Prediction algorithms and confidence measures based on algorithmic randomness theory

Alex Gammerman*, Volodya Vovk

Computer Learning Research Centre and Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK

Abstract

This paper reviews some theoretical and experimental developments in building computable approximations of Kolmogorov’s algorithmic notion of randomness. Based on these approximations a new set of machine learning algorithms have been developed that can be used not just to make predictions but also to estimate the confidence under the usual iid assumption. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Confidence; Machine learning; Randomness; Transduction

1. Introduction

In many problems of pattern recognition and regression analysis a program must be capable of learning from known examples (represented by attribute vectors with corresponding labels) and extrapolating to predict a new unknown (unlabelled) example. Traditional, low-dimensional, small scale data have been successfully dealt with by conventional software engineering and classical statistical methods, such as discriminant analysis, neural networks, genetic algorithms and others. But the change of scale in data collection and the dimensionality of modern data sets has profound implications on the type of analysis that can be done. Recently, several kernel-based learning algorithms have been developed [9]; for example, the support vector machine (SVM) [10] is quickly gaining popularity, and it is widely believed that it will help to meet the challenge of analyzing very large data sets.

Learning machines such as the SVM often perform well in a wide range of applications and have nice theoretical properties without requiring any parametric statistical
assumptions about the source of data (unlike traditional statistical techniques); the only assumption made is the iid assumption (the examples are generated from the same probability distribution independently of each other). However, a typical drawback of techniques such as the SVM is that they usually do not provide any useful measure of confidence in the predicted labels of new, unclassified examples.

Confidence estimation is a well-studied area of both parametric and non-parametric statistics; however, usually only low-dimensional problems are considered. In this paper, we review the approach currently being developed at the Computer Learning Research Centre (Royal Holloway, University of London) which allows us to compute prediction with confidence for high-dimensional iid (independent and identically distributed) data; for details, see [1,8,11]. This approach is based on practical approximations to the universal measures of confidence given by the algorithmic theory of randomness [3]. We first describe the SVM method as a technique of supervised machine learning. Then we briefly outline the basic ideas of algorithmic randomness and its approximation. In the final part of the paper, we present experimental results using a well-known data set.

2. Background

Let us assume that we are given a training set of examples \((x_1, y_1), \ldots, (x_l, y_l)\), where \(x_i\) is a vector of attributes and \(y_i\) is a label, and our goal is to predict the classifications \(y_{l+1}, \ldots, y_{l+k}\) for a test set \(x_{l+1}, \ldots, x_{l+k}\). We make only one assumption about the data generating mechanism: all the examples have been generated by some fixed but unknown stochastic mechanism (the iid assumption).

Our problem is to construct a learning machine that must for any \(x\) provide a label \(y\) which is as close to the true label as possible (traditionally, the goal is to minimize some measure of discrepancy between the prediction \(\hat{y}\) and the true label \(y\) of the new example \(x\)). In the case of pattern recognition, the label takes only two values, \(y \in \{-1, 1\}\); in the regression case, the label is a real value, \(y \in \mathbb{R}\). To illustrate our main points, we briefly consider here only the pattern recognition case, and refer the reader to [10] for details and the regression case.

2.1. Support vector machines

The main idea of a SVM is to map the original set of vectors into a high-dimensional feature space, and then to construct a linear separating hyperplane (or a linear regression function, in the regression case) in this feature space. According to the SVM approach we should look for a separating hyperplane with a small number of errors (or, more generally, a small sum of penalties reflecting the grossness of errors) and a large “margin” (which is, in the simplest case, just the distance from the separating hyperplane to the nearest vector). Formally, this is done by finding the minimum of the objective function,

\[
\frac{1}{2} (w \cdot w) + C \left( \sum_{i=1}^{l} \xi_i \right) \rightarrow \min,
\]
subject to the constraints
\[ y_i((x_i \cdot w) + b) \geq 1 - \xi_i, \quad i = 1, \ldots, l. \]
Here \( C \) is a fixed positive constant (maybe \( \infty \)), \( w \) are weights, \( b \) is the intercept, and \( \xi_i \) are non-negative “slack variables”.

