Computationally Efficient Transductive Machines

Craig Saunders, Alex Gammerman, Volodya Vovk

Royal Holloway, University of London,
Egham, Surrey, England, TW20 0EX
{craig,alex,vovk}@dcs.rhbnc.ac.uk

Abstract. In this paper[1] we propose a new algorithm for providing confidence and credibility values for predictions on a multi-class pattern recognition problem which uses Support Vector machines in its implementation. Previous algorithms which have been proposed to achieve this are very processing intensive and are only practical for small data sets. We present here a method which overcomes these limitations and can deal with larger data sets (such as the US Postal Service database). The measures of confidence and credibility given by the algorithm are shown empirically to reflect the quality of the predictions obtained by the algorithm, and are comparable to those given by the less computationally efficient method. In addition to this the overall performance of the algorithm is shown to be comparable to other techniques (such as standard Support Vector machines), which simply give flat predictions and do not provide the extra confidence/credibility measures.

1 Introduction

Many risk-sensitive applications such as medical diagnosis, or financial analysis require predictions to be qualified with some measure of confidence. Indeed in general, any predictive machine-learning algorithm which requires human-computer interaction, often benefits from giving qualified predictions. The usability of the system is improved, and predictions with low confidence can be filtered out and processed in a different manner. In this paper we have two aims: firstly, we wish to provide confidence and credibility values for our predictions, rather than the simple "flat" answer given by many Machine Learning techniques (such as a standard Support Vector Machine [10]); secondly we want to obtain these values in an efficient manner so that the algorithm is practical for large data sets, and does not suffer the time penalties of previously proposed algorithms (e.g. those in [1, 7]). To achieve the confidence and credibility measures, we build on ideas of algorithmic information theory (see [12]). By using these ideas, we are able to provide confidence measures with a strong theoretical foundation, and which do not rely on stronger assumptions than the standard i.i.d. one (we actually make a slightly weaker assumption, that of *exchangeability*). This is in contrast to many alternative methods (such as the Bayesian approach), which often require a prior probability (which is not known and has to be estimated), and confidence measures are given on the assumption that this prior is the correct one. In order to compute these values we use Support Vector Machines and the statistical notion of p-values, in an extension of the ideas presented in [7]. The multi-class method presented in that exposition however, was processing-intensive, and the length of time required meant that the algorithm was not practical for medium to large datasets. The method presented here (and originated in [11]) however, overcomes these difficulties, and in section 4 experiments are conducted on much larger data sets (e.g. 7900 training, 2000 test). The layout of this paper is as follows. In section 2 we describe the theoretical motivation for the algorithm, then in section 3 we concentrate on a specific implementation which uses Support Vector machines. In this section we briefly describe a previous method of qualifying Support Vector method predictions, and extend the technique to the multi-class case. The inefficiencies of this method are presented, and a new algorithm is proposed. Experimental evidence is presented in section 4 which indicates that as well as providing confidence and credibility values, the algorithm's predictive performance is comparable to a standard Support Vector machine when

[1] Published in the Proceedings of the Eleventh International Conference on algorithmic Learning Theory 2000 (ALT '00), Lecture Notes in Artificial Intelligence, Springer-Verlag, 2000

using the same kernel function. Specifically, experiments were carried out on the US Postal Service digit database, and a comparison is made between the new algorithm, the algorithm presented in [7], and a standard Support Vector Machine. In section 5 we discuss the merits of this approach and suggest future directions of research.

2 Randomness

In [12] it was shown that approximations to universal confidence measures can be computed, and used successfully as a basis for machine learning. In this section we present a summary of the relevant ideas, which will provide a motivation for the technique described in section 3. What we are principally interested in is the randomness of a sequence $z = (z_1, \dots, z_n)$ of elements of $z_i \in Z$ where Z is some sample space (for the applications presented in this paper, z is a sequence $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_l, y_l), (\mathbf{x}_{l+1}, y_{l+1})$ where $\mathbf{x}_i \in \mathrm{I\!R}^n, y \in \mathbb{Z}$, containing l training examples and one test example along with some provisional classification). Let $\mathcal{P} = \mathcal{P}_1, \mathcal{P}_2, \dots$ be a sequence of statistical models such that, for every $n = 1, 2, \dots$, $\mathcal{P}_n$ is a set of probability distributions in Z^n. In this paper we will only be interested in specific computable $\mathcal{P}$ (namely, the iid and exchangeability models). We say that a function $t : Z^* \to \mathbb{N}$ (where $\mathbb{N}$ is the set $\{0, 1, \dots\}$ of non-negative integers) is a *log-test for $\mathcal{P}$-typicalness* if

