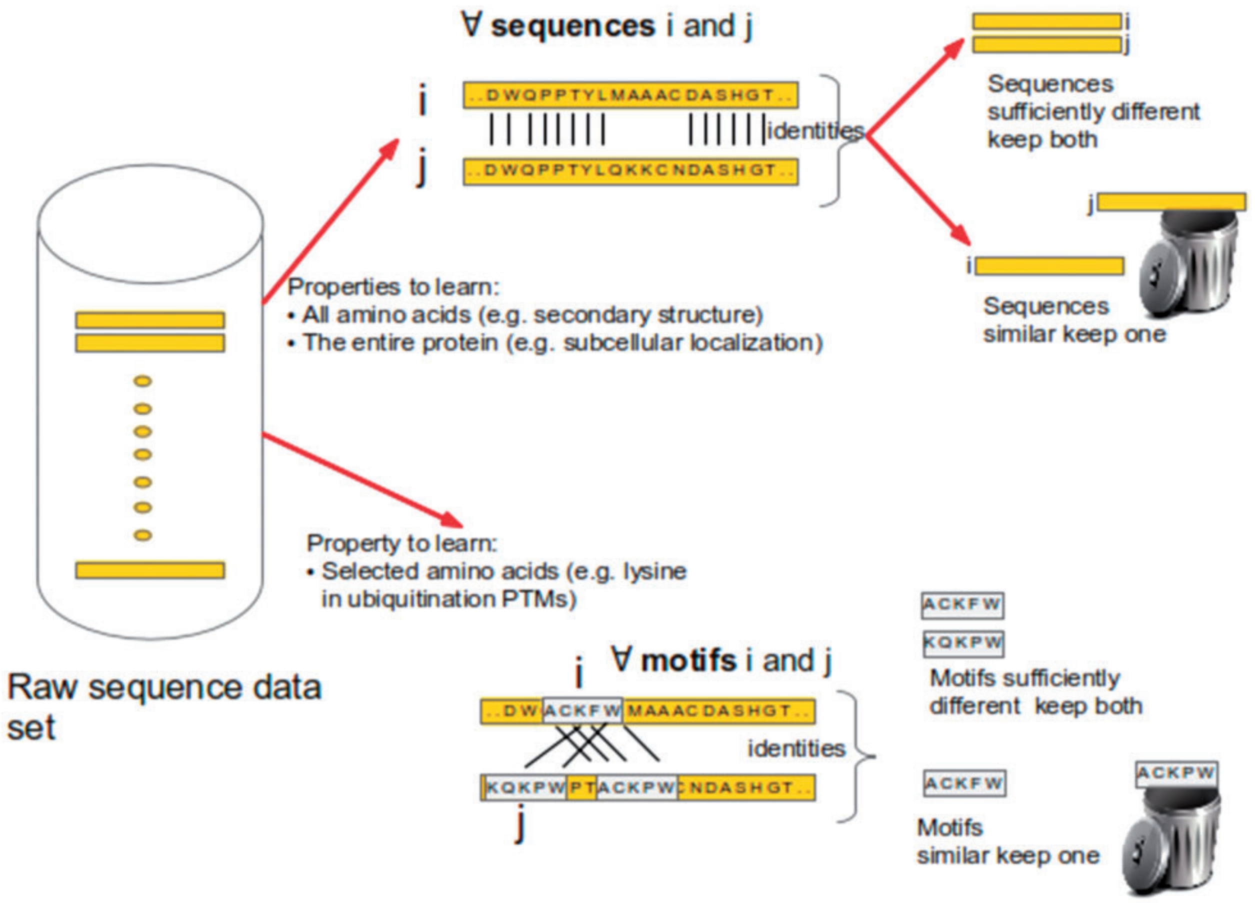
∀ sequences i and j
i
..DWQPPTYLMAAACDASHGT..
identities
j
..DWQPPTYLQKKCNDASHGT..
Sequences sufficiently different keep both
Sequences similar keep one
Properties to learn:
• All amino acids (e.g. secondary structure)
• The entire protein (e.g. subcellular localization)
Property to learn:
• Selected amino acids (e.g. lysine in ubiquitination PTMs)
Raw sequence data set
∀ motifs i and j
..DWACKFWMAAACDASHGT..
identities
KQKPWPTACKPWCNDASHGT..
ACKFW
KQKPW
Motifs sufficiently different keep both
ACKFW
ACKPW
Motifs similar keep one

Figure 2. Two possible focuses for redundancy reduction. The top scenario assumes that the full sequence is important for learning. In this case, every amino acid or the entire protein has learnable molecular properties. The bottom shows a special case where the learning is focused on particular sites or motifs (e.g. lysines (K) for ubiquitination).

