REVIEW

Assessing the Accuracy of Prediction Algorithms for Classification: An Overview

Pierre Baldi *
Dept. of Information and Computer Science
University of California, Irvine
Irvine, CA 92697
pfbaldi@ics.uci.edu
(949) 824-5809
(949) 824-4056 (FAX)

Søren Brunak
Center for Biological Sequence Analysis
The Technical University of Denmark
DK-2800 Lyngby, Denmark
brunak@cbs.dtu.dk
(+45) 45252477
(+45) 45931585 (FAX)

Yves Chauvin
Net-ID, Inc.
San Francisco, CA 94107
yves@netid.com
(415) 647-9402
(415) 642-9265 (FAX)

Claus A. F. Andersen
Center for Biological Sequence Analysis
The Technical University of Denmark
DK-2800 Lyngby, Denmark
ca2@cbs.dtu.dk
(+45) 45252422
(+45) 45931585 (FAX)

Henrik Nielsen
Center for Biological Sequence Analysis
The Technical University of Denmark
DK-2800 Lyngby, Denmark
hnielsen@cbs.dtu.dk
(+45) 45252470
(+45) 45931585 (FAX)

Abstract

We provide a unified overview of methods that currently are widely used to assess the accuracy of prediction algorithms, from raw percentages, quadratic error measures and other distances, correlation coefficients and to information theoretic measures such as relative entropy and mutual information. We briefly discuss the advantages and disadvantages of each approach. For classification tasks, we derive new learning algorithms for the design of prediction systems by directly optimising the correlation coefficient. We observe and prove several results relating sensitivity and specificity of optimal systems. While the principles are general, we illustrate the applicability on specific problems such as protein secondary structure and signal peptide prediction.

*and Department of Biological Sciences, University of California, Irvine. To whom all correspondence should be addressed.

Keywords: prediction, classification, performance measures, secondary structure, signal peptides, correlation, mutual information

1 Introduction and Notation

With the expansion of computer methods in bioinformatics and other fields, researchers are more and more frequently faced with the problem of evaluating the accuracy of a particular prediction algorithm. As several methods aiming at solving the same problem are often available it is also important to be able to select a particular method based on the performance features that can be inferred from the principles that went into its construction. Some methods are optimized such that they produce very few false positives, while others produce very few false negatives, and so on. Normally it is of prime interest to secure, for any type of prediction algorithm, that the method will be able to perform well on novel data that have not been used in the process of constructing the algorithm. That is, the method should be able to generalize to new examples from the same data domain.

A recurrent problem haunting method evaluation is the redundancy of the data: if the sequence examples used for training and testing a particular algorithm are very similar the apparent predictive performance may be overestimated, reflecting the method's ability to reproduce its own input rather than its ability to interpolate and extrapolate. Thus, the actual level of prediction accuracy is intimately related to the degree of similarity between the training and test sets, or in a cross-validated study, to the average degree of pairwise similarity in a data set. Here, however, we shall focus on the definition of relevant criteria for the performance evaluation, and not on issues that relate to the selection of data [22, 11, 17].

While conceptually the evaluation issues are the same for a wide range of different problems, for the sake of concreteness and for historical reasons we shall in this review concentrate on two extensively studied bioinformatics problems: prediction of protein secondary structure and secretory signal peptides.

It is often relevant to measure accuracy of prediction at different levels. For signal peptide prediction, for example, accuracy may be measured by counting how many *sequences* are correctly classified as signal peptides or non-secretory proteins, instead of counting how many residues are correctly predicted to belong to a signal peptide.

At higher levels, however, the measures tend to be more complicated and problem-specific. In the signal peptide example, it is also relevant to ask how many signal peptide sequences have the position of the cleavage site correctly predicted. In gene finding, a predicted exon can have have both ends correct, or only overlap to some extent. Burset and Guigo [5] have defined four simple measures for gene finding accuracy at the exon level — sensitivity, specificity, "missing exons", and "wrong exons" — counting only predictions that are completely correct or completely wrong. For secondary structure prediction, this approach would be too crude, since the borders of structure elements (helices and sheets) are not precisely defined. Instead, the segment overlap measure (SOV) can be applied; it was first introduced by Rost [21] and later slightly modified and applied in the third Critical Assessment of Structure Prediction (CASP3) competition [24]. It is a set of segment-based heuristic evaluation measures, where a correctly predicted segment position can give maximal score even though the prediction is not identical to the assigned segment. The score punishes broken predictions strongly, such as two predicted helices where only one is observed compared to one too small unbroken helix. In this manner the uncertainty of the assignment's exact borders is reflected in the evaluation measure. As this example illustrates, a high-level accuracy measure can become rather *ad hoc* when the precise nature of the prediction problem is taken into consideration.

For the sake of generality, we will therefore focus our attention on single residue/nucleotide assessment measures. For the secondary structure problem, consider an amino acid sequence of length N. The structural data $\mathbf{D}$ available for the comparison is the secondary structure assignments $\mathbf{D} = d_1, \ldots, d_N$. For simplicity, we will first consider the dichotomy problem of two alternative classes, for instance: α-helix versus non-α-helix. In this case, the d_i's are in general equal to 0 or 1. We can also consider the case where d_i has a value between 0 and 1, for example representing the surface exposure of amino acids, or the probability or degree of confidence, reflecting the uncertainty of our knowledge of the correct assignment at the corresponding position. The analysis for the multiple class case, corresponding for example to three states, α-helices, β-sheets and coil, is very similar and will be sketched in a later section. We now assume that our prediction algorithm or model, outputs a prediction of the form $\mathbf{M} = m_1, \ldots, m_N$. In general, m_i is a probability between 0 and 1 reflecting our degree of confidence in the prediction. Discrete 0/1 outputs, obtained for instance by thresholding, or "the-winner-takes-all" approaches, are also possible and fall within the theory considered here. The fundamental and general question we address is how do we assess the accuracy of $\mathbf{M}$, or how do we compare $\mathbf{M}$ to $\mathbf{D}$?

A variety of approaches have been suggested in different contexts and at different times and this may have created some confusion. The issue of prediction accuracy is strongly related to the frequency of occurrence of each class. In protein secondary structure prediction the non-helix class covers roughly 70% of the cases in natural proteins, while only 30% belong to the helix class. Thus a constant prediction of "non-helix" is bound to be correct 70% of the time, although it is highly non-informative and useless.

Thus the purpose of the next section is to review all the approaches and clarify the connections between them and their respective advantages and disadvantages. A fundamental simplifying assumption underlying all these approaches is that the amino acid positions are weighted and treated equally (the independence and equivalence assumption). Thus, we assume e.g. that there is no weighting scheme reducing the influence of positions near the N- or C-termini, or no built-in mechanism that takes into account the fact that particular predictions must vary somewhat "smoothly" (for instance, if a residue belongs to the α-helix category, its neighbors have a slightly higher chance of being also in the α-helix category). Conversely, when predicting functional sites such as intron splice sites, translation start sites, glycosylation or phosphorylation sites, we assume the prediction of a site is either true or false, so that there is no reward for almost correctly placed sites.