1. for all $n \in \mathbb{N}$ and $m \in \mathbb{N}$ and all $P \in \mathcal{P}_n$, $P\{z \in Z^n : t(z) \geq m\} \leq 2^{-m}$.
2. t is semi-computable from below.

As proven by Kolmogorov and Martin-Löf (1996) (see also [4]), there exists a largest, to within an additive constant, log-test for $\mathcal{P}$-randomness, which is called *$\mathcal{P}$-randomness deficiency.* When $\mathcal{P}_n$ consists of all probability distributions of the type P^n, P being a probability distribution in Z, we omit "$\mathcal{P}$-" and speak of just *randomness deficiency.* If $d(z)$ is the randomness deficiency of a data sequence z, we call $\delta(z) = 2^{-d(z)}$ the *randomness level* of z. The randomness level δ is the smallest, to within a constant factor, p-value function; the latter notion is defined as follows: a function $t : Z^* \to [0, 1])$ is a *p-value function w.r.t. the iid model* if

1. for all $n \in \mathbb{N}$ and $r \in [0, 1]$ and all distributions $P \in Z$,
   $P^n\{z \in Z^n : t(z) \leq r\} \leq r$. (1)
2. t must be semi-computable from above.

The randomness level is a universal measure of typicalness with respect to the class of iid distributions: if the randomness level of z is close to 0, z is untypical. Functions t which satisfy the above requirement are called *p-typicalness tests.*

2.1 Using Randomness

Unfortunately, this measure of typicalness is non-computable (and in practice one has to use particular, easily computable, p-value functions). If however one could compute the randomness deficiency of a sequence and we accept the iid assumption and ignore computation time, then the problem of prediction would become trivial. Assuming we have a training set $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_l, y_l)$ and an unlabelled test example $\mathbf{x}_{l+1}$, we can do the following:

1. Consider all possible values Y for the label y_{l+1}, and compute the randomness level of every possible completion
$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_l, y_l), (\mathbf{x}_{l+1}, Y)$$
2. Predict Y 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* the randomness level of the prediction.

The intuition behind confidence can be described with the following example. Suppose we choose a "significance level" of 1%. If the confidence in our prediction exceeds 99% and we are wrong, then the actual data sequence belongs to the set of all data sequences with randomness level less than 1%, (which by (1) is a very rare event). Credibility can be seen as a measure of quality of our data set. Low credibility means that either the training set is non-random or the test example is not representative of the test set.

2.2 Use in Practice

In order to use these ideas in practice, we will associate a strangeness measure with each element in our extended training sequence (denoted α_i). If we have a strangeness measure which is invariant w.r.t. permutation of our data, the probability of our test example being the strangest in the sequence is $\frac{1}{l+1}$. Because all permutations of strangeness measures are equiprobable, we can generalise this into a valid p-typicalness function :

$$t(z) = \frac{\#\{i : \alpha_i \geq \alpha_{l+1}\}}{l+1}.$$

This is the type of function we will use in order to approximate the randomness level of a sequence. In this paper, our strangeness measures (α_i) are constructed from the Lagrange multipliers of the SV optimisation problem, or the distances of examples from a hyperplane.

3 SV Implementation

In this section we describe a way of computing confidence and credibility values which uses Support Vector Machines. We first describe and extend the method outlined in [7] to the multi-class case. The new method presented later in this section is more computationally efficient than the one presented in [7] (for timings see section 4), allowing much larger datasets to be used.