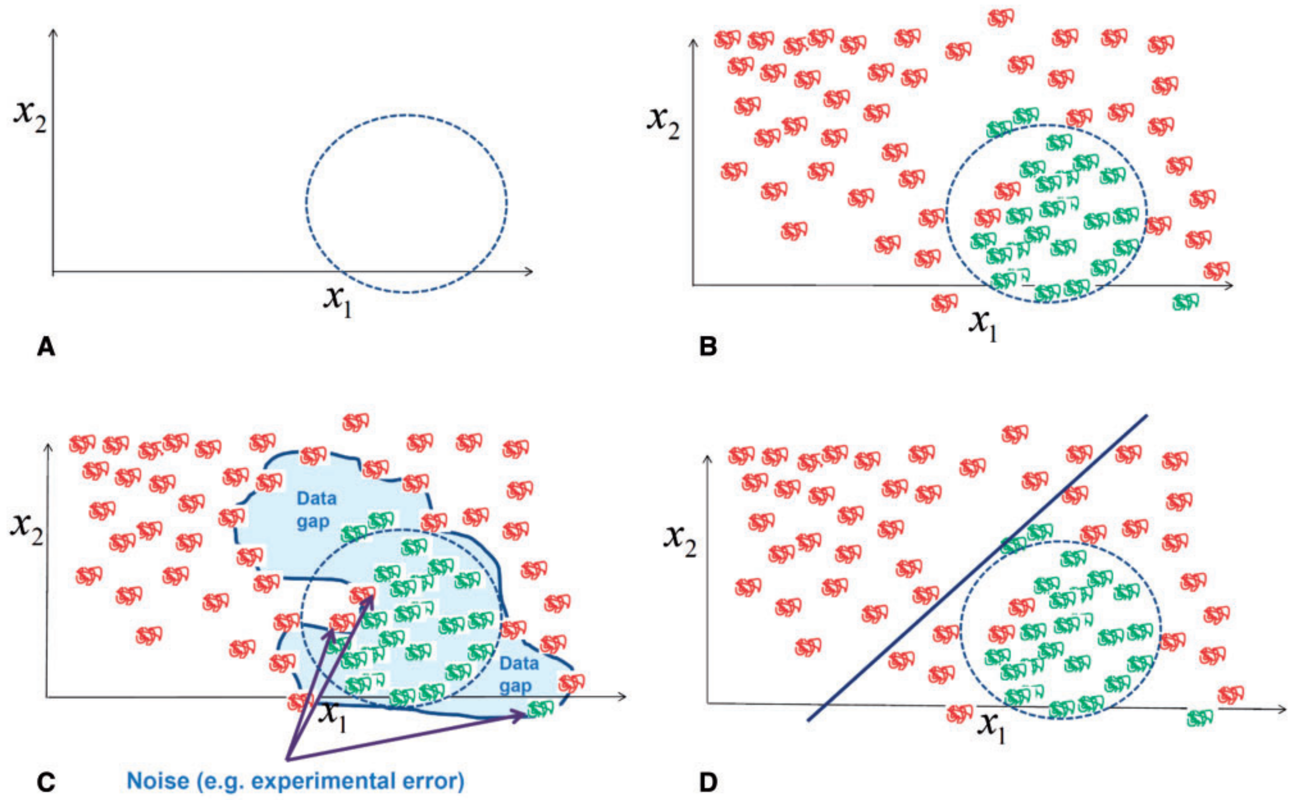

Figure 3. A simple hypothetical example showing the problems of over-fitting and under-fitting in supervised classification. (**A**) The underlying process to model is represented in two-dimensional feature space as a circle. (**B**) *In vivo/in vitro* protein experiments reveal some data about the problem in a binary classification scenario. (**C**) Over-fitting: a model which has too much variance models the noise and the data gaps. (D) Under-fitting: a model with high bias is too simple to capture the underlying process. Ideally, a balance needs to be met, and some simple techniques should be used to strike the balance.

Over-fitting and under-fitting

An ML algorithm can only model the underlying process through the training data. The more expressive an algorithm (i.e. the more free parameters it has), the more complex processes it can model, but also the more prone it is to over-fitting the data. Figure 3 shows a simple two-dimensional modeling problem with experimentally determined labels. Over-fitting occurs when the algorithm has enough complexity to model noise and gaps in the data alongside signal, generally leading to perfect or near-perfect performances on the training set, but poor generalization on unseen data. On the contrary, a model that is not expressive enough may under-fit, that is, perform more poorly than the alternatives on 'both' training and unseen data. Typically in ML, finding a model with the ideal expressive power for a problem is a tricky objective. In some bioinformatics applications, simple algorithms are enough, or at least they are all that can be inferred from scarce data, but they must be rigorously benchmarked on unseen data showing acceptable accuracy. However, in most cases, sequence-based learning requires sophisticated nonlinear methods (e.g. multi-layer perceptrons) that have the potential to over-fit if not carefully designed and dimensioned.

The main indicator of over-fitting is the difference between training and testing performance. However, the training performance is often omitted in the literature, giving no indication of the generalization drop. Another clue is a large number of parameters relative to the number of training examples. Many input features or hidden neurons in NNs increase the number of parameters and should also give clues. It is however not always the case that low parameter count implies low expressiveness, for example complex kernels in SVMs [33] and the number of trees in RF [34] can produce high variance with few explicit parameters. Unfortunately, this requires expert knowledge of each ML algorithm, which is often not available in the peer review system. It is therefore important, especially when a small data set is used, that both the training and testing accuracies are reported. A simple plot as in Figure 4A is the clearest evidence for over-fitting when error can be measured over time or Figure 4B when the algorithm finds an optimal configuration.