Under the independence and equivalence assumption, if both $\mathbf{D}$ and $\mathbf{M}$ are binary, it is clear that their comparison can be entirely summarized by four numbers:

- $TP =$ number of times d_i is helix, m_i is helix (true positive)
- $TN =$ the number of times d_i is non-helix, m_i is non-helix (true negative)
- $FP =$ the number of times d_i is non-helix, m_i is helix (false positive)
- $FN =$ the number of times d_i is helix, m_i is non-helix (false negative)

satisfying: $TP + TN + FP + FN = N$. When $\mathbf{D}$ and/or $\mathbf{M}$ are not binary, then of course the situation is more complex and four numbers do not suffice to summarize the situation. When $\mathbf{M}$ is not binary, binary predictions can still be obtained by using cutoff thresholds. The numbers TP, TN, FP, and FN will then vary with the threshold choice. The numbers TP, TN, FP, and FN are often arranged into a 2×2 contingency matrix:

	$\mathbf{M}$	$\bar{\mathbf{M}}$
$\mathbf{D}$	TP	FN
$\bar{\mathbf{D}}$	FP	TN

Even with four numbers alone, it is not immediately clear how a given prediction method fares. This is why a lot of the comparison methods aim at constructing a single number measuring the "distance" between $\mathbf{D}$ and $\mathbf{M}$. But it must be clear from the outset, that information is always lost in such a process, even in the binary case, i.e. when going from the four numbers above to a single one. In general, several different vectors (TP, TN, FP, FN) will result in the same distance. We now review several ways of measuring the performance of $\mathbf{M}$ and their merits and pitfalls.

2 Performance Measures

2.1 Percentages

The first obvious approach is to use percentages derived from TP, TN, FP, and FN. For instance, Chou and Fasman [6, 7] used the percentage of correctly predicted helices

$$PCP(\mathbf{D}, \mathbf{M}) = 100 \frac{TP}{TP + FN}. \tag{1}$$

which is the same as the sensitivity (see section 2.9) expressed as a percentage. This number alone provides no information whatsoever about false positives. It can be complemented by the percentage of correctly predicted non-helices

$$PCN(\mathbf{D}, \mathbf{M}) = 100 \frac{TN}{TN + FP}. \tag{2}$$

The average of the previous two numbers has been used in the literature [6, 7] and is often called Q_α. While Q_α is a useful indicator, it can be misleading [23] and can be computed only if both $\mathbf{D}$ and $\mathbf{M}$ are binary. Intuitively, any single number that is constructed using only two numbers out of the four (TP, TN, FP, FN) is bound to be highly biased in some trivial way. If the two numbers are TP and FP for instance, then any two situations (TP, TN, FP, FN) and (TP, TN', FP, FN') lead to the same score regardless of how different they may be.

2.2 Hamming Distance

In the binary case, the Hamming distance between $\mathbf{D}$ and $\mathbf{M}$ is defined by

$$HD(\mathbf{D}, \mathbf{M}) = \sum_i |d_i - m_i|. \tag{3}$$

This sum is obviously equal to the total number of errors $FP + FN$. Thus it is equivalent to a single percentage measure. This distance does not take into account the proportion of examples that belong to a given class. It becomes less and less useful as this proportion moves away from 50%. In the non purely binary case, the Hamming distance is called the L^1 distance and is discussed below.

2.3 Quadratic "Distance"

The quadratic or Euclidean or LMS (least means square) "distance" is defined by

$$Q(\mathbf{D}, \mathbf{M}) = (\mathbf{D} - \mathbf{M})^2 = \sum_i (d_i - m_i)^2. \tag{4}$$

Strictly speaking, a proper distance is defined by taking the square root of the above quantity (see the L^2 distance below). In the purely binary case, the quadratic distance reduces to the Hamming distance and is again equal to $FP + FN$. This quantity has the advantage of being defined for non-binary variables, and it is often associated with a negative log-likelihood approach for a Gaussian model of the form

$$P(d_i|m_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(d_i - m_i)^2/2\sigma^2} \tag{5}$$

where σ acts as a scaling factor with respect to $Q(\mathbf{D}, \mathbf{M})$. For binary variables, the quadratic distance is identical to the Hamming distance. The main drawback is that the Gaussian model is often not relevant for prediction problems and the value of the quadratic distance again poorly reflects the proportion of positions that belongs to a given class. Another problem discussed later in the paper is that the LMS distance has a limited dynamic range due to the fact that m_i and d_i are between 0 and 1. This is not ideal for learning algorithms where large error signals can be used to accelerate the learning process. A logarithmic variation on the LMS distance discussed in the appendix that obviates this problem is given by

$$LQ(\mathbf{D}, \mathbf{M}) = -\sum_i \log[1 - (d_i - m_i)^2]. \tag{6}$$

This modified error function has been used in several neural network implementations, see for example [4, 10, 9].

2.4 L^p Distances

More generally, the L^P distance is defined by

$$LP(\mathbf{D}, \mathbf{M}) = [\sum_i |d_i - m_i|^p]^{1/p} \tag{7}$$

Such distance applies of course to any numerical values. When $p = 1$ we find the Hamming distance, and when $p = 2$ we find the proper Euclidean distance. When $p \to \infty$, the L^∞ distance is the sup distance: $\max_i |d_i - m_i|$. This distance provides an upper bound associated with the worst case, but is not very useful in assessing the performance of a protein secondary structure prediction algorithm. Other values of p are rarely used in practice, and are of little help for assessing prediction performance in this context. In the binary case, the L^p distance reduces to $(FP + FN)^{1/p}$. For $p = 1$, this reduces again to the Hamming distance.

2.5 Correlation

One of the standard measures used by statisticians is the correlation coefficient also called the Pearson correlation coefficient

$$C(\mathbf{D}, \mathbf{M}) = \sum_i \frac{(d_i - \bar{d})(m_i - \bar{m})}{\sigma_{\mathbf{D}} \sigma_{\mathbf{M}}} \tag{8}$$

where $\bar{d} = \sum d_i/N$ and $\bar{m} = \sum m_i/N$ are the averages, and $\sigma_{\mathbf{D}}, \sigma_{\mathbf{M}}$ the corresponding standard deviations. In the context of secondary structure prediction, this is also known as the Matthews correlation coefficient in the literature since it was first used in [16]. The correlation coefficient is always between -1 and $+1$ and can be used with non-binary variables. It is a measure of how the normalized variables $(d_i - \bar{d})/\sigma_{\mathbf{D}}$ and $(m_i - \bar{m})/\sigma_{\mathbf{M}}$ tend to have the same sign and magnitude. A value of -1 indicates total disagreement and $+1$ total agreement. The correlation coefficient is 0 for completely random predictions. Therefore it yields easy comparison with respect to a random baseline. If two variables are independent, then their correlation coefficient is 0. The converse in general is not true.

In vector form, the correlation coefficient can be rewritten as a dot product between normalized vectors

$$C(\mathbf{D}, \mathbf{M}) = \frac{(\mathbf{D} - \bar{d}\mathbf{1})(\mathbf{M} - \bar{m}\mathbf{1})}{\sqrt{(\mathbf{D} - \bar{d}\mathbf{1})^2}\sqrt{(\mathbf{M} - \bar{m}\mathbf{1})^2}} = \frac{\mathbf{DM} - N\bar{d}\bar{m}}{\sqrt{(\mathbf{D}^2 - N\bar{d}^2)(\mathbf{M}^2 - N\bar{m}^2)}} \tag{9}$$

where $\mathbf{1}$ denotes the N-dimensional vector of all ones. As such, $C(\mathbf{D}, \mathbf{M})$ is related to the L^2 distance, but is not a distance itself since it can assume negative values. If the vectors $\mathbf{D}$ and $\mathbf{M}$ are normalized, then clearly $Q(\mathbf{D}, \mathbf{M}) = (\mathbf{D} - \mathbf{M})^2 = 2 - 2\mathbf{DM} = 2 - 2C(\mathbf{D}, \mathbf{M})$. Unlike some of the previous measures, the correlation coefficient has a global form rather than being a sum of local terms.

In the case where $\mathbf{D}$ and $\mathbf{M}$ consist of binary 0/1 vectors, we have $\mathbf{D}^2 = TP + FN$, $\mathbf{M}^2 = TP + FP$, $\mathbf{DM} = TP$, etc. With some algebra the sum above can be written as

$$C(\mathbf{D}, \mathbf{M}) = \frac{TP - N\bar{d}\bar{m}}{N\sqrt{\bar{d}\bar{m}(1 - \bar{d})(1 - \bar{m})}} \tag{10}$$

Here the average number of residues in the helix class satisfies $\bar{d} = (TP + FN)/N$, and similarly for the predictions $\bar{m} = (TP + FP)/N$. Therefore

$$\begin{aligned} C(\mathbf{D}, \mathbf{M}) &= \frac{N \times TP - (TP + FN)(TP + FP)}{\sqrt{(TP + FN)(TP + FP)(TN + FP)(TN + FN)}} \\ &= \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FN)(TP + FP)(TN + FP)(TN + FN)}} \end{aligned} \tag{11}$$

The correlation coefficient uses all four numbers (TP, TN, FP, FN) and may often provide a much more balanced evaluation of the prediction than for instance the percentages. There are situations, however, where even the correlation coefficient is unable to provide a completely fair assessment. The correlation coefficient will, for example, be relatively high in cases where a prediction algorithm gives very few or no false positives, but at the same time very few true positives. One simple observation that will be useful in a later section is that C is symmetric with respect to FP and FN.

One interesting property of the correlation coefficient is that there is a simple approximate statistical test for deciding whether it is significantly better than zero, i.e. whether the prediction is significantly more correlated with the data than a random guess with the same $\bar{m}$ would be. If the Chi-squared test is applied to the 2×2 contingency matrix containing TP, TN, FP, and FN to decide, it is easy to show that the test statistic is $\chi^2 = N \times C^2(\mathbf{D}, \mathbf{M})$.

2.6 Approximate Correlation

Burset and Guigó [5] defined an "approximate correlation" measure to compensate for an alleged problem with the Matthews correlation coefficient: that it is not defined when any of the sums $TP+FN$, $TP+FP$, $TN+FP$, or $TN+FN$ reaches zero, e.g. if there are no positive predictions. Instead, they use the Average Conditional Probability (ACP) which is defined as

$$ACP = \frac{1}{4}\left[\frac{TP}{TP+FN} + \frac{TP}{TP+FP} + \frac{TN}{TN+FP} + \frac{TN}{TN+FN}\right] \tag{12}$$

if all the sums are non-zero; otherwise, it is the average over only those conditional probabilities that are defined. Approximate Correlation (AC) is a simple transformation of the ACP:

$$AC = 2 \times (ACP - 0.5). \tag{13}$$

Like C, AC gives 1, 0, and -1 for perfect, random, and all-false predictions, respectively; and Burset and Guigó observe that it is close to the real correlation value.

However, the problem that they intend to solve does not exist, since it is easy to show that C approaches 0 if any of the sums approaches 0. This also makes intuitive sense, since a prediction containing only one category is meaningless and does not convey any information about the data. On the contrary, it can be shown that the AC approach introduces an unfortunate discontinuity in this limit because of the deletion of undefined probabilities from the expression for ACP, so it does *not* give 0 for meaningless predictions. Since there is furthermore no simple geometrical interpretation for AC, it is an unnecessary approximation and we see no reason to encourage its use.

2.7 Relative Entropy

The relative entropy, or cross entropy, or KL (Kullback-Leibler) contrast between two probability vectors $\mathbf{X} = (x_1, \ldots, x_M)$ and $\mathbf{Y} = (y_1, \ldots, y_M)$ with $x_i, y_i \geq 0$ and $\sum x_i = \sum y_i = 1$ is defined as [14, 13, 2]

$$H(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{M} x_i \log \frac{x_i}{y_i} = -H(\mathbf{X}) - \sum_i x_i \log y_i \tag{14}$$

where $H(\mathbf{X}) = -\sum x_i \log x_i$ is the usual entropy. It has its roots in information theory [2]. It is well known that $H(\mathbf{X}, \mathbf{Y})$ is always positive, convex in both its variables, and equal to 0 if and only if $\mathbf{X} = \mathbf{Y}$. Strictly speaking it is not a distance, for instance because it is not symmetric. It is easy to construct a distance using a symmetrized version. In practice, this is rarely necessary and the form above is sufficient. If $\mathbf{Y} = \mathbf{X} + \epsilon$ is close to

$\mathbf{X}$, then a simple Taylor expansion shows that

$$H(\mathbf{X}, \mathbf{X}+\epsilon) = -\sum_i x_i \left[\log(1+\frac{\epsilon_i}{x_i})\right] \approx \sum_i \frac{\epsilon_i^2}{x_i} \tag{15}$$

In particular, if $\mathbf{X}$ is uniform, then in its neighborhood the relative entropy behaves like the LMS error.

Returning to the secondary structure prediction problem, we can then assess the performance of the prediction $\mathbf{M}$ by the quantity:

$$H(\mathbf{D}, \mathbf{M}) = \sum_{i=1}^{N} \left[d_i \log \frac{d_i}{m_i} + (1-d_i) \log \frac{(1-d_i)}{(1-m_i)}\right] \tag{16}$$

This is just the sum of the relative entropies at each position i. This form of course works perfectly well on non-binary data (for example binding affinities), or when $\mathbf{D}$ alone is binary. When $\mathbf{M}$ is also binary, then the relative entropy has $FP+FN$ components which are infinite (it behaves like $H(\mathbf{D}, \mathbf{M}) \approx (FP+FN)\infty$) and cannot really be used.

2.8 Mutual Information

Consider two random variables $\mathcal{X}$ and $\mathcal{Y}$ with probability vectors $\mathbf{X} = (x_1, \ldots, x_M)$ and $\mathbf{Y} = (y_1, \ldots, y_K)$. Let $\mathcal{Z}$ be the joint random variable $\mathcal{Z} = (\mathcal{X}, \mathcal{Y})$ over the cartesian product with probability vector $\mathbf{Z}$. The mutual information $I(\mathcal{X}, \mathcal{Y})$ or $I(\mathbf{X}, \mathbf{Y})$ between $\mathcal{X}$ and $\mathcal{Y}$ is defined as the relative entropy between $\mathcal{Z}$ and the product $\mathcal{X}\mathcal{Y}$

$$I(\mathcal{X}, \mathcal{Y}) = H(\mathcal{Z}, \mathcal{X}\mathcal{Y}) \tag{17}$$

As such it is always positive. It is easy to understand the mutual information in Bayesian terms: it represents the reduction in uncertainty of one variable when the other is observed, that is between the *prior and posterior distributions* [2]. The uncertainty in $\mathcal{X}$ is measured by the entropy of its prior $H(\mathcal{X}) = \sum x_i \log x_i$. Once we observe $\mathcal{Y} = y$, the uncertainty in $\mathcal{X}$ is the entropy of the posterior distribution, $H(\mathcal{X}|\mathcal{Y} = y) = \sum_x P(\mathcal{X} = x|\mathcal{Y} = y) \log P(\mathcal{X} = x|\mathcal{Y} = y)$. This is a random variable that depends on the observation y. Its average over the possible ys is called the *conditional entropy*

$$H(\mathcal{X}|\mathcal{Y}) = \sum_y P(y) H(\mathcal{X}|\mathcal{Y} = y). \tag{18}$$

Therefore the difference between the entropy and the conditional entropy measures the average information that an observation of $\mathcal{Y}$ brings about $\mathcal{X}$. It is straightforward to check that

$$I(\mathcal{X}, \mathcal{Y}) = H(\mathcal{X}) - H(\mathcal{X}|\mathcal{Y}) = H(\mathcal{Y}) - H(\mathcal{Y}|\mathcal{X}) = H(\mathcal{X}) + H(\mathcal{Y}) - H(\mathcal{Z}) = I(\mathcal{Y}, \mathcal{X}) \tag{19}$$

or, using the corresponding distributions,

$$I(\mathbf{X}, \mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X}|\mathbf{Y}) = H(\mathbf{Y}) - H(\mathbf{Y}|\mathbf{X}) = H(\mathbf{X}) + H(\mathbf{Y}) - H(\mathbf{Z}) = I(\mathbf{Y}, \mathbf{X}). \tag{20}$$