3.1 Original Method

In [7], a method for two-class classification problems was presented. The method involved adding a test example to the training set, along with a provisional classification (say -1). A Support Vector machine was then trained on this extended set, and the resultant Lagrange multipliers were used as a strangeness measure. That is the following optimisation problem was solved :

$$\max \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1,\ldots,l+1} \alpha_i \alpha_j y_i y_j \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j),$$

subject to the constraints,

$$\sum_{i=1,\ldots,l+1} \alpha_i = 0, \ \alpha_i \geq 0, \ i = 1, \ldots, l+1. \tag{1}$$

The p-typicalness function took the form :

$$p_- = \frac{\#\{i : \alpha_i \geq \alpha_{l+1}\}}{l+1}.$$

The test example was then added to the training set with a provisional classification of $+1$, and p_+ was calculated in a similar fashion. Confidence and credibility were then calculated as outlined in section 2.1.

Extension to Multi-Class Problems The method above can easily be extended to the multi-class case. Consider an n-class pattern recognition problem. This time, for each test example, n optimisation problems have to be solved (one for each possible classification). We generate n "one against the rest" classifiers, each time using the resultant α-values to calculate p-typicalness as follows. For each class $m \in \{1, \ldots, n\}$, train an m-against-the-rest Support Vector machine, and calculate p_m as :

$$p_m = \frac{\#\{i : (\alpha_i \geq \alpha_{l+1}) \wedge (y_i = m)\}}{|S_m|},$$

where

$$S_m = \{(\mathbf{x}_i, y_i) : y_i = m)\}.$$

That is, for each classifier, we only use the α-values which correspond to the provisional classification given, in our calculation of p-typicalness. Unfortunately, although this method works in practice, it is rather inefficient and can only be used on small data sets. Consider as an example of a medium-large problem, the well known 10-class digit recognition problem of the US Postal Service data set. To train a single "one vs. the rest" SV machine on this data set takes approximately 2 minutes. Therefore, to use the above method to classify a test set of 2000 examples, it would take approximately $2 \times 10 \times 2007 = 40140$ minutes. Which is roughly 1 month! Clearly this is unacceptable, and an improvement has to be found.

3.2 New Method

The general idea is as follows; we create a hash function $f_h : \mathbb{R}^d \rightarrow \{1, \ldots, h\}$, which when given a training vector $\mathbf{x}_i$, returns a value in the range $\{1, \ldots, h\}$. This is used to create a total of $h * n$ subsets of our training data (where n is the number of classes in our training set). For each class in the training set, a Support Vector Machine is trained in the following way. For every possible output of the hash function j, train a Support Vector Machine each time leaving out of the training process those examples which both are a member of the class being considered, and return a value of j from the hash function. More formally, we have the following. We are given a training set T which consists of l examples and their labels $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$, where $\mathbf{x}_k \in \mathbb{R}^d$ and $y_k \in \{1, \ldots, n\}$. We also have a hash function $f_h : \mathbb{R}^d \rightarrow \{1, \ldots, h\}$. Note that the hash function should be chosen so that it is "pseudo-random" and splits the training set into roughly equal portions. The hash function used in the experiments in this paper simply computed the sum of all attribute values modulo h plus 1. First of all we create nh sets $S_{i,j}$ from our training set

$$S_{i,j} = \{(\mathbf{x}_k, 1) : y_k = i, f_h(\mathbf{x}_k) \neq j\} \cup \{(\mathbf{x}_k, -1) : y_k \neq i\}, \tag{2}$$

where $i = 1, \ldots, n$ and $j = 1, \ldots, h$. On each of these sets we train a Support Vector Machine. That is, we obtain hn functions of the form

$$F_{i,j}(\mathbf{x}) = \sum_{k:(\mathbf{x}_k, y_k) \in S_{i,j}} \alpha_k y_k \mathcal{K}(\mathbf{x}_k, \mathbf{x}),$$

where $\mathcal{K}$ is some kernel function, and the α_i's are obtained by solving the following optimisation problems; maximise

$$\sum_{k=1}^{l} \alpha_k - \frac{1}{2} \sum_{k,m:(\mathbf{x}_k, y_k),(\mathbf{x}_m, y_m) \in S_{i,j}} \alpha_k \alpha_m y_k y_m \mathcal{K}(\mathbf{x}_k, \mathbf{x}_m),$$

subject to the constraints,

$$\sum_{k:(\mathbf{x}_k, y_k) \in S_{i,j}} y_k \alpha_k = 0, \quad \alpha_k \geq 0, \quad k = 1, \ldots, |S_{i,j}|.$$

This is similar to the "one against the rest" method which is often used in multi-class Support Vector Machines [9]. For our purposes though, we create several "one against the rest" classifiers for every class, each time only including positive examples which have a particular value when the hash function is applied.