Constraining a model to the optimal expressive power may be based on: (a) stopping learning early to prevent the over-optimization of parameters, e.g. introduce a third validation set to flag for over-fitting, see Supplementary Figure S1A; (B) regularization of the parameters, e.g. weight decay in NNs or width of the separating margin in SVMs; (C) reduction of the number of parameters by manual or automatic input feature reduction; (D) testing different algorithms of different complexity. Often, the ideal model is found by trial and error, based on considerable expertise. Crucially, trial and error may lead to many different estimates on the test set as many trials are performed. As performance on the test set is itself a random variable drawn out of a probability distribution (with a deviation that is larger when data sets are small), running many full cross-validation experiments and reporting the results for the 'best' one typically leads to overestimating real performances. We will discuss this issue in more detail in 'Magic numbers'. In general, a peer reviewer should be concerned about small data sets of low quality and overly complex or simple learning algorithms. In particular,

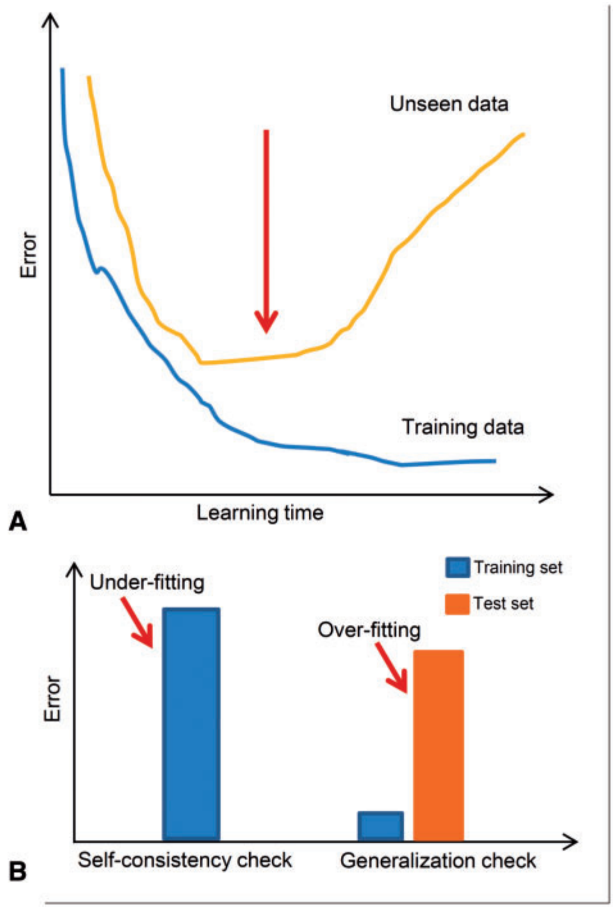

Figure 4. Clear examples of over-fitting and under-fitting. Red arrows flag dangers. (**A**) Over-learning: the performance on the training data increases (error decreases), while at some point (red arrow), the unseen generalization performance degrades significantly. (**B**) Self-consistency check for under-fitting and generalization check for over-fitting.

if no attempts are made to report the generalization performance drop or to describe the model selection process, reasons should be given. A good overview of bias, variance and fitting data sets can be found in [35].

Automatically selecting input features

Once a sequence is encoded into a set of features, many can be redundant. One obvious advantage of feature selection (FS) is a decrease in the overall number of tunable parameters in the algorithm and, as a consequence, over-fitting is less likely. Decreased numbers of input features may also increase speed, which is important if the algorithm is to be used on a large scale. Most importantly, a reduced list of relevant features aids in understanding the important characteristics of the problem at hand. FS can be divided into three categories [36]: (i) 'wrappers' use the ML algorithm as a black box to select features based on their performance, (ii) 'filters' select feature subsets without considering the ML algorithm and (iii) 'embedded' techniques are part of the ML algorithm training procedure. A description of various FS methods is beyond the scope of this article, but for a review, see [37].

One obvious danger that biases performance is to conduct FS on the entire data set and then proceed with the separation of training and test splits. In this case, training parameters are indirectly influenced by the evaluation set, and therefore overoptimistic performances are possible. More complicated issues arise in FS methods with small amounts of data. In this case, the selected features may produce good performances on the training set but perform poorly on the unseen split [38]. Other dangers when using the ML algorithm to select its own features (i.e. wrappers and embedded) may be less obvious. For example, with one simple train–test split, an FS algorithm may do well by chance. More alarmingly, a researcher may adopt a trial and error approach choosing, and only reporting, one of the many FS algorithms that generalize best. The number of selected features and the chosen selection algorithm are generally hidden parameters, FS should be evaluated in cross-validation (or in resampled train/test splits) varying the evaluation. Figure 5 shows the flow of a 10-fold cross-validation procedure with FS. The most important factor is that all parameters, including the hidden FS parameters, are tuned in the training only. Using this approach, the performance, number and type of features may change, depending on the training fold, allowing assessment of the FS stability and variations (Figure 5). Another approach that is gaining popularity is nested cross-validation, where the part shown in Figure 5A is split into an additional cross-validation loop. For further reading on nested cross-validation, see [39]. Ideally, the selected features across the variant training data will show some consistency. However, they may produce slightly different feature subsets on each training fold, and how to combine all models to produce a final model is debatable. In our experience, we found averaging the output of the individual models to be effective. After combining, ideally the final model should be validated on an independent test set (i.e. independent of the cross-validation). Finally, after varying the selection process, a brief description of the selected features and why they are important to the problem should be reported. This is a rich source of information not only for future work in that particular field, but also for the biological meaning of the predictor.