Going back to the secondary structure problem, when $\mathbf{D}$ and $\mathbf{M}$ are both binary, the mutual information is measured by

$$\begin{aligned} I(\mathbf{D},\mathbf{M}) &= -H\left(\frac{TP}{N},\frac{TN}{N},\frac{FP}{N},\frac{FN}{N}\right) \\ &\quad -\frac{TP}{N}\log\left[\frac{TP+FP}{N}\frac{TP+FN}{N}\right]-\frac{FN}{N}\log\left[\frac{TP+FN}{N}\frac{TN+FN}{N}\right] \\ &\quad -\frac{FP}{N}\log\left[\frac{TP+FP}{N}\frac{TN+FP}{N}\right]-\frac{TN}{N}\log\left[\frac{TN+FN}{N}\frac{TN+FP}{N}\right] \end{aligned} \tag{21}$$

or

$$\begin{aligned} I(\mathbf{D},\mathbf{M}) &= -H\left(\frac{TP}{N},\frac{TN}{N},\frac{FP}{N},\frac{FN}{N}\right) \\ &\quad -\frac{TP}{N}\log[\bar{d}\bar{m}]-\frac{FN}{N}\log[\bar{d}(1-\bar{m})] \\ &\quad -\frac{FP}{N}\log[(1-\bar{d})\bar{m}]-\frac{TN}{N}\log[(1-\bar{d})(1-\bar{m})] \end{aligned} \tag{22}$$

(see also [23]), where $\bar{d}=(TP+FN)/N$ and $\bar{m}=(TP+FP)/N$ (as before), and

$$H(\frac{TP}{N},\frac{TN}{N},\frac{FP}{N},\frac{FN}{N})=-\frac{TP}{N}\log\frac{TP}{N}-\frac{TN}{N}\log\frac{TN}{N}-\frac{FP}{N}\log\frac{FP}{N}-\frac{FN}{N}\log\frac{FN}{N} \tag{23}$$

is the usual entropy. Like the correlation, the mutual information is a global measure rather than a sum of local terms. It is clear that the mutual information always satisfies $0\leq I(\mathbf{D},\mathbf{M})\leq H(\mathbf{D})$. Thus in the assessment of prediction performance, it is customary to use the normalized mutual information [20, 21] coefficient

$$IC(\mathbf{D},\mathbf{M})=\frac{I(\mathbf{D},\mathbf{M})}{H(\mathbf{D})} \tag{24}$$

with

$$H(\mathbf{D})=-\frac{TP+FN}{N}\log\left[\frac{TP+FN}{N}\right]-\frac{TN+FP}{N}\log\left[\frac{TN+FP}{N}\right] \tag{25}$$

or, more briefly expressed, $H(\mathbf{D})=-\bar{m}\log\bar{m}-(1-\bar{m})\log(1-\bar{m})$. The normalized mutual information satisfies $0\leq IC(\mathbf{D},\mathbf{M})\leq 1$. When $IC(\mathbf{D},\mathbf{M})=0$, then $I(\mathbf{D},\mathbf{M})=0$ and the prediction is totally random ($\mathbf{D}$ and $\mathbf{M}$ are independent). When $IC(\mathbf{D},\mathbf{M})=1$, then $I(\mathbf{D},\mathbf{M})=H(\mathbf{D})=H(\mathbf{M})$ and the prediction is perfect. Like the correlation coefficient, the mutual information coefficient is a global measure rather than a sum of local terms. The mutual information is symmetric in FP and FN, but the mutual information coefficient is *not* because of its denominator.

2.9 ROC Curves, Sensitivity and Specificity

In a two-class prediction case where the output of the prediction algorithm is continuous, the numbers TP, TN, FP and FN depend on how the threshold is selected. Generally, there is a tradeoff between the amount of false positives and the amount of false negatives produced by the algorithm. ROC (receiver operating characteristics) summarize such results by displaying for threshold values within a certain range the "hit rate" (sensitivity, $TP/(TP+FN)$) versus the "false alarm rate" (also known as false positive rate, $FP/(FP+TN)$. Typically the hit rate increases with the false alarm rate (see Figure 1).

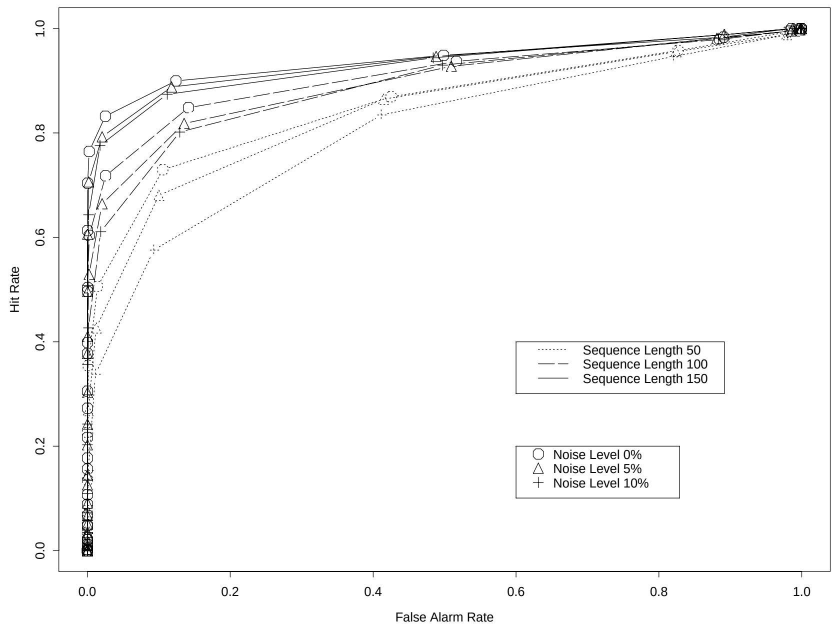

Figure 1: ROC curves for the detection of protein fragments as a function of the length and noise level of the fragments (see [2] for details).

Alternatively, one can also display the sensitivity $(TP/(TP+FN))$ versus the specificity $(TP/(TP+FP))$ in a similar plot or separately as a function of threshold in two different plots. An example will be given below.

While the sensitivity is the probability of correctly predicting a positive example, the specificity as defined above is the probability that a positive prediction is correct. In medical statistics, the word "specificity" is sometimes used in a different sense, meaning the chance of correctly predicting a negative example: $TN/(FP+TN)$, or 1 minus the false positive rate (see e.g. [5]). We would prefer to refer to this as the sensitivity of the negative category.

If we write $x = TP/(TP+FN)$ for the sensitivity and $y = TP/(TP+FP)$ for the specificity, then:

$$TP+FP = \frac{TP}{y} \qquad TP+FN = \frac{TP}{x}$$
$$TN+FP = N-(TP+FN) = \frac{Nx-TP}{x}$$
$$TN+FN = N-(TP+FP) = \frac{Ny-TP}{y} \tag{26}$$

provided $x \neq 0$ and $y \neq 0$, which is equivalent to $TP \neq 0$, a rather trivial case. In other words, we just reparameterize (TP, TN, FP, FN) using (TP, x, y, N). In this form, it is clear that we can substitute these values in Equation 11 to derive, after some algebra, an expression for the correlation coefficient as a function of the specificity and the sensitivity:

$$C(\mathbf{D}, \mathbf{M}) = \frac{Nxy - TP}{\sqrt{(Nx - TP)(Ny - TP)}} \tag{27}$$

Notice that this expression is entirely symmetric in x and y, i.e. in the specificity and sensitivity, or equivalently also in FP and FN, the number of false positive and false negative. In fact, for a given TP, exchanging FP and FN is equivalent to exchanging x and y. A similar calculation can be done in order to re-express the mutual information of Equation 22 or the mutual information coefficient of Equation 24 in terms of TP, x, y, and N. The mutual information is entirely symmetric in x and y, or FP and FN (this is not true of the mutual information coefficient).

2.10 Summary

In summary, under the equivalence and independence assumption, if both $\mathbf{D}$ and $\mathbf{M}$ are binary, then all the performance information is contained in the numbers TP, TN, FP, and FN. Any measure of performance using a single number discards some information. The Hamming distance and the quadratic distance are identical. These distances, as well as the percentages and the L^p distances are based on only two out of the four numbers TP, TN, FP, and FN. The correlation coefficient or the mutual information coefficient are based on all four parameters and provide a better summary of performance in this case. In the continuous case, the recommended measures are the correlation coefficient and the relative entropy.

3 More Than Two Classes

In the case of a multi-class prediction problem with K classes, one obtains a $K \times K$ contingency or confusion matrix $\mathbf{Z} = (z_{ij})$. Thus in the secondary structure prediction case, one often has to deal with a 3×3 contingency matrix associated with the three basic classes: helix H, sheet E, and coil C. The number z_{ij} represents the number of times the input is predicted to be in class j while belonging in reality to class i. Naturally, if the prediction algorithms outputs continuous quantities, it is clear that this definition assumes thresholding has already taken place. Even more so than in the binary case, it is difficult to collapse the contingency matrix into a single number. The number of inputs associated with class i is given by $x_i = \sum_j z_{ij}$. Likewise, the number of inputs predicted to be in class i is given by $y_i = \sum_j z_{ji}$. Obviously $N = \sum_{ij} z_{ij} = \sum_i x_i = \sum_i y_i$.

It should be fairly clear which measures can be generalized to the multiple class case and how to do so. Here we shall work out the details only for the percentages and the mutual information. We can define the percentage

$$Q_i = Q_i^{\mathbf{D}} = 100 \frac{z_{ii}}{x_i} \tag{28}$$

which captures the percentage of inputs correctly predicted to belong to class i relative to the total number of inputs in i (sensitivity for class i). One can similarly define another percentage

$$Q_i^{\mathbf{M}} = 100\frac{z_{ii}}{y_i} \tag{29}$$

which captures the number of inputs correctly predicted to be in i with respect to the total number of inputs predicted to be in i (specificity for class i). This provides an estimate of the conditional probability of correct prediction, given that the predicted class is i. Finally, the overall percentage is often used in the literature

$$Q_{total} = 100\frac{\sum_i z_{ii}}{N} \tag{30}$$

This is just the percentage of all correct predictions. As in the the two-class case, these percentages often hide important information. For instance, individual class frequencies are completely absent from Q_{total}. In the case of secondary structure prediction, for instance, one can also compute slightly different percentages defined on a per protein chain basis.

To derive the mutual information or mutual information coefficient, it is sufficient to apply the definition using $\mathbf{X} = (x_i/N)$, $\mathbf{Y} = (y_i/N)$ and $\mathbf{Z} = (z_{ij}/N)$. For instance,

$$IC(\mathbf{D},\mathbf{M}) = \frac{H(\mathbf{X}) + H(\mathbf{Y}) - H(\mathbf{Z})}{H(\mathbf{X})} = \frac{-\sum_i \frac{x_i}{N}\log\frac{x_i}{N} - \sum_i \frac{y_i}{N}\log\frac{y_i}{N} + \sum_{ij}\frac{z_{ij}}{N}\log\frac{z_{ij}}{N}}{-\sum_i \frac{x_i}{N}\log\frac{x_i}{N}} \tag{31}$$

The mutual information may also be expressed as a sum of the information contribution for each assignment category [23]:

$$I(\mathbf{D},\mathbf{M}) = \sum_i I_i^{\mathbf{D}} \tag{32}$$

where $I_i^{\mathbf{D}}$ is the information provided by the prediction about the category i:

$$I_i^{\mathbf{D}} = -\frac{x_i}{N}\log\frac{x_i}{N} + \sum_j \frac{z_{ij}}{N}\log\frac{z_{ij}}{y_j}. \tag{33}$$

The Matthews correlation coefficient as defined in Equation 11 is not readily generalized to more than two classes; but it is possible to use the analogy with the χ^2 statistic to define a generalized measure for K classes which is equal to $C^2(\mathbf{D},\mathbf{M})$ when $K = 2$. We can refer to this as a *generalized squared correlation*, GC^2:

$$GC^2(\mathbf{D},\mathbf{M}) = \frac{\sum_{ij}\frac{(z_{ij}-e_{ij})^2}{e_{ij}}}{N(K-1)} \tag{34}$$

where $e_{ij} = x_i y_j/N$ is the expected number of data in cell i,j of the contingency matrix under the null hypothesis assumption that there is no correlation between assignments and predictions. Like IC, GC^2 ranges between 0 and 1.

3.1 Example: Secondary Structure prediction

As an illustration of the application of performance measures on more than two classes, we show in Table 1 a comparison of two neural networks for prediction of protein secondary structure [1] with three "non-informative" predictors.

	Q_{total}	I	IC	GC^2	$Q_i^{\mathbf{D}}$ H/E/C	$Q_i^{\mathbf{M}}$ H/E/C	$I_i^{\mathbf{D}}$ H/E/C	C_i H/E/C
NN Balanced	0.6714	0.2116	0.2024	0.2328	0.6717 0.5821 0.7110	0.6899 0.5319 0.7290	0.0895 0.0607 0.0614	0.5389 0.4304 0.4710
NN Unbalanced	0.6508	0.1653	0.1582	0.1811	0.6433 0.3995 0.7674	0.6330 0.5581 0.6879	0.0703 0.0383 0.0567	0.4723 0.3583 0.4508
Random 1/3	0.3333	0	0	0	0.3333 0.3333 0.3333	0.3118 0.2117 0.4765	0 0 0	0 0 0
Random BG	0.3691	0	0	0	0.3118 0.2117 0.4765	0.3118 0.2117 0.4765	0 0 0	0 0 0
Only coil	0.4765	0	0	0	0 0 1	– – 0.4765	0 0 0	0 0 0

Table 1: Prediction performances for two neural networks (using balanced and unbalanced training) and three "non-informative" predictors (random guess with uniform or background probabilities and coil-only prediction). The performances are assessed with four different 3-category measures (overall % correct, Q_{total}; mutual information, I; mutual information coefficient, IC; and generalised squared correlation, GC^2) and four different 2-category measures (sensitivity, $Q_i^{\mathbf{D}}$; specificity, $Q_i^{\mathbf{M}}$; mutual information contribution, $I_i^{\mathbf{D}}$; and correlation coefficient, C_i) applied separately to each class (α-helix, H; extended β-sheet, E; and coil, C).

The neural networks have one hidden layer with 200 hidden units and were trained using standard backpropagation on the relative entropy (Equation 16). The secondary structure was predicted from the amino acid sequence based on the DSSP [12] definitions of α-helix (H), extended β-sheet (E) and coil (C). One of the networks used *balanced* training: at each training cycle, only a subset of the examples was shown so that the three classes were presented with equal weights. For both the balanced and unbalanced network, we have selected the training cycle with the optimal Q_{total}.

The comparison between the balanced and unbalanced network illustrates how seemingly similar Q_{total} values for a prediction can hide significant underlying differences. There is only 2% difference in their Q_{total} values; but the sensitivities $Q_i^{\mathbf{D}}$ show that the unbalanced network vastly overpredicts coil (C) at the expence of β-sheet (E). The generalised correlation and the information-based measures (I, IC, and GC^2) reflect this difference much more substantially than Q_{total}.

For the two-category measures, another difference is worth noting: judged by the sensitivity and specificity measures ($Q_i^{\mathbf{D}}$ and $Q_i^{\mathbf{M}}$), coil seems to be the most easily predicted category. This, however, again merely reflects that coil is more abundant than the two secondary structure classes; and the $I_i^{\mathbf{D}}$ and C_i measures show a lower precision for coil than for α-helix. If the balanced neural network is optimized with respect to one of the information based measures instead of Q_{total} (results not shown), the information contribution $I_i^{\mathbf{D}}$ shows coil to be even worse predicted than β-sheets, while the correlation coefficient C_i is always higher for coil than for β-sheets. This discrepancy has also been noted by Wang [23].

For comparison, we have included three "predictions" without any information content in the table. The "Random 1/3" predictor makes a random guess giving each category the same probability and therefore gets 1/3 of the predictions right. If the background distributions are used as guessing probabilities ("Random BG"), 37% will be correctly predicted. Finally, if the predictor always returns the largest category ("Only coil") a Q_{total} of 48% can be achieved. Note that all of these "non-informative" predictors achieve zero by the measures I, IC and GC^2.

4 Probabilistic Models and Learning

It is important to realize that several of the error functions described above come with a natural underlying probabilistic model [2] and this impacts the parameterization of the prediction models, as in the case of a neural network. For simplicity of notation here we consider the case of a single prediction m for a single pattern d. If d has a wide range, then the output transfer function of the network should not be sigmoidal but rather linear. Under a Gaussian noise model, the likelihood is given by $P(d|m) = Z \exp(-(d-m)^2/2s)$, where Z is a normalizing constant. The negative log-likelihood is obviously the LMS error function, up to a scaling factor provided by the standard deviation s. In the case of 0/1 classification, we can consider a simple binomial model for the data. In this case the likelihood is $P(d|m) = m^d(1-m)^{1-d}$. The output transfer function should be the logistic function $m = f(u) = 1/(1+e^{-u})$ (see also Appendix). The negative log-likelihood is essentially the relative entropy $H = d \log m + (1-d)\log(1-m)$. Finally, in the case of a classification into K classes $m = (m_1, \ldots, m_K)$ and $d = (d_1, \ldots, d_K)$, where all the d_i are 0 except one of them which is equal to 1. The simple multinomial model yields the likelihood $P(d|m) = \prod_i m_i^{d_i}$ and the log-likelihood is again the relative entropy $H = \sum_i d_i \log(m_i)$, except for a constant offset term. In this case, the output transfer functions in the output layer of the neural networks should be the normalized exponentials $m_i = \exp u_i / \sum_k \exp u_k$ which reduce to the logistic function when $K = 2$. Needless to say, in all the cases we could also included a prior $P(w)$ in the model and maximize the posterior rather than the likelihood, as in the well-known weight decay neural network approach.

The choice of a performance evaluator is particularly important when one considers machine learning approaches to prediction such as neural networks. Often in these approaches a model with many parameters is selected and then fitted to the training data by some performance optimisation algorithms, such as gradient descent. It is clear that the resulting algorithm depends to a great extent on the measure that has been optimised during the learning from examples procedure. A related aspect has to do with the difference between confidence and classification. Consider for instance the prediction of four α-helical residues. With a threshold at 0.5, the prediction $(.6, .6, .6, .6)$ gives perfect classification, but the confidence is rather poor. The prediction vector $(0.9, 0.9, 0.9, 0.4)$ classifies correctly only three out of the four residues, but with much higher confidence.

Another issue has to do with dynamic range. As pointed out in [2] and above, there is a number of theoretical reasons why the output of a neural network binary classifier should be implemented by a logistic sigmoidal unit trained using the relative entropy error measure rather than the usual LMS error. Likewise, in a multi-class classification problem, the output of the network should be implemented by a set of normalized exponential units (the so-called "soft max" units) with the relative entropy error. Reasonable results have been obtained in these cases, however, also with the standard LMS error. In fact, in simple cases [2] both error functions lead to the same global optimum. But the dynamic range is very different and can impact the learning process. The LMS error is much more shallow. This is because for each i the LMS error is limited to a narrow range whereas the relative entropy is potentially unbounded. This can lead to larger gradients and potentially better or faster learning. In the Appendix, we compare the relative entropy and the logarithmic LMS error—which is also unbounded— from this perspective.

Some of the coefficients above are additive with respect to single predictions. This is the case of the LMS and the relative entropy error measures. In this case, the global derivative of the error can easily be computed as a sum of the local errors. Gradient descent learning can then be applied "on-line", i.e. after the presentation of each example. Other error measures, such as the correlation coefficient or the mutual information coefficient, do not have this simple additive structure. The derivatives however can still be computed and this suggests a slightly different class of learning algorithms—based on correlation or mutual information maximization—which, to the best of our knowledge, have not been tried on the protein secondary structure prediction problem. The derivatives of the correlation measure are computed in the Appendix. Maximization of mutual information has been used in a different context [15].

5 Experiments and discussion

SignalP [18] is a program that was derived by some of us in order to predict signal peptides and cleavage sites in proteins. In its basic form, it consists of two types of neural networks that are trained to provide two different scores in the 0 to 1 range, for each amino acid in a given sequence, the C and the S score. The score is typically associated with the amino acid located in the central position of a fixed network input window. The S (signal/non-signal) score can be interpreted as an estimate of the probability that the amino acid belongs to a signal peptide. The C score (cleavage/non-cleavage) can be interpreted as an estimate of the probability that the amino acid has the first position in the mature protein (position +1 relative to the cleavage site). One important observation has to do with the skewedness of the corresponding training sets. Cleavage sites are single amino acid events: thus they are quite rare. The cleavage/non-cleavage site problem has a very skewed distribution with very low positive/negative examples ratio. Signal peptides amino acids are more numerous and therefore can lead to a more balanced training set with a positive/negative ratio of examples close to 1.

Here we computed the correlation coefficient, mutual information, sensitivity and specificity of some of these networks. Typical plots are given in the following figures. Available examples can be partitioned into training and test sets in many ways. The plots above are just one representative example of such partition. But in all the training/test set partitions we experimented with a number of phenomena were generally observed:

For a fixed skewed data set (cleavage/non-cleavage), the mutual information and the mutual information coefficient seem to peak at a lower cutoff than the Matthews correlation coefficient (see Figure 2). In addition, for a fixed balanced set (see Figure 3), the mutual information, the information coefficient, and the Matthews correlation coefficient often peak near the point where the sensitivity is equal to the specificity (and often with value close to 0.5). This is not too surprising since the data set being fixed and balanced, the variables TP, TN, FP, and FN are constrained by $TP+FN = N/2$ and $TN+FP = N/2$ for some $0 \leq C \leq N$. Now we have seen that the expressions for $I(\mathbf{D}, \mathbf{M})$, $IC(\mathbf{D}, \mathbf{M})$,

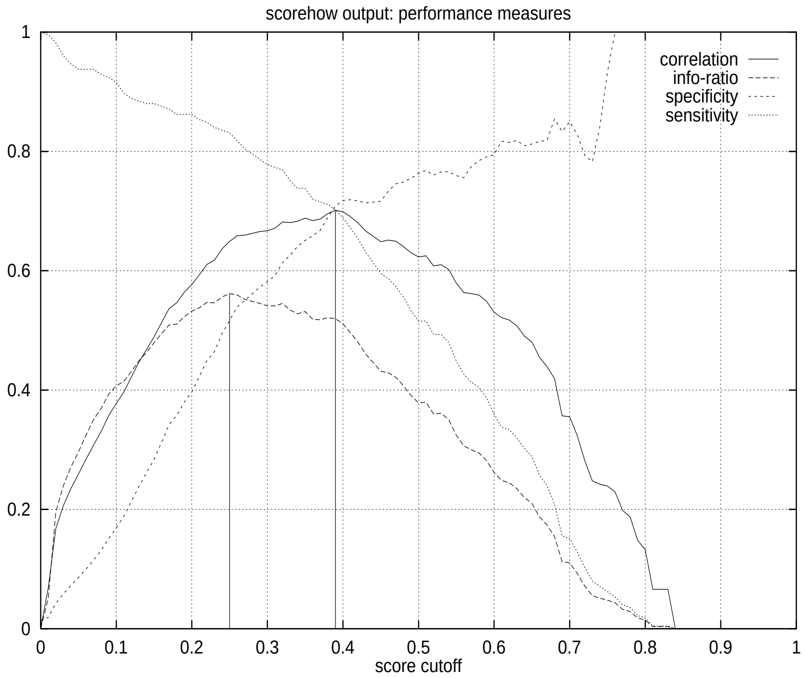

Figure 2: Sensitivity, specificity, (Matthews) correlation coefficient, and mutual information coefficient as a function of the cutoff for assigning a positive prediction for cleavage site/non-cleavage site prediction. The maximum of both coefficients are indicated.

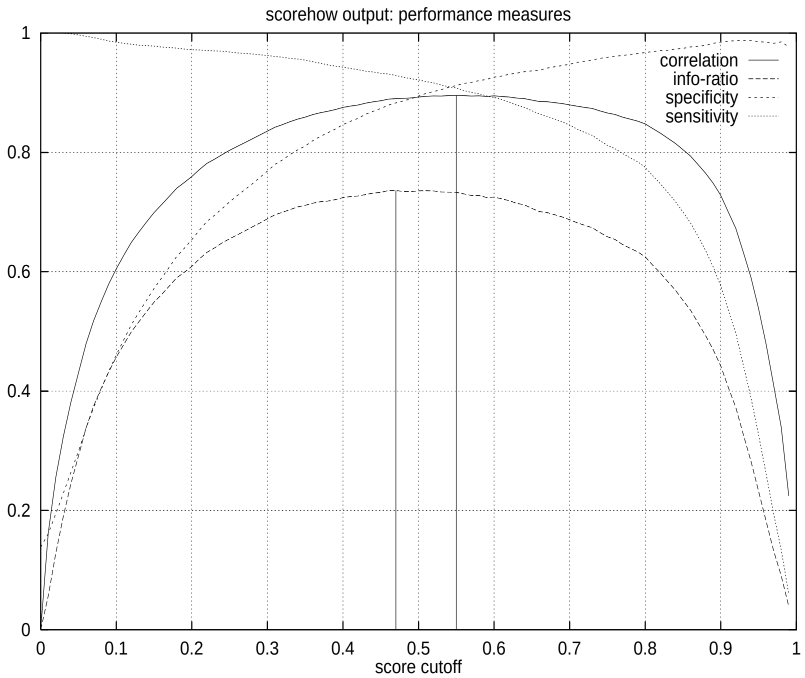

Figure 3: Sensitivity, specificity, (Matthews) correlation coefficient, and mutual information coefficient as a function of the cutoff for assigning a positive prediction for signal peptide/non-signal peptide prediction. The maximum of both coefficient are indicated.

and $C(\mathbf{D}, \mathbf{M})$ are entirely symmetric in FP and FN (notice that $H(\mathbf{D}) = 1$ (bit) is a constant). Thus the partial derivatives of these functions with respect to FP and FN are also symmetric. With the constraints above, the Lagrange equations for the optimum remain similar and therefore around the optimum $FP \approx FN$, which implies $x \approx y$ and therefore the specificity is close to the sensitivity.

A Mathematical Appendix

A.1 Relation Between Relative Entropy and Logarithmic LMS

The following two error functions, computed here for simplicity on a single input,

$$\begin{aligned} H &= d \log \frac{d}{m} + (1-d) \log \frac{1-d}{1-m} \\ LQ &= -\log[1-(d-m)^2] \end{aligned} \tag{35}$$

have been used in classification problems, for instance in combination with neural networks. $H = H(d, m)$ is just the relative entropy or KL distance and LQ is a logarithmic form of the LMS distance. It is clear that both H and LQ are symmetric in m and d. They are 0 if and only if $m = d$ and they infinite if $m = 1$ and $d = 0$ or $m = 0$ and $d = 1$. The latter property, i.e. a large dynamic range leading to large derivatives, is the main practical reason why these functions have been used in practice and have lead to better results than those obtained using the mean square error over the 0–1 range. In fact, H and LQ behave very similarly. To see this, consider first the case of large errors. This is the case, for instance, when $d = 1$ and m is close to 0. In this case we have

$$\begin{aligned} H &= -\log m \\ LQ &= -\log m - \log(2-m) \end{aligned} \tag{36}$$

Thus the two functions behave similarly when the error is large. In the small error regime, we write $m = d + \epsilon$ and Taylor expand both functions. Interestingly, the first order terms vanish and we are left with the second order terms

$$\begin{aligned} H &\approx \frac{1}{d(1-d)} \frac{\epsilon^2}{2} \\ LQ &\approx \epsilon^2 \end{aligned} \tag{37}$$

Again the behavior is very similar, quadratic in the error ϵ, except that H magnifies the error as d approaches 1 or 0, i.e. when the target value is more certain—a sensible thing to do. In conclusion both H and LQ achieve a large dynamic range and in similar fashion. In our opinion, however, H is preferable because of the probabilistic model that comes with it and because of the error modulation near the boundaries. The sometimes-used argument that the derivative of LQ is particularly simple is not really valid. Indeed, the derivative of LQ is simple $\partial LQ / \partial m = 2(m-d)/[1-(m-d)^2]$ and corresponds to the derivative obtained with the LMS error rescaled by the factor $1-(m-d)^2$, which is large when m and d differ a lot in the 0–1 range. But the derivative of the relative entropy is even simpler since $\partial H/\partial m = (m-d)/m(1-m)$. As we have seen, m should be computed as the output of a logistic function $m = f(u) = 1/(1+e^{-u})$. Since $\partial m/\partial u = f(u)(1-f(u)) = m(1-m)$ this finally yields the most simple expression $\partial H/\partial u = (m-d)$.

A.2 Learning by Maximizing Correlations

In general the correlation coefficient is a continuous and differentiable function of the prediction values m_i. If these in turn depend on a parameterized model, as in the case of neural networks, then we can compute the derivative of $C(\mathbf{D}, \mathbf{M})$ with respect to any parameter w of the model. But before we proceed with this calculation it must be pointed out that C varies between -1 and 1 and therefore it has poor dynamic range. To improve the range we can take the logarithm of C as our measure of performance. This requires that C is positive. Thus we must start with an initial learning system that has a positive correlation. This is easily realized using a few steps from a different learning algorithm, or by trying several different starting points, or even with a random starting point since a model with a prediction that is negatively correlated can easily be transformed into a model with a positive correlation. In what follows we assume this has been done. One caveat is that the ensuing learning trajectory will be constrained to remain in the space where correlations remain positive. Thus we choose as a measure of performance the logarithmic correlation

$$LC = \log C(\mathbf{D}, \mathbf{M}) \tag{38}$$

Any parameter w in the model is then updated by gradient ascent to maximize the correlation

$$w(t+1) = w(t) + \eta \frac{\partial LC}{\partial w} = w(t) + \eta \frac{1}{C(\mathbf{D}, \mathbf{M})} \frac{\partial C(\mathbf{D}, \mathbf{M})}{\partial w} \tag{39}$$

where η is some positive learning rate. Thus we are left with the calculation of

$$\frac{\partial C(\mathbf{D}, \mathbf{M})}{\partial w} = \sum_i \frac{\partial C(\mathbf{D}, \mathbf{M})}{\partial m_i} \frac{\partial m_i}{\partial w} \tag{40}$$