The mapping of the original set of vectors often leads to a problem in dealing with a very large number of parameters, and therefore to serious computational difficulties. Vapnik suggested reformulating the problem using Lagrange multipliers; the original setting of the problem can be replaced by the following “dual” setting: maximize a quadratic form
\[
\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j x_i x_j K(x_i, x_j) \rightarrow \max
\]
under the constraints
\[ 0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, l. \]
Here, \( K \) is the kernel and the values \( \alpha_i, i = 1, \ldots, l, \) are the Lagrange multipliers corresponding to the training vectors. For each non-zero \( \alpha_i \) the corresponding vector \( x_i \) is called a support vector. There is empirical evidence that the number of support vectors is typically a small fraction, 3–5%, of the training set. If \( x \) is a new vector, the prediction \( \hat{y} \) is
\[
\hat{y} = \text{sign} \left( \sum_{i=1}^{l} \alpha_i y_i K(x_i, x) + b \right).
\]
We can see that the dual approach allows the construction of hyperplanes even in spaces with very a high number of dimensions. Computer experiments with real-world and standard benchmark data sets show that the SVM technique provides us with an excellent tool for making predictions.

2.2. Limitations

Unfortunately, the SVM method provides us with just “bare” predictions without estimating confidence in those predictions. Although there are several theorems that allow us to assess the performance of the algorithm, they typically are not directly applicable in practice. Let us consider, e.g., Theorem 10.5 on p. 414 in [10], which bounds the probability of an error on one test example in terms of the expectation of the number of support vectors:
\[
\text{prob}\{y_{i+1} \neq \hat{y}_{i+1}\} \leq \frac{E(#SV_{i+1})}{l+1},
\]
where \( #SV_{i+1} \) is the number of (essential) support vectors among \( l+1 \) random examples (and prob means probability over random choices of both the test example and the training set). But this formula does not allow us to estimate the probability of an error
from the available data (the training set): to estimate \( E(\#SV_{t+1}) \) one needs to perform extra experiments (another possibility is to estimate it from the given training set, but such an estimate will inevitably be very crude).

Many other machine learning algorithms simply do not have any measure of confidence, or have bounds on errors (e.g., those in PAC theory) that are too crude to be useful in practice. Indeed, as has been shown in [6], PAC analysis, can provide bounds on probability of error, but these return values > 1.

Bayesian approaches, on the other hand, can give strong confidence bounds but require one to make a priori assumptions about the data’s distribution. If these assumptions do not reflect the true distribution the bounds will not be valid [4,5].

Our goal is therefore to complement bare predictions with some measure of confidence which is valid under the general iid assumption and applicable in practice. As it happens, the problem of assigning confidence to predictions is closely connected to the problem of defining random sequences, and we next look at the basic ideas of randomness.

3. Randomness

The idea of using algorithmic randomness for predicting \( y_{t+1} \) is as follows: for every possible value \( Y \) of \( y_{t+1} \) we estimate the “randomness” (or “typicalness”) of the sequence \((x_1, y_1), \ldots, (x_t, y_t), (x_{t+1}, Y)\) with respect to the iid model; we can make a confident prediction if and only if exactly one of these two sequences is typical.

The “universal” notion of randomness was defined by Kolmogorov, Martin-Löf and Levin (see, e.g., [3]) based on the existence of the Universal Turing Machine. For finite sequences there is no clear-cut difference between typical and untypical sequences (all sequences are typical but to a different degree), so one needs a “randomness deficiency”, or “randomness level”. The definition of randomness deficiency [3] is in fact a universal version of the standard statistical notion of \( p \)-values.

3.1. Definitions

Let \( Z \) be the set of all possible labelled examples; \( Z^* \) is the set of all finite sequences of labelled examples.

**Definition.** A function \( f: Z^* \rightarrow [0, \infty) \) is a *randomness test* if

1. for all \( r \geq 0 \), all \( n \in \{1, 2, \ldots\} \) and all probability distributions \( P \) in \( Z \), \( P^n\{z \in Z^n: f(z) \leq r\} \leq r \);
2. \( f \) is upper semicomputable.

The first condition means that the randomness test is required to be valid: if, for example, we observe \( f \leq 1\% \) on our data set, then either the data set is not generated by the iid model or a rare (of probability 1\%) event has occurred. The second condition means that we should be able to compute the test, in some weak sense (we cannot
require computability in the usual sense, since the universal test can only be upper semicomputable: it can work forever to discover all regularities in the data).