Magic numbers

As a rule, no parameter should be chosen based on the test data. A common occurrence in the literature is to assign hyper-parameter values without any description of the criteria used for their selection. By hyper-parameters, we mean high-level features of the algorithm that are not directly optimized as a function of the input, e.g. window size used to capture the local surrounding of an amino acid, number of clusters in *K*-means, number of trees in RF, regularization parameter C or kernel parameters for SVMs, number of hidden neurons and model selected during a training run in NNs. An often-observed scenario in manuscripts is one in which many different values for one or a group of hyper-parameters are tested and results (e.g. in cross-validation) are produced for each one of them, and the final performances of the algorithm are reported as those of the best combination of hyper-parameters 'on the test set'. The problem with this approach is that, especially when sets are small, much of the variation observed between different tests may be because of stochastic factors rather than to genuine advantages of a particular choice. Running many full cross-validations and assuming that the final performances are the highest value obtained is the same as estimating the average of a random variable by sampling it many times and picking the largest value. When sets of only a few hundred data points are available, this can lead to a drastic overestimate. The simplest approach to guard against this is to introduce a third validation set and select hyper-parameters on it (see Supplementary

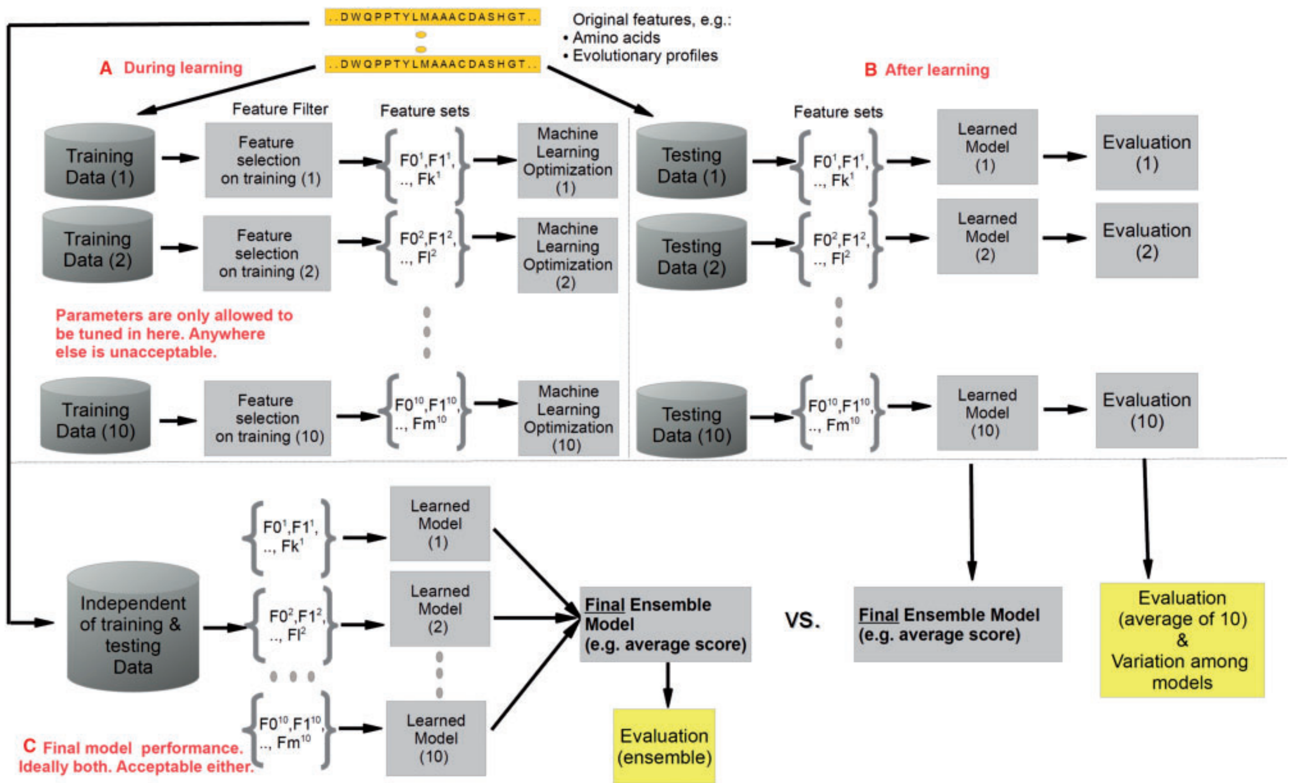
..DWQPPTYLMAAACDASHGT..
Original features, e.g.:
• Amino acids
• Evolutionary profiles
A During learning
B After learning
Feature Filter
Feature sets
Training Data (1)
Feature selection on training (1)
Machine Learning Optimization (1)
Testing Data (1)
Learned Model (1)
Evaluation (1)
Parameters are only allowed to be tuned in here. Anywhere else is unacceptable.
Independent of training & testing Data
Final Ensemble Model (e.g. average score)
VS.
Evaluation (ensemble)
Evaluation (average of 10) & Variation among models
C Final model performance. Ideally both. Acceptable either.

Figure 5. Flow of a 10-fold cross-validation (or 10 resampled train/test split) for a filter FS algorithm. (**A**) Initially the features are extracted and models learned on the features. Parameters can only be optimized in this part of the procedure. (**B**) After learning, the features found during training are used to evaluate the 10 different learned models. (**C**) How to evaluate the 10 different models. The easiest approach is to combine the models in an ensemble. Two approaches can be used to evaluate the final model: either the average of the 10-fold cross-validation or ideally an independent set should also be used.

Figure S1B). A final (single) cross-validation can then be run using the best hyper-parameters found on this validation set. When this is possible (e.g. for multi-layer perceptrons), cross-validation for the 'best' hyper-parameters obtained by trial and error can be repeated many times using different initial random seeds and the average result of these cross-validations reported as an unbiased performance estimate.

Probably, the most common magic number is the window size used to capture the local sequence surroundings. This contains a wealth of information affecting performance considerably and picking a size without an appropriate justification is suspicious. Ideally, all parameter selections should only be determined using the training data, a third validation set or, at the least, literature should be cited with previously determined values. What should be proven beyond doubt is that none of these parameters were 'cherry picked' to perform best on the testing data. Reviewers could ask for a plot as in Figure 4A, with the *x*-axis replaced to show variations of the concerned parameter.

Evaluation

An evaluation of the method is essential. Moreover, it must be comprehensible and statistically sound. Many works simply measure basic aspects of the performance. For the algorithm to be of benefit, it must perform well on unseen data, and it must be an improvement over currently available methods and simple baselines, based on all or most of the appropriate performance measures for the given problem.

Performance calculation must be comprehensive

In binary classification, all typical performance measures are combinations of true positives (tp), true negatives (tn), false positives (fp) and false negatives (fn). A tool to measure both fp and tp is the receiver operating characteristic (ROC) curve (see Figure 6). This measures the fp rate, or 1 − specificity (tn/(fp + tn)), on the *x*-axis and the tp rate, or sensitivity (tp/(tp + fn)), on the *y*-axis, with each point on the curve calculated at different decision thresholds. A ROC can be summarized in a single value, the area under the curve (AUC), with perfect classifiers having an AUC of 1 and random behavior an AUC of 0.5. While the benefit of the ROC and AUC are their independence of any decision threshold, the ROC should be used to define one. Once chosen, the decision threshold defines the proportion of tp, tn, fp and fn often measured as sensitivity and specificity (see Figure 6). This threshold is a parameter and should only be selected on the training set. ROCs can be extended to multiple classes [40], with each class in turn picked as positive and the rest assumed negative, resulting in *N* ROC curves. Similar to the ROC, the precision/recall (PR) curve plots its two measures at varying decision thresholds. Recall and precision can be defined on positives or negatives but are generally used to calculate performance on the rarer positive class among a much larger negative background (e.g. PTM residues among other residues). On positives, recall is simply the sensitivity, while precision measures the fraction of tp in all predicted positives (tp/(tp + fp)). A classification-based predictor must be comprehensively evaluated on all or most of the measures mentioned above. Figure 6 shows an example of both ROC and PR curves and summarizes all the measures using an analogy of picking good fruits.

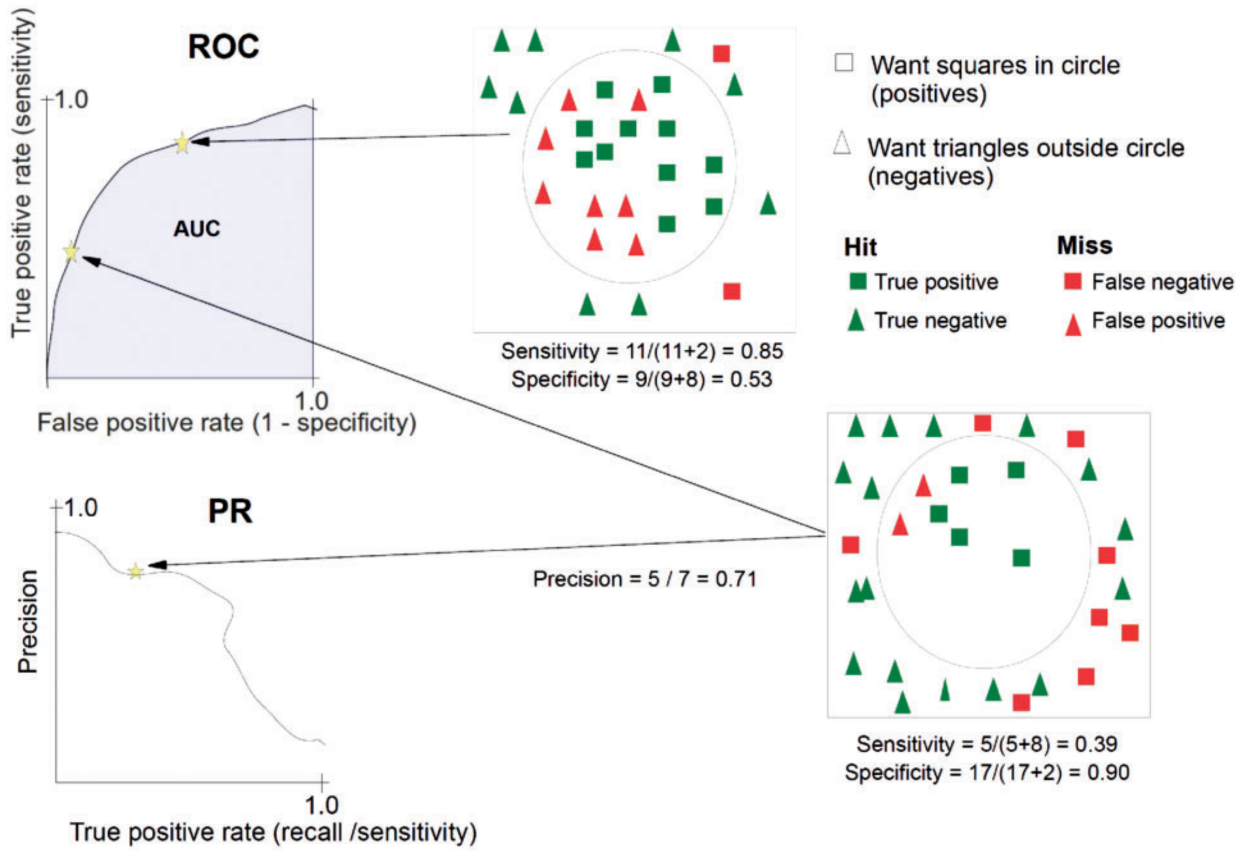
ROC
True positive rate (sensitivity)
1.0
AUC
1.0
False positive rate (1 - specificity)
Want squares in circle (positives)
Want triangles outside circle (negatives)
Hit
True positive
True negative
Miss
False negative
False positive
Sensitivity = 11/(11+2) = 0.85
Specificity = 9/(9+8) = 0.53
PR
1.0
Precision
Precision = 5 / 7 = 0.71
1.0
True positive rate (recall /sensitivity)
Sensitivity = 5/(5+8) = 0.39
Specificity = 17/(17+2) = 0.90

Figure 6. A binary classification setting showing ROC and PR curves. The example shows an objective to collecting good fruits. There are good (square) and rotten (triangle) fruits (positive and negative classes). We want to collect all good fruits in the basket. All rotten (triangle) fruits should remain outside the circle. Sensitivity is the amount of correctly identified good (square) fruits. Specificity is the amount of correctly identified rotten (triangle) fruits. Precision on squares is 'squares correct inside the circle'.

For regression, it is possible to bin the real-numbered property into classes and perform the above analysis. However, it may be more effective to measure the correlation of the predicted and experimental numbers using a simple two-dimensional graph (*y*-axis true values; *x*-axis predicted values). In addition, measures such as the Pearson correlation coefficient and the root mean squared error should be sufficient. For a complete list of supervised classification, regression and unsupervised clustering metrics, we point the reader to Supplementary Table S3.

Imbalanced classification

In classification, one class (often the minority) may be the important property to determine. For example, phosphorylated serines, ubiquitinated lysines and intrinsically disordered residues will be much rarer than their regular counterpart. Because of this imbalance, some ML algorithms may find learning difficult, and many approaches overcome this by extracting balanced examples for training. That is, artificially modifying the background frequencies of different classes. Whether ML algorithms are trained on balanced or imbalanced data sets, it is important that the predictor is evaluated on the distribution in natural sequences and not on artificially constructed balanced distributions. This is also true for classification of naturally skewed distributions in multiple classes and regression.

Specificity on imbalanced binary classification

As mentioned, it is common in learning to have two classes, with the rarer positive class as the main interest. In these scenarios, experimentalists interested in using predictors often desire low fp rates also known as high specificity. Adapting an example from [9], let us imagine a hypothetical situation where an ML algorithm achieves 40%/95% sensitivity/specificity. An experimentalist knows he/she has to examine 100 motifs for a phosphorylation PTM, but unknown to him/her, 10 are phosphorylated, and 90 are not. He/she decides to use an ML algorithm to help cut time. Given its performance evaluation, it returns 9 candidate motifs for examination, correctly predicting 4 of 10 positives and incorrectly determining 5 of the 90 negative motifs. With no a priori knowledge and using the software to guide experiments, a laboratory test of these nine peptides would reveal four of nine were phosphorylated, a favorable scenario for most experimentalists. On the other hand, evaluating candidate motifs of low-specificity algorithms would be rather time-consuming for the experimentalist. For example, at 40%/80% sensitivity/specificity, with 24 candidates, 4 would be correct, with 20 incorrect ones leading to waste of resources. The importance of specificity must be stressed if imbalance is present, but can be relaxed for naturally balanced data.

Comparison with related software, baselines and statistical tests

Probably, the most important evaluation is to compare the final software with competing work. Many papers calculate performances on their self-made data set and claim superiority by reporting self evaluations. This approach is generally flawed. If there are previously constructed methods, their differing performances should be compared 'on the same data'. Two sources of data could be used: (i) independent test set(s) for the method paper under review or (ii) a test set independent of (i.e. redundancy reduced against) the training sets used for previously published methods and the method under review. The latter case requires knowledge on the comparing methods training set and some further data generation, but is completely fair to all methods. The former, although usually less time-consuming, may be biased in favor of the competing methods, as their training data may intersect with the test data. Either case is acceptable as long as the performances are shown to be superior or at least comparable for the data in (i).

To a lesser extent, papers construct a complex algorithm on a novel problem, report good performances and claim originality. Although at the time of submission, there may be no algorithms tackling the same problem, simple baselines must still be constructed. One example of a poorly developed algorithm would have a complex feature encoding for a NN, while a simpler encoding (e.g. basic amino-acid frequencies) can be proven similar or better. In all cases, valid statistical significance tests such as the Student *t*-test (or nonparametric tests if normal distribution does not hold) must be performed. Simply reporting method A has better performance than method B is not sufficient. Statistical tests prove beyond reasonable doubt that the method is indeed a superior and publishable piece of work. For a good review of statistical tests when comparing differing algorithms, see [41].

Meta-algorithms

A recent category of methods use a combination of previously published component methods, preferably producing orthogonal output. Meta techniques have the potential to be state-of-the-art, e.g. in fold recognition [42], secondary structure [43], protein model quality [44], transmembrane helix [45] and PTM prediction [46]. However, there are some concerns, which the peer-review system should acknowledge. First, proof that the method is statistically better than all its individual components and other related algorithms is vital. Second, computational time should be addressed because high accuracy is frequently obtained at the cost of running several predictors in parallel and averaging their output. Third, it has to be acknowledged that these approaches can often turn out to be a simple average of other predictors. Further analysis, e.g. of biologically relevant results, may be necessary to make the manuscript sufficiently interesting for publication.

Availability

Making an algorithm available for use by biologists and developers trying to create improved algorithms is of utmost importance. Packaging the algorithm as freely available software such as an executable or user-friendly web server greatly benefits a manuscript. This also benefits the authors and the journal, as it will increase the impact of the work on the research of biologists and algorithm developers alike. Although software availability is not a necessary requirement, at the least, the paper should be written to a standard where readers can re-implement the algorithm from scratch, and the training/testing data (including alternative splits if present) exactly as described in the paper should be released.

Conclusions

Finding bad practice in ML approaches on protein sequences is difficult. The work is invariably cross discipline, mixing technical computer science, statistics and molecular biology topics. Depending on the journal of submission, there may be a lack of expertise among peer reviewers. For instance, biological journals may lack ML basics, and a computer science journal may find the molecular aspects troublesome. In this article, we address the former problem by summarizing possible concerns that may arise when applying ML to a biological domain. The hope is that this article can become a point of reference, collecting a checklist to aid protein sequence ML design for both authors and peer reviewers. The checklist is summarized in Supplementary Table S2, providing a concise list of all the issues raised in the article and how to address them.

Key Points

- Machine learning methods pose a series of specific challenges to the reviewer, as critical flaws may be concealed.
- Data sets require biological knowledge to compile and must be large enough to allow meaningful results.
- Training and testing sets have to be carefully separated to avoid over-reporting method performance.
- Arguments should be presented for the choice of several parameters.
- Comparisons with available methods should be as extensive as possible, include statistical significance tests, and also include simple baseline strategies such as similarity searches.

Supplementary Data

Supplementary data are available online at http://bib.oxfordjournals.org/.

Acknowledgements

The authors are grateful to members of the BioComputing UP laboratory for insightful discussions.

Funding

This work was supported by COST [BM1405]; Italian Ministry of Health [GR-2011-02346845 to S.T., GR-2011-02347754 to S.T.]; FIRB Futuro in Ricerca [RBFR08ZSXY to S.T.]; and Science Foundation Ireland [10/RFP/GEN2749 to G.P.].

References

1. Schuster SC. Next-generation sequencing transforms today's biology. *Nat Methods* 2008;**5**:16–18.
2. Howe D, Costanzo M, Fey P, *et al*. Big data: the future of biocuration. *Nature* 2008;**455**:47–50.
3. Larrañaga P, Calvo B, Santana R, *et al*. Machine learning in bioinformatics. *Brief Bioinform* 2006;**7**:86–112.
4. Tarca AL, Carey VJ, Chen X, *et al*. Machine learning and its applications to biology. *PLoS Comput Biol* 2007;**3**:e116.
5. Jensen LJ, Bateman A. The rise and fall of supervised machine learning techniques. *Bioinformatics* 2011;**27**:3331–2.
6. Libbrecht MW, Noble WS. Machine learning applications in genetics and genomics. *Nat Rev Genet* 2015;**16**:321–32.