It is easy to check that

$$\frac{\partial C(\mathbf{D}, \mathbf{M})}{\partial m_i} = \frac{(d_i - \bar{d})}{||\mathbf{D} - \bar{d}\mathbf{1}||\,||\mathbf{M} - \bar{m}\mathbf{1}||} - \frac{C(\mathbf{D}, \mathbf{M})(m_i - \bar{m})}{||\mathbf{M} - \bar{m}\mathbf{1}||^2} \tag{41}$$

where we use the standard notation $||\mathbf{X}|| = \sqrt{\mathbf{X}^2} = \sqrt{\sum_i x_i^2}$ and $\mathbf{1}$ is the column vector containing all 1's. While learning is not exactly on line, notice that the terms $||\mathbf{D} - \bar{d}\mathbf{1}||$, need to be computed only once for all and the terms $||\mathbf{M} - \bar{m}\mathbf{1}||$, its square, and C need to be computed only once at most per each pass through the entire training set (training epoch). Finally, we need to compute the partial derivatives $\partial m_i/\partial w$ and, in the case of a neural network this is just the back-propagation algorithm—a straightforward application of the chain rule. For completeness, we include the derivation here.

To be more specific then, consider a feed-forward neural network with K layers $L^1, \ldots, L^K$. In reality, it is not necessary that the network be layered as long as there are no directed loops—but layered networks are the most commonly used (see also [3]). Let W^h represents the matrix of connections from L^{h-1} to L^h. each unit in the network as a total input I and an output O satisfying

$$\begin{aligned} I_i^h &= \sum_j w_{ij}^h O_j^{h-1} \\ O_i^h &= f_i^h(I_i^h) \end{aligned} \tag{42}$$

where f_i^h is the transfer function of unit i in layer h, typically a logistic function $f(u) = 1/(1+e^{-u})$ but other transfer functions are possible. In this notation we also assume that there exists a unit in each layer with a fixed output of 1 that is used to create affine biases in the following layer. In vector notation, Equation 42 can be rewritten as:

$$I^h = W^h O^{h-1} \tag{43}$$

$$O^h = F^h(I^h) \tag{44}$$

Thus m depends on a parameter w_{ij}^h only through the quantity I_i^h. Thus we can compute the partial derivatives

$$\frac{\partial m}{\partial w_{ij}^h} = \frac{\partial m}{\partial I_i^h} \frac{\partial I_i^h}{\partial w_{ij}^h} = \epsilon_i^h O_j^{h-1} \tag{45}$$

Thus the partial derivative can be written as the product of two terms: the pre-synaptic term O_j^{h-1} and the post-synaptic term $\epsilon_i^h = \partial m / \partial I_i^h$. The post-synaptic term is easily computed recursively by backpropagation from the output layer towards the input layer:

$$\epsilon_i^h = \frac{\partial m}{\partial I_i^h} = \frac{df_i^h}{dI_i^h} \sum_k \epsilon_k^{h+1} w_{ki}^{h+1} \tag{46}$$

In vector notation, by letting ϵ^h be the vector of back-propagated values at layer h

$$\epsilon^h = \frac{dF^h}{dI^h} [W^{h+1}]^t \epsilon^{h+1} \tag{47}$$

where W^t denotes the transpose of W—the signature of back-propagation. Thus learning is achieved by putting together Equations 39, 40, 41, 45, and 46

Acknowledgements

The work of PB and YC is in part supported by an NIH SBIR grant to Net-ID, Inc. The work of SB and HN is supported by a grant from the Danish National Research Foundation.

References

[1] C. A. Andersen. *Neural Network Assignment of Protein Secondary Structure with Increased Predictability.* Master's thesis, Technical University of Denmark, May 1998.

[2] P. Baldi and S. Brunak. *Bioinformatics: The Machine Learning Approach.* MIT Press, Cambridge, MA, 1998.

[3] P. Baldi. Gradient descent learning algorithms overview: A general dynamical systems perspective. *IEEE Trans. on Neural Networks*, 6:182–195, 1995.

[4] S. Brunak and J. Engelbrecht. Correlation between protein secondary structure and the mRNA nucleotide sequence. *Proteins*, 25:237–252, 1996.

[5] M. Burset and R. Guigó. Evaluation of gene structure prediction programs. *Genomics*, 34:353–367, 1996.

[6] P. Y. Chou and G. D. Fasman. Empirical predictions of protein conformations. *Ann. Rev. Biochem.*, 47:251–276, 1978.

[7] P. Y. Chou and G. D. Fasman. Prediction of the secondary structure of proteins from their amino acid sequence. *Adv. Enzymol. Relat. Areas Mol. Biol.*, 47:45–148, 1978.

[8] D. Frishman and P. Argos. Knowledge-based secondary structure assignment. *Proteins*, 23:566–579, 1995.

[9] J. E. Hansen, O. Lund, N. Tolstrup, K. Rapacki, and S. Brunak. NetOglyc — Prediction of mucin type O-glycosylation sites based on sequence context and surface accessibility, *Glycoconjugate J.*, 15:115–130, 1998.

[10] S. M. Hebsgaard, P. G. Korning, N. Tolstrup, J. Engelbrecht, P. Rouzé, and S. Brunak. Splice site prediction in *Arabidopsis thaliana* pre-mRNA by combining local and global sequence information. *Nucl. Acids Res.*, 24:3439–3452, 1996.

[11] U. Hobohm, M. Scharf, R. Schneider, and C. Sander. Selection of representative protein data sets. *Protein Sci.*, 1:409–417, 1992.

[12] W. Kabsch and C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22:2577–2637, 1983.

[13] S. Kullback. *Information theory and statistics.* Dover Publications, New York, 1959.

[14] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Stat.*, 22:79, 1986.

[15] R. Linsker. How to generate ordered maps by maximizing the mutual information between input and output signals. *Neural Comp.*, 1:402-411, 1989.

[16] B. W. Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochim. Biophys. Acta*, 405:442–451, 1975.

[17] H. Nielsen, J. Engelbrecht, G. von Heijne, and S. Brunak. Defining a similarity threshold for a functional protein sequence pattern: The signal peptide cleavage site. *Proteins*, 24:316–320, 1996.

[18] H. Nielsen, J. Engelbrecht, S. Brunak, and G. von Heijne. Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *Prot. Eng.*, 10:1–6, 1997.

[19] F. M. Richards and C. E. Kundrot. Identification of structural motifs from protein coordinate data: Secondary structure and first-level supersecondary structure. *Proteins*, 3:71–84, 1988.

[20] B. Rost and C. Sander. Prediction of protein secondary structure at better than 70% accuracy. *J. Mol. Biol.*, 232:584–599, 1993.

[21] B. Rost, C. Sander, and R. Schneider. Redefining the goals of protein secondary structure prediction. *J. Mol. Biol.*, 235:13–26, 1994.

[22] C. Sander and R. Schneider. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins*, 9:56–68, 1991.

[23] Z. X. Wang. Assessing the accuracy of protein secondary structure. *Nat. Struct. Biol.*, 1:145–146, 1994.

[24] A. Zemla, C. Venclovas, K. Fidelis, and B. Rost. A modified definition of SOV, a segment-based measure for protein secondary structure prediction assessment. *Proteins*, 34:220–223, 1999.