Weighted quality estimates in machine learning

Levon Budagyan\textsuperscript{1,*} and Ruben Abagyan\textsuperscript{2}

\textsuperscript{1}Molsoft LLC, 3366 North Torrey Pines Court, Suite 300, San Diego, CA 92037, USA and \textsuperscript{2}The Scripps Research Institute, 10550 North Torrey Pines Road, Mail TPC-28, San Diego, CA 92037, USA

Received on March 7, 2006; revised on August 9, 2006; accepted on August 22, 2006

ABSTRACT

Motivation: Machine learning methods such as neural networks, support vector machines, and other classification and regression methods rely on iterative optimization of the model quality in the space of the parameters of the method. Model quality measures (accuracies, correlations, etc.) are frequently overly optimistic because the training sets are dominated by particular families and subfamilies. To overcome the bias, the dataset is usually reduced by filtering out closely related objects. However, such filtering uses fixed similarity thresholds and ignores a part of the training information.

Results: We suggested a novel approach to calculate prediction model quality based on assigning to each data point inverse density weights derived from the postulated distance metric. We demonstrated that our new weighted measures estimate the model generalization better and are consistent with the machine learning theory. The Vapnik-Chervonenkis theorem was reformulated and applied to derive the space-uniform error estimates. Two examples were used to illustrate the advantages of the inverse density weighting. First, we demonstrated on a set with a built-in bias that the unweighted cross-validation procedure leads to an overly optimistic quality estimate, while the density-weighted quality estimates are more realistic. Second, an analytical equation for weighted quality estimates was used to derive an SVM model for signal peptide prediction using a full set of known signal peptides, instead of the usual filtered subset.

Contact: levon@molsoft.com

1 INTRODUCTION

Machine learning methods, such as artificial neural networks (ANN) (Baldi and Brunak, 2001), support vector machines (SVM) (Scholkopf and Smola, 2001) and partial least squares regression (PLSR) (Geladi and Kowalski, 1986) are widely used to predict the theoretical prediction. Molecular biology is a particularly important application in which tens of thousands of genes and corresponding proteins in each organism need to be annotated and functionally characterized. Amino acid sequences or DNA sequences are often the input data for various predictions of biological properties. Weighting contributions from each sequence was used in a number of applications, for example, in derivation of the hidden Markov models (HMMs) and sequence profiles (Krogh and Mitchison, 1995; Vingron and Sibbald, 1993; Altschul et al., 1989; Thompson et al., 1994; Henikoff and Henikoff 1994; Abagyan and Batalov, 1997; Pollastri et al., 2002). Detailed reviews of the weighting schemes can be found in Baldi and Brunak (2001) and in Heringa (2002). It seems logical to extend the weighting approach to calculating prediction quality and other machine learning problems.

In the absence of object weights, a practical way of dealing with the data bias is to select a more equally spaced subset of objects based on a similarity cutoff. This method is frequently applied to reduce a set of protein sequences. For example, Nielsen et al. (1997) used a data filtering method of Hobohm et al. (1992) to construct a training set for the signal peptide discrimination problem. We used their signal peptide training set to illustrate our density-weighting alternative approach by proposing a target-object-space and deriving inverse density weights.

Some other examples of the similarity filtering in the sequence space include subcellular location prediction using SVM (Park and Kanehsa, 2003), prediction of major histocompatibility complex with SVM (Dönnès and Elofsson, 2002), and membrane protein predictions where sequence clustering was used for filtering (Möller et al., 2000).

In this work we justified weighting quality measures as a mean to eliminate the training set bias in the target-object-space. We estimated the density of training set objects, and used the inverse density as the object weight. In such scheme, weights are...
determined by the choice of a target-object-space, which depends heavily on the nature of the objects in the training set and the predicted property.

2 METHODS

2.1 Machine learning framework: generalization error and its estimates

In most cases the popular machine learning techniques fall under a model of deterministic learning. There are several mathematical formalizations of this problem (e.g. Scholkopf and Smola, 2001; Anthony and Holden, 1998; Kearns and Ron, 1997; Wolpert, 1996). This section describes the error-hypothesis notation we used.

We are given set $X$ of objects and set $Y$ of labels. More specifically, $Y = \{-1, 1\}$ for binary classification, $Y = \{1, \ldots, k\}$, $k > 2$ for $k$-class classification, $Y = \mathbb{R}$ for regression estimation. For each object $x \in X$ there is an original label $y = c(x) \in Y$, i.e. we have a target concept $c \in \{X \rightarrow Y\}$.

The training set consists of $n$ object-label pairs, $(x_1, y_1), \ldots, (x_n, y_n), y_i = c(x_i)$, where objects are selected independently according to the probability $P$ from $X$.

The goal of a learning procedure is to approximate the target ($c$) using only the training data $(D)$. The learning algorithm $L$ can be defined as a function that produces a hypothesis $h \in \mathcal{H}$ for each given training set. Here $\mathcal{H}$ is a subset of the $X$ to $Y$ functions: $\mathcal{H} \subset \{X \rightarrow Y\}$, $\mathcal{H}$ is called the hypothesis space\(^1\) of the algorithm $L$, and every element of $\mathcal{H}$ is called a hypothesis. As we are interested in producing a hypothesis closely approximating the target concept $c$, we introduce a generalization (or true) error $e_T(h, c)$ as the discrepancy between the labels produced by $h$ and $c$ over $X$.

This measure, in fact, highly depends on the type of labels we use, for example, in the binary classification case a natural choice would be the empirical error to produce hypotheses that are very good on the training set. The absolute knowledge of how good a hypothesis may be, is an appealing, yet not reachable, task for a real-life machine learning problem. The estimates of the generalization error often become a part of the learning procedure, as we may want to use them for choosing between different algorithms $L$. It makes the generalization error estimation a substantial part of the property prediction process. There are various results justifying $k$-fold cross-validation, and other estimates: Kearns and Ron (1997), Anthony and Holden (1998), Blum et al. (1999) and Gavin and Teytaud (2002).

2.2 Neutralizing bias of a training set: replacing the generalization error by a uniform estimate

Often the training set is biased and unbalanced with some parts of the object space overpopulated and others underpopulated. Many learning algorithms implicitly or explicitly assign labels based on the similarity, for example, nearest neighbor algorithms (Devroye et al., 1996) and SVMs (Scholkopf and Smola, 2001). Generalization error estimates for the prediction models will be optimistic in those cases.

Therefore, there is a need to replace the traditional object-uniform quality estimates by a space-uniform quality estimate. What can be a natural way to describe the unwanted bias due to uneven object distribution in the training set?

The training set may be considered biased only in terms of some uniformness concept in $X$, i.e. when there is a known structure in $X$, not defined by the training set. Usually that structure is given as some distance or similarity function. Distances and similarity functions are related; given a similarity, some distance may be defined to be an anti-similarity function. So let us generalize the learning model defined above in the following way: let us assume that $X$ is a metric space (target-object-space) with the distance function $p$. We emphasize that the distance function is postulated independently from the rest of the machine learning framework, and is meant to reflect our vision of the actual object distribution in $X$.

This distance function defines another probability measure $\mu$ on $X$, not necessarily related to the probability measure $P$.

The postulated probability function $\mu$ modifies the formula for generalization error:

$$e_T(h, c) = \mu(\{x \in X : h(x) \neq c(x)\})$$

This expression will be referred to as the uniform error.

There are more general models for the learning process. For instance, the Bayesian approach uses a joint distribution over $X \times Y$ for drawing training objects; thus, the hypotheses are obtained from the conventional probability $P(y|x)$. These approaches still define the generalization error via the probability used for drawing the training set, and will not be considered in this work.

2.3 Overview of the theoretical results for classifier generalization error estimation

Both empirical error and cross-validation error do not use the distribution $P$ explicitly in their definitions. Nevertheless, the classes of learning algorithms are designed so that standard error estimates, such as empirical, cross-validation and hold-out errors, stays close to the generalization error. Those estimates are also explicitly minimized during the course of the model derivation.

Thus, if we want to estimate the uniform error instead of the generalization error, we have to consider that both the empirical error and the cross-validation error are related to $P$. Thus, we have to modify the definitions of error estimates, in order to make them work for $e_T(h, c)$.

The quality of approximation of the $e_T(h, c)$ by the empirical, hold-out and cross-validation estimates was considered in a number of publications (Kearns and Ron, 1997; Anthony and Holden, 1998; Blum et al., 1999; Devroye et al., 1996; Scholkopf and Smola, 2001).

The quality measures are mostly formulated for the case of a hypothesis space with finite Vapnik–Chervonenkis dimension, $V_H$, and use the Vapnik–Chervonenkis theorem for confidence bounds.

\(^1\)Hypothesis space may not be a space in a mathematical sense.
Vapnik–Chervonenkis dimension \( (V_n) \) of the hypothesis space \( \mathcal{H} \) is the maximal number for which the following is true: for any selection of \( V_n \) points from \( X \), and for any possible labelling of them\(^2\), there exists a hypothesis \( h \in \mathcal{H} \) which classifies these points without mistakes.

**Vapnik–Chervonenkis theorem:** For a hypothesis space \( \mathcal{H} \) with Vapnik–Chervonenkis dimension \( V_n < n \), for any given \( \delta > 0 \),

\[
P\left( \sup_{h \in \mathcal{H}} \left| \epsilon_r(h,c,D) - \epsilon_r(h,c) \right| < \epsilon_r(V_n,n,\delta) \right) > 1 - \delta.
\]

(2)

\[
\epsilon_r(V_n,n,\delta) = \frac{8}{n} \left( V_n \left( \ln \frac{n}{V_n} + 1 \right) + \ln \frac{4}{\delta} \right)
\]

(3)

This theorem gives a confidence interval for the distance between the empirical error and the generalization error; moreover, the confidence interval is uniform over the hypothesis space \( \mathcal{H} \).

Supplementing the Vapnik–Chervonenkis dimension constrain with some kind of algorithm stability condition, confidence bounds for the cross-validation and hold-out estimates can be derived. Algorithm stability conditions usually require the hypothesis, chosen by the learning algorithm \( L \), or at least the hypothesis error, not to change drastically with the removal of one element from the training set (Kearns and Ron, 1997).

### 2.4 Using density to estimate the generalization error

We have the following estimates of \( \epsilon_r(h,c) \): the empirical error \( \epsilon_r(h,c,D) \), the cross-validation error and the hold-out error. We want to replace them by similar estimates for \( \epsilon_r(h,c) \). By definition,

\[
\epsilon_r(h,c) = P(\{ x \in X : h(x) \neq c(x) \}) = \int I_{h(x) \neq c(x)} dP
\]

and

\[
\epsilon_r(h,c) = \mu(\{ x \in X : h(x) \neq c(x) \}) = \int I_{h(x) \neq c(x)} d\mu,
\]

where

\[
I_a(x) = \begin{cases} 
1, & x \in A \\
0, & \text{else} 
\end{cases}
\]

is the indicator function. Let us assume that measures \( P \) and \( \mu \) are such that Radon–Nikodym’s theorem (Devroye et al., 1996) holds true, i.e. the measure derivative \( f = \frac{d\mu}{dP} \) exists. Then we can rewrite the uniform error formulation as follows:

\[
\epsilon_r(h,c) = \int I_{h(x) \neq c(x)} f \, dP.
\]

**Example.** In a special case, when \( X \) is a bounded Euclidean space \([0,1]^d\), \( \mu \) is the volume measure\(^3\) in \( X \), and \( P \) is a probability with density \( p(x) \), the measure derivative \( f = \frac{d\mu}{dP} = p(x) \). \( \epsilon_r(h,c) \) may be looked upon as an \( f \)-weighted version of \( \epsilon_r(h,c) \). We will use \( f \)-weighting to modify \( \epsilon_r(h,c) \)-estimates into \( \epsilon_r(h,c) \)-estimates.

The Vapnik–Chervonenkis theorem has a more general formulation with errors replaced by risks with customized loss function (Scholkopf and Smola, 2001). Risk is defined as \( R(h,c,D) = \int l(x,c,h(x)) dP \), and empirical risk is defined as \( \hat{R}(h,c,D) = \frac{1}{n} \sum_{i=1}^{n} l(x_i,c,h(x_i)) \), where the loss function \( l : X \times Y \times Y \rightarrow [0,\infty) \) is a function with property \( l(x,y,y) = 0 \). Classification errors are risks with \( l(x,c,h(x)) = I_{h(x) \neq c(x)} \) loss function.

The formulation of the Vapnik–Chervonenkis theorem in risk-loss terms states that for a hypothesis space \( \mathcal{H} \) with \( V_n < n \), for any given \( \delta > 0 \),

\[
P\left( \sup_{h \in \mathcal{H}} \left| \hat{R}(h,c,D) - R(h,c) \right| < \epsilon_r(V_n,n,\delta) \right) > 1 - \delta.
\]

(4)

Since \( \epsilon_r(h,c) \) can be viewed as a risk with loss function \( l(x,c,h(x)) = f(x)I_{h(x) \neq c(x)} \), and \( f \)-weighted empirical risk is the empirical risk with that loss function, we get an upper bound for their difference. So, we suggest the following algorithm for obtaining uniform error \( \epsilon_r(h,c) \)-estimates:

1. Estimate the \( \mu \)-to-\( P \) measure derivative \( f \). As we have illustrated in the example above, in many cases this may be done using density estimation methods (Devroye and Lugosi, 2001; Scott, 1992).
2. Use the estimate of \( f \) to weight objects in empirical, hold-out and cross-validation estimates.

The drawbacks are as follows:

1. Density estimation cannot be performed with 100% accuracy, so all confidence bounds will get an extra term.
2. The re-formulated Vapnik–Chervonenkis theorem still depends on \( P \). And when some event has \( P \) probability close to 1, not necessarily the \( \mu \) probability is as close to 1.

Yet when \( P \) is distorted a lot in relation to \( \mu \), trying to establish the uniform error by the means of weighted error estimates is a better choice. In addition, density analysis may be helpful to establish the difference between \( P \) and \( \mu \).

### 2.5 Density-weighted quality measures

The error \( \epsilon_r(h,c) \) is not the only measure used for assessing the quality of binary classification. Other measures are accuracy, specificity, sensitivity, Mathews correlation, etc. Accuracy is opposite to the error: \( 1 - \epsilon_r(h,c) \).

Sensitivity and specificity are the accuracies calculated separately for positively and negatively labeled objects in the dataset. Mathews correlation\(^4\) is defined as follows:

\[
\text{Mathews correlation} = \frac{S_{PP}S_{NN} - S_{PN}S_{NP}}{\sqrt{(S_{PP}+S_{NN})(S_{PP}+S_{NP})(S_{PN}+S_{NP})(S_{PN}+S_{NN})}}.
\]

(5)

where \( S_{PP} \) is the number of training set objects which have actually a positive (P) label, but were predicted to have negative (N) label by the hypothesis.

The three other values \( S_{NP}, S_{PN}, S_{NN} \) are defined accordingly. The range of Mathews correlation is \([-1,1]\); larger correlation means better hypothesis. The weighted Mathews correlation has the same definition, except the four sums in its representation are weighted, e.g. \( S_{PN} \) is the sum of the weights of objects with positive observed and negative predicted label. All other binary classification quality measures can also be defined using these four numbers: \( S_{PP}, S_{PN}, S_{NP}, S_{NN} \). For instance, we can define accuracy as \( \frac{S_{PP} + S_{NP}}{S_{PP} + S_{NP} + S_{NP} + S_{NN}} \), sensitivity as \( \frac{S_{PP} + S_{NP}}{S_{PP} + S_{PN}} \), and specificity as \( \frac{S_{NP} + S_{NN}}{S_{PP} + S_{NN}} \). Weighted versions of classification quality measures may be obtained by using weighted \( S_{PP}, S_{PN}, S_{NP}, S_{NN} \).

### 2.6 Kernel density estimation

Kernel density estimation (KDE) (Devroye and Lugosi, 2001) is a popular approach to the density estimation problem. Given the training set objects \( \{x_1,...,x_n\} \) from the target-object-space \( S \), the density for any object \( x \in S \) is defined as follows:

\[
p(x) := \frac{1}{n} \sum_{i=1}^{n} K(x_i, x),
\]

(6)

where the kernel \( K(x,y) \) is a symmetric function with the following property:

\[
\forall y \int_{S} K(x,y)dx = 1.
\]

(7)

\(^3\)That is any separation in two classes—there are \( 2^n \) separations possible for \( V_n \) fixed points.

\(^4\)For this specific measure, \( \mu \) will change in \([0,1]\) range, and thus will be a probability measure, too.

\(^2\)some properties of Mathews correlation can be found in Gower and Legendre (1986), where it is denoted as the \( S_{14} \) similarity coefficient for binary variables.
Table 1. Uniform error is better approximated by weighted estimates

<table>
<thead>
<tr>
<th></th>
<th>Actual values</th>
<th>Unweighted estimates</th>
<th>Weighted cross-validation estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Generalization</td>
<td>Uniform</td>
<td>Empirical</td>
</tr>
<tr>
<td>Error</td>
<td>0.04 ± 0.01</td>
<td>0.33 ± 0.04</td>
<td>0.00 ± 0.00</td>
</tr>
<tr>
<td>Mathews correlation</td>
<td>0.92 ± 0.01</td>
<td>0.35 ± 0.08</td>
<td>1.00 ± 0.00</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>0.96 ± 0.01</td>
<td>0.68 ± 0.10</td>
<td>1.00 ± 0.00</td>
</tr>
<tr>
<td>Specificity</td>
<td>0.96 ± 0.01</td>
<td>0.67 ± 0.10</td>
<td>1.00 ± 0.00</td>
</tr>
</tbody>
</table>

A thousand case summary. Only 5-fold cross-validation was used. All weighted cross-validation estimates were generated using the inverse-density-weighting technique. We used two density calculation schemes: density given by its analytical formula and density estimated from the training set data. We also calculated the sensitivity, the specificity and the Mathews correlation of the obtained classification models. The weighted specificity, sensitivity and Mathews correlation were calculated as described in Section (2.5).

3 RESULTS AND DISCUSSION

3.1 Building weighted quality estimates for a set with simulated bias

3.1.1 Generating a biased dataset and applying a classification algorithm. We built a biased synthetic training set in which points on a line were classified into two classes. This example problem was built to show how strong the impact of bias can be and to compare the estimated and the actual density weights.

The bias was designed as follows. The [0,1] segment was divided into s (an even number) equal subsegments, using a grid with a step (1/s). Then points in odd and even-numbered subsegments received, respectively, y = 1 and y = -1 labels. Varying parameter s allowed to produce prediction tasks of different complexity. The training set points were extracted using the following highly distorted distribution P on [0, 1]: it returned a point from the large subsegment [0, 0.9) with only 10% chances and a point from [0.9, 1] subsegment with 90% chances, i.e. 90% of the points would be in the 10% of the segment and vice versa. Inside these two segments, P had uniform distribution of points.

As a learning algorithm L, a very simple 1-Nearest Neighbor (1-NN) learner (Devroye et al., 1996; Aha et al., 1991) was used. 1-NN learner memorizes the training set, and then assigns to any object the label of the closest object in the training set. As a dissimilarity measure, the Euclidean distance was used. The area measure defined on [0,1] served us as the uniform measure µ, i.e. we were looking for the answer to the question: what part of the [0, 1] segment would be correctly predicted by our classifier?

3.1.2 Approximating the density, generalization and uniform errors. A series of experiments was made with the labelling complexity s = 40. The training set consisted of n = 400 points generated with the P-distribution. We calculated the empirical (self-prediction) error, the 5-fold cross-validation error and several inverse-density-weighted versions of the 5-fold cross-validation error. Two types of weights were used: weights obtained from the exact analytical formula for the density of the distribution P and weights from the density estimate. The density was estimated using the KDE algorithm described in section 2.6 with a simple one-dimensional kernel (the Epanechnikov kernel)

\[ K_h(x, x) = \frac{3}{4h} \max \left( 1 - \frac{|x_1 - x_2|^2}{h^2}, 0 \right), \]

where h is the kernel bandwidth—a density estimation parameter. We used h = 0.1.