7. Moult J, Fidelis K, Kryshtafovych A, *et al*. Critical assessment of methods of protein structure prediction (CASP)—round x. *Proteins* 2014;**82**:1–6.
8. Radivojac P, Clark WT, Oron TR, *et al*. A large-scale evaluation of computational protein function prediction. *Nat Methods* 2013;**10**:221–7.
9. Schwartz D. Prediction of lysine post-translational modifications using bioinformatic tools. *Essays Biochem* 2012;**52**: 165–77.
10. Samuel AL. Some studies in machine learning using the game of checkers. *IBM J Res Dev* 1959;**3**:210–29.
11. Lee D, Redfern O, Orengo C. Predicting protein function from sequence and structure. *Nat Rev Mol Cell Biol* 2007;**8**:995–1005.
12. Shen J, Zhang J, Luo X, *et al*. Predicting protein-protein interactions based only on sequences information. *Proc Natl Acad Sci USA* 2007;**104**:4337–41.
13. Jones DT. Protein secondary structure prediction based on position-specific scoring matrices. *J Mol Biol* 1999;**292**:195–202.
14. Cuff JA, Clamp ME, Siddiqui AS, *et al*. JPred: a consensus secondary structure prediction server. *Bioinformatics* 1998;**14**:892–3.
15. Rost B, Sander C, Schneider R. PHD—an automatic mail server for protein secondary structure prediction. *Comput Appl Biosci* 1994;**10**:53–60.
16. de Ridder D, de Ridder J, Reinders MJT. Pattern recognition in bioinformatics. *Brief Bioinform* 2013;**14**:633–47.
17. Berman HM, Battistuz T, Bhat TN, *et al*. The protein data bank. *Acta Crystallogr D Biol Crystallogr* 2002;**58**:899–907.
18. UniProt Consortium. Activities at the Universal Protein Resource (UniProt). *Nucleic Acids Res* 2014;**42**:D191–8.
19. Mirabello C, Pollastri G. Porter, PaleAle 4.0: high-accuracy prediction of protein secondary structure and relative solvent accessibility. *Bioinformatics* 2013;**29**:2056–8.
20. Hobohm U, Scharf M, Schneider R, *et al*. Selection of representative protein data sets. *Protein Sci* 1992;**1**:409–17.
21. Arlot S, Celisse A. A survey of cross-validation procedures for model selection. *Stat Surv* 2010;**4**:40–79.
22. Stone M. Cross-validatory choice and assessment of statistical predictions. *J R Stat Soc Ser B Methodol* 1974;**36**:111–47.
23. Li W, Godzik A. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics* 2006;**22**:1658–9.
24. Altschul SF, Madden TL, Schäffer AA, *et al*. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* 1997;**25**:3389–402.
25. Wang G, Dunbrack RL. PISCES: a protein sequence culling server. *Bioinformatics* 2003;**19**:1589–91.
26. Sirocco F, Tosatto SCE. TESE: generating specific protein structure test set ensembles. *Bioinformatics* 2008;**24**:2632–3.
27. Finn RD, Bateman A, Clements J, *et al*. Pfam: the protein families database. *Nucleic Acids Res* 2014;**42**:D222–30.
28. Rost B. PHD: predicting one-dimensional protein structure by profile-based neural networks. *Methods Enzymol* 1996;**266**:525–39.
29. Ward JJ, Sodhi JS, McGuffin LJ, *et al*. Prediction and functional analysis of native disorder in proteins from the three kingdoms of life. *J Mol Biol* 2004;**337**:635–45.
30. Chothia C, Lesk AM. The relation between the divergence of sequence and structure in proteins. *EMBO J* 1986;**5**:823–6.
31. Schwartz D, Chou MF, Church GM. Predicting protein post-translational modifications using meta-analysis of proteome scale data sets. *Mol Cell Proteomics* 2009;**8**:365–79.
32. Chen H, Xue Y, Huang N, *et al*. MeMo: a web tool for prediction of protein methylation modifications. *Nucleic Acids Res* 2006;**34**:W249–53.
33. Shawe-Taylor J, Cristianini N. *Kernel Methods for Pattern Analysis*. 2004, Cambridge University Press, Cambridge, UK.
34. Biau G. Analysis of a random forests model. *J Mach Learn Res* 2012;**13**:1063–95.
35. Alpaydin E. *Introduction to Machine Learning*. 2004.
36. Guyon I, Elisseeff A. An introduction to variable and feature selection. *J Mach Learn Res* 2003;**3**:1157–82.
37. Saeys Y, Inza I, Larrañaga P. A review of feature selection techniques in. *Bioinformatics* 2007;**23**:2507–17.
38. Jain A, Zongker D. Feature selection: evaluation, application, and small sample performance. *IEEE Trans Pattern Anal Mach Intell* 1997;**19**:153–8.
39. Varma S, Simon R. Bias in error estimation when using cross-validation for model selection. *BMC Bioinformatics* 2006;**7**:91.
40. Fawcett T. An introduction to ROC analysis. *Pattern Recogn Lett* 2006;**27**:861–74.
41. Dietterich TG. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput* 1998;**10**:1895–923.
42. Bujnicki JM, Elofsson A, Fischer D, *et al*. Structure prediction meta server. *Bioinformatics* 2001;**17**:750–1.
43. Albrecht M, Tosatto SCE, Lengauer T, *et al*. Simple consensus procedures are effective and sufficient in secondary structure prediction. *Protein Eng* 2003;**16**:459–62.
44. Benkert P, Schwede T, Tosatto SC. QMEANclust: estimation of protein model quality by combining a composite scoring function with structural density information. *BMC Struct Biol* 2009;**9**:35.
45. Bernsel A, Viklund H, Hennerdal A, *et al*. TOPCONS: consensus prediction of membrane protein topology. *Nucleic Acids Res* 2009;**37**:W465–8.
46. Plewczynski D, Basu S, Saha I. AMS 4.0: consensus prediction of post-translational modifications in protein sequences. *Amino Acids* 2012;**43**:573–82.