**Basic Lemma** (Kolmogorov, Martin-Löf, Levin). There exists a smallest, to within a constant factor, randomness test (the universal test).

We fix a universal randomness test and call the value it takes on a data sequence the *randomness level* of this sequence. Randomness level is not computable; it is a number between 0 and 1, and when it is close to 0 the sequence is untypical.

### 3.2. Algorithm for prediction with confidence and credibility

Let us imagine that the randomness level can be computed (for example, we can ask an oracle questions about its values); in this subsection we describe the optimal algorithm for making predictions complemented by some measures of confidence and credibility (the latter will be discussed below).

Assuming that we have training set \((x_1, y_1), \ldots, (x_l, y_l)\) and test set \(x_{l+1}, \ldots, x_{l+k}\) (usually \(k = 1\)) and that our goal is to predict the classifications \(y_{l+1}, \ldots, y_{l+k}\) for \(x_{l+1}, \ldots, x_{l+k}\), we can act as follows:

1. Consider all possible values \(Y_1, \ldots, Y_k\) for labels \(y_{l+1}, \ldots, y_{l+k}\) and compute (in practice, approximate from above) the randomness level of every possible completion 
   \[
   (x_1, y_1), \ldots, (x_l, y_l), (x_{l+1}, Y_1), \ldots, (x_{l+k}, Y_k).
   \]
2. Predict the set \(Y_1, \ldots, Y_k\) corresponding to the completion with the largest randomness level.
3. Output as the *confidence* in this prediction one minus the second largest randomness level.
4. Output as the *credibility* of this prediction the randomness level of the output prediction \(Y_1, \ldots, Y_k\) (i.e., the largest randomness level for all possible predictions).

To understand the intuition behind confidence, let us tentatively choose a conventional “significance level” such as 1%. If the confidence in our prediction exceeds 99% and the prediction is wrong, the actual data sequence belongs to an a priori chosen set of probability \(< 1\%\) (namely, the set of all data sequences with randomness level \(< 1\%\)).

Intuitively, low credibility means that either the training set is non-random or the test examples are not representative of the training set (say, in the training set we have images of digits and in the test set we have those of letters).

### 3.3. Practical approximations

As we said before, the randomness level is non-computable. A powerful way to approximate it is to use the SVM. With every possible label \(Y \in \{-1, 1\}\) for \(x_{l+1}\) we associate the SVM optimization problem for the \(l+1\) examples (the training examples plus the test example labelled with \(Y\)). The solutions (Lagrange multipliers) \(\alpha_1, \alpha_2, \ldots, \alpha_{l+1}\) to this problem reflect the “strangeness” of the examples (\(\alpha_i\) being the
strangeness of \((x_i, y_i), i = 1, \ldots, l\), and \(x_{l+1}\) being the strangeness of the \((x_{l+1}, Y)\). All \(x_i\) are non-negative and, in practice, only few of them are different from zero (the support vectors). An easily computable approximation to the randomness level is given by the \(p\)-values associated with every completion \((x_1, y_1), \ldots, (x_l, y_l), (x_{l+1}, Y)\):

\[
\frac{\#\{i : x_i \geq x_{l+1}\}}{l + 1}
\]

in words, the \(p\)-value is the proportion of \(x\)’s which are at least as large as the last \(x\). It easy to show that these \(p\)-values are valid in the sense that they define a randomness test.

3.4. Approximations with the nearest neighbours algorithm

So far we assumed that the “strangeness values” \(x_i\) used to approximate randomness level are obtained from the SVM algorithm. However, we can get useful \(x\)’s from many other learning algorithms; for example the nearest neighbours algorithm \([7]\) also provides good approximations to the randomness level. We can set \(x\)’s in the following way:

\[
x_i := \frac{\sum_{j=1}^{k} d_{ij}^+}{\sum_{j=1}^{k} d_{ij}^-},
\]

where \(d_{ij}^+\) is the \(j\)th shortest distance from \(x_i\) to other examples classified as \(x_i\), and \(d_{ij}^-\) is the \(j\)th shortest distance from \(x_i\) to the examples classified differently from \(x_i\).

4. Experiments and results

The experiments were conducted on a well-known pattern recognition problem of classifying handwritten digits. A database of US postal data of 9298 digits was used in the experiments, and each digit was a \(16 \times 16\) matrix \([2]\). Several kernels were used, and the results have shown that the method works well in predicting classifications. In addition, of course, this method also provides valid measures of confidence and credibility for predictions. Table 1 illustrates the results for several examples with \(p\)-values for each possible classification, the actual classification, and the predicted classification with confidence and credibility; the polynomial kernel \(K(x, x') = (x \cdot x')^5/256\) is used.

To interpret the numbers in Table 1, remember that high (i.e., close to 100%) confidence means that all classifications except the predicted one are unlikely. If, say, the first example were classified wrongly, this would mean that a rare event (of probability <1%) had occurred; therefore, we expect the prediction to be correct (which it is). In the case of the second example, confidence is also quite high (>95%), but we can see that the credibility is low (<5%). From the confidence we can conclude that the labels other than 4 are excluded at level 5%, but the label 4 itself is also excluded at the level 5%. This shows that the prediction algorithm was unable to extract from the training set enough information to allow us to confidently classify this example: the
strangeness of the labels different from 4 is mainly due to the fact that the unlabelled example itself is strange; perhaps the test example is very different from all examples in the training set. Unsurprisingly, the prediction for the second example is wrong.

In general, high confidence shows that all alternatives to the predicted classification are unlikely. Low credibility means that the whole situation is suspect; as we have already mentioned, we will obtain a very low credibility if the new example is a letter (whereas all training examples are digits). Credibility will also be low if the new example is a digit written in an unusual way. Notice that typically credibility will not be low provided the data is generated by an iid mechanism: the probability that credibility will be less than some threshold $\gamma$ (such as 1%) is $<\gamma$.

To summarize, we can trust a prediction if (1) the confidence is close to 100% and (2) the credibility is not low (say, is not $<5\%$). Table 1 gives credibility values typical when using the SVM for computing $p$-values: credibility is exactly 100% on a few occasions. This happens because most of the $\alpha$’s computed by the SVM are 0s. For many other learning methods typical values of credibility are in the range 5–95%.

Another set of experiments included the $\alpha$’s obtained from the nearest neighbours algorithm. Applying formula for calculating $p$-values gives the overall accuracy of 2.2% on the same US postal data where 7291 digits were used for training set, and the remaining 2007 for testing. We are also able to provide some valid measures of confidence; for example, we can say that we also can state that 95% of the test examples give very confident (confidence $\geq 99\%$) predictions.

5. Conclusion

This paper shows the connections between the algorithmic theory of randomness and the problem of giving confident predictions in machine learning. In addition to “bare” predictions, it is also possible to obtain valid and computable measures of confidence which work well in practice.
The algorithm described here is a transductive algorithm, in the sense that it outputs only the predicted label of a particular new example rather than a general inductive rule for classifying future examples [1]. An important class of transductive algorithms is provided by the theory of instance-based, or case-based, learning. Perhaps the most well-known algorithm in this class is the $k$-nearest neighbours algorithm. This paper shows that transduction is useful not just as an efficient means for producing bare predictions but also as a way to obtain confidence and credibility associated with the output predictions.

The method described in this paper has been extended to regression problems, where the classifications $y_i$ are real values. The required $p$-values can be obtained from, for example, the SVM for regression [10] or the Ridge Regression procedure.

One of the most interesting results that follow from the algorithmic theory of randomness is that some problems are impossible to solve under the general iid assumption. For example, on-line prediction under the iid assumption is impossible. Similarly, the problem of density estimation can only be solved under stronger assumptions than the iid assumption. For precise definitions and details of these impossibility results, see [11]. In principle, once “positive results” (such as the prediction algorithms described in this paper) are obtained, it is possible to eliminate the algorithmic theory of randomness from them (for the price of some loss in the clearness of intuitive motivation); for the negative results (such as the impossibility results mentioned above) the algorithmic theory of randomness becomes indispensable.

The work on applying algorithmic theory of randomness to practical problems of computer learning is under way at the Computer Learning Research Centre; the reader can consult the web page http://www.clrc.rhul.ac.uk.

Acknowledgements

This research is partially supported by EPSRC through Grants GR/L35812 (“Support Vector and Bayesian learning algorithms”), GR/M14937 (“Predictive complexity: recursion-theoretic variants”) and GR/M16856 (“Comparison of Support Vector Machine and Minimum Message Length methods for induction and prediction”).

References