We also calculated the ‘actual’ values of the hypothesis’ generalization error erP and uniform error erµ. We generated 40 000 points using the P-distribution. The error of the hypothesis on these points was considered the generalization error. Analogously, we used 40 000 points generated with uniform distribution to define the uniform error.

3.1.3 Comparing the biased and unbiased quality estimates Table 1 summarizes the results for 1000 runs. Weighted error estimates gave a good approximation of the uniform error, close in average and with a small deviation. At the same time, the unweighted estimates were very close to the generalization error. We also see that there may be a considerable difference between the generalization error and the uniform error. The same applies to the other quality measure we used: the sensitivity, the specificity and the Mathews correlation (see Section 2.5).

Owing to the nature of the data, we had more information about the P distribution than about µ, thus any P-related statistics were more stable, and the generalization error erP was approximated better than the uniform error erµ.

3.2 Classifying a biased set of signal peptide sequences versus non-signals

3.2.1 The sequence sets used for classification. We used the training sets of Nielsen et al. (1997) for the binary classification task of distinguishing signal peptides from the cytoplasmic proteins. Specifically, we used the Gram-positive bacteria dataset that was present in two forms: as the total set collected and filtered from SWISS-PROT (Boeckmann et al., 2003), and as its non-redundant subset. For each pair of sequences with the alignment score-based distance below a certain threshold, one was excluded.

Our goal was to compare the redundancy reduction approach with the inverse density weights applied to the full sequence set. First, we researched the similarity between the two approaches. Second, we used the KDE procedure to find out if the reduction of the training set affects the performance of the model. The ability of a model to classify the rare sequences in the 5-fold cross-validation procedure served as a model performance measure.

3.2.2 The sequence descriptors and the classification algorithm. The authors of the original publication (Nielsen et al., 1997) used scores based on the local predictions for different positions in...
the sequences. The amino acid sequence information was sparsely encoded for fixed-size windows centered at different positions in the sequence to produce descriptors for a neural networks prediction algorithm.

We used gapped pair counts as descriptors and an SVM classifier as the prediction method. The gapped pair count vector of a sequence with gap length \( l \geq 0 \) is a vector with coordinates indexed by sequence alphabet symbol pairs, i.e. it has \( 20 \times 20 \) coordinates, one for each pair of amino acids from the alphabet ACDEFGHIKLMNPQRSTVWY. For each pair, the corresponding vector component contains the quantity of such ordered pairs with a given gap between them. For example, for gap length \( l = 2 \) and alphabet pair (A,A), the corresponding vector component will contain the number of A**A subsequences in the sequence, where \( * \) stands for any symbol. Our descriptor vectors were composed from the pair count vectors with different gaps. We concatenated the pair count vectors for the gap sizes from 0 up to some gap length \( l_0 \), which was one of the experiment parameters. In total, we had \( m = 20^2(l_0 + 1) \) components in each sequence descriptor vector.

For our predictions, we used the \( l_0 = 20 \) value.

The gapped pair counts contain information about both the sequence composition and the offset-tolerant order of residues.

To separate signal peptide sequences from the cytoplasmic proteins, we used the SVM classifier algorithm (see Scholkopf and Smola, 2001) with dot kernel\(^3\). Sequences were transformed into concatenated pair count vectors. Dot kernel uses vector inner product as the measure of similarity between the training set objects. The gapped pair count vectors are sparse, i.e. they contain many zero elements; hence, calculation of their inner product can be carried out effectively in a time proportional to the number of non-zero elements, which is bounded by \((l_0 + 1)n\). We did not use more complex kernels, such as polynomials and Gaussian, because the input vector space was abundant enough to allow SVM to produce a good approximation of the solution.

3.2.3 Deriving density estimates from the pairwise sequence scores

We wanted to weight the sequences in the training set according to their uniqueness.

This was performed using the KDE algorithm with a kernel function based on sequence pairwise alignment scores.

In order to apply the KDE algorithm and derive the sequence weights, we had to go through the following steps:

1. Calculate pairwise alignment scores for \( n \) sequences in the training set: \( A_{ij} = A(x_i, x_j) \).
2. Define the sequence space where the density should be estimated (the target-object-space). Find the distribution of alignment scores \( A(x, y) \) in the target-object-space.
3. Define the kernel function as a function of the alignment score: \( K(x, x_i) = K(A_{ij}) \). The kernel function was constructed to score the sequence similarity, i.e. have higher values for more similar sequences. At the same time the kernel was required to satisfy the main kernel property (7) in the target-object-space.
4. Substitute the kernel \( K(A) \) into the KDE formula (6) and calculate weights.

To calculate the pairwise sequence scores we used the Needleman and Wunsch (1970) sequence alignment algorithm with zero end-gap penalties (ZEGA) implemented in the ICM program (Molsoft, 2006). The ZEGA score is calculated from the residue comparison matrix and the gap open and gap extension parameters. The Gonnet et al. (1992) residue comparison matrix and gap open/gap extension penalties 2.4/0.15 were used.

The average length of sequences in our dataset was 65 amino acids with SD 7. So we defined a set of all sequences with length 65 as the target-object-space. We decided that the distribution of alignment scores \( A(x, y) \) between a variable sequence \( x \) and some fixed sequence \( y \) could be replaced by the distribution of alignment scores for structurally unrelated sequences. Abagyan and Batalov (1997) derived an analytical estimate for the empirical distribution of the ZEGA score of structurally unrelated sequences:

\[
P(A < t) = e^{-\frac{t}{\sigma(L)}} \quad z(t) = \frac{t - m(L)}{\sigma(L)},
\]

where the distribution parameters \( m \) and \( \sigma \) depend on the choice of the residue comparison matrix, the alignment gap penalties and the length \( L \) of the shorter sequence in the alignment. The distribution parameters such as \( m(L) = 2.819 + 0.46L \) and \( \sigma(L) = 1.57 + 0.0056L \) at \( L = 65 \), were applied to calculate the \( z(t) \) value.\(^6\)

Our kernel had to satisfy the kernel property Equation (7) with a proper integration base \( S \) (the target-object-space). The extreme value distribution density function is

\[
D(t) = \frac{1.618}{\sigma(L)} e^{\frac{1.618(t) - e^{1.618(t)}}{\sigma(L)}}.
\]

The left-hand side of Equation (7) can be rewritten as follows:

\[
\int_S K(A(x,y))dx = \int_0^\infty K(A)D(A)dA = \int_0^\infty k(A)dA,
\]

where \( k(A) = K(A)D^{-1}(A) \). We chose a simple triangular function as a prototype for \( k \): \( k(A) = 0 \) for alignment scores below some threshold \( \delta \), and \( k \) linearly grows from 0 to 1 as \( A \) grows from \( \delta \) to its maximal value. The inverse distribution density is:

\[
D^{-1}(t) = \frac{\sigma(L)}{1.618} e^{\frac{1.618(t)}{\sigma(L)} e^{-1.618(t)}}.
\]

So we defined the kernel function as \( K(x, y) = D^{-1}(A(x, y)) \) \( k(A(x, y)) \) with two modifications. First, we found using ‘\( e \)’ as the exponent base impractical, so we replaced it by 1.1. That was enough to rank rare alignment scores considerably higher. Second, we omitted the scaling coefficient and ignored the less important \( e^{1.618} \) part of \( D^{-1} \).

Summarizing, our kernel function was as follows:

\[
K(x, y) := \begin{cases} 1.1^{1.618(z(A))}k(A), & A > \delta, \\ 0, & A \leq \delta, \end{cases}
\]

where \( A = A(x, y) \), \( k(A) \) mapped alignment scores above the threshold to the [0,1] range: pairwise scores were divided by the geometric average of self-scores:

\[
k(A(x, y)) := \frac{A(x, y) - \delta}{\sqrt[A(x,y)]{A(x,y)}}
\]

\(^3\)Kernels used for SVM predictions differ from those used in KDE.

\(^6\)The estimated distribution parameters for the Gonnet residue comparison matrix and the 2.4/0.15 gap penalties were taken from Abagyan and Batalov (1997).
The threshold value δ was defined as the 20% percentile of the score distribution: \( P\{A < \delta\} = 0.2 \).

We used a modified version of the general KDE formula (6) for calculating the densities of the elements in the training set:

\[
p(x_i) = \frac{1}{n} \left( \sum_{j \neq i} K(x_i, x_j) + 0.000001 \right). \tag{10}\]

This formula differs from the original KDE formula as follows. It contains some small value 0.000001 instead of the \( K(x_i, x_j) \) ‘diagonal’ term. The correction was carried out because the kernel function was constructed in assumption that the occurrence of high alignment scores is practically improbable, so the kernel depends on the alignment score exponentially. At the same time, the \( K(x_i, x_j) \) term is present in each standard KDE formula and it would dominate over all other KDE terms, making density weights less influential. The corrected KDE guarantees that any singleton, i.e. a sequence without a close neighbor in the training set, will get a lower weight compared to non-singletons.

**Connection with the redundancy reduction approach.** Density estimation based weights were designed to extend the cut-off rules used for redundancy reduction; therefore, we expected that the sequences which survived after the elimination of redundant sequences would mostly be estimated as singletons, i.e. have small density weights. The histogram in Figure 1 shows that this is indeed the case.

3.2.4 Discussion. Our goal was to introduce an algorithm to estimate unbiased quality measures of a predictive model built from a biased dataset. We wanted the quality measures to be unbiased in a sense that they would describe the generalization quality of the model in the space of all sequences of certain length (our target-object-space). Unweighted quality estimates depend on the distribution of the training set in that space; thus, they are biased when the training set distribution is distorted.

Let us note that our aim was not to produce the best predictive model, but rather to assess the uniform error of a classifier. Nevertheless, the learning algorithm we use produced a high-quality predictive model.

Table 2 contains statistics of the prediction parameters for 1000 runs on the Gram-positive bacteria dataset. Results are given for two models: the model built from the whole set and the model built from the reduced set. Using weights allowed us to get more adequate results without reducing the training set. We found out that cross-validation estimates for both unweighted models were too optimistic in average, though the statistics for reduced model (model, built on the reduced set) was less subject to overestimation. It can be explained by the observation that filtering out similar data points in the training set partially straightens the distribution of the training set in the target-object-space.

The prediction model based on the full set gave better performance even in terms of unbiased quality. Density analysis showed us that it was due to the model quality, i.e. the model built on the large set correctly classified some unique sequences in the training set, which the reduced model misclassified. We took an arbitrary run of our program on the Gram-positive bacteria dataset, where the weighted Mathews correlation estimate differed drastically for two models: the total dataset weighted Mathews correlation was 0.85(unweighted 0.89) and the reduced dataset Mathews correlation was 0.91(unweighted 0.96). Table 3 lists the names and densities of

![Kept sequences](image)

**Fig. 1.** Correlation between the threshold-based training data filtration and the density weights. All the sequences in the Gram-positive signal peptide dataset were sorted by their density (sequences from dense regions first, singletons last). This histogram shows the number of sequences in each successive group of 10 that survived training data filtration.
objects in the dataset that were mispredicted in the 5-fold random cross-validation procedure. We can see that the non-reduced model performed better on some sequences with density values close to 0 (singleton).

4 CONCLUSIONS

- A dataset used for machine learning does not need to be artificially reduced via a similarity cutoff. The entire set can be used if the contribution of objects is weighted.
- The Vapnik–Chervonenkis theorem establishing the bounds of the error estimation can be extended to include the weights. The quality estimates, including cross-validation and hold-out estimates, can be modified to include the weights.
- The density-weighted quality measures are more realistic and lead to better models upon optimization of the error in the parameter space of a machine learning method.
- A recipe for estimating sequence density weights from the sequence pairwise alignment scores is presented.

ACKNOWLEDGEMENTS

We are grateful to our colleagues at Molsoft and TSRI for valuable comments and discussions: Irina Kufareva, Eugene Rausch, Maxim Totrov, Andrew Orry and George Nicola. We are also thankful to Nielsen et al. (1997) for making their datasets publicly available.

Conflict of Interest: none declared.

REFERENCES