3.3 Classification, Confidence, and Credibility

The procedure for classifying a new test example is given by Algorithm 1. In a nutshell the procedure simply applies the hash function to some new example $\mathbf{x}_{\text{new}}$, then for each class identifies a working set (denoted W_i) and a particular function $F_{i,j}$ (which did not use any element of the working set in its creation). The function $F_{i,j}$ is then used to obtain the distance to the hyperplane for each element of the working set, and our new example (these distances are denoted by $d_1, \ldots, d_{|W_i|}, d_{\text{new}}$). Note that "distance" here is defined as the output of a function $F_{i,j}(\mathbf{x})$, and therefore can be negative (if the point $\mathbf{x}$ lies on a specific side of the hyperplane). In order to give

Algorithm 1 Classifying a new test sample $\mathbf{x}_{\text{new}}$

```
Obtain $j_{\text{new}} = f_h(\mathbf{x}_{\text{new}})$.
for Each class $i$ in training set do
    Create a working set $W_i$ which includes all examples in the training set with $y_k = i$ and $f_h(\mathbf{x}_k) = j_{\text{new}}$ (i.e. $W_i = \{x : f_h(\mathbf{x}_k) = j_{\text{new}}, y_k = i, k = 1, \ldots, l\}$).
    For every example in $W_i$ and $\mathbf{x}_{\text{new}}$ use $F_{i,j_{\text{new}}}$ (see eq (2)) to get the distance $d_k$ from the hyperplane.
    Compute p-value ($p_i$) for new example, where $p_i = \frac{\#\{k : d_k \leq d_{\text{new}}\}}{|W_i|+1}$
end for
Predicted classification is $\arg\max_i p_i$.
Confidence in prediction is $1 - \max_{j \neq i} p_j$.
Credibility of prediction is $\max_i p_i$.
```

confidence and credibility values for the new example, we compute the example's p-value for each possible classification. Once the distances $d_1, \ldots, d_{|W_i|}, d_{\text{new}}$ to the hyperplane for a particular working set W_i (including our new test example) have been calculated, the p-value is simple to compute. The ideal situation is where our new example is the "strangest" example of the working set. For this algorithm the strangest example is the one with the smallest distance to the hyperplane (recall that "distance" in this sense can be negative, so the smallest d_k is either the example furthest on the "wrong" side of the hyperplane for classification c, or if all examples are on the positive side, the example closest to the hyperplane). The probability that our example $\mathbf{x}_{\text{new}}$ has the smallest valued distance to the hyperplane out of all examples in the working set is simply

$$P\left\{ d_{\text{new}} < \min_{1 \leq k \leq |W_i|} d_k \right\} \leq \frac{1}{|W_i| + 1},$$

(since all permutations of $d_1, \ldots, d_{|W_i|}, d_{\text{new}}$ are equiprobable). The distances from the hyperplane are a valid strangeness measure (i.e. they are invariant under permutation), so we can construct a valid p-typicalness function as follows :

$$p_i = \frac{\#\{k : d_k \leq d_{\text{new}}\}}{|W_i| + 1}.$$

As stated in Algorithm 1, our prediction for $\mathbf{x}_{\text{new}}$ is given by the classification which yielded the highest p-value. In an ideal case, the p-value associated with the correct classification will be high, say $\geq 95\%$, and for all other classifications it will be low, say $\leq 5\%$. In this case both confidence and credibility will be high and our prediction is deemed to be reliable. If however the example looks very strange when given all possible classifications (i.e. the highest p-value is low, e.g. $\leq 10\%$), then although confidence may be high (all other p-values may still be $\leq 5\%$), our credibility will be low. The intuition here would be: although we are confident in our prediction (the likelihood of it being another candidate is low), the quality of the data upon which we base this prediction is also low, so we can still make an error. This would concur with the intuition in section 2. In this situation our test example may not be represented by the training set (in our experiments this would correspond to a disfigured digit).

4 Experiments and Results

Experiments were conducted on the well known benchmark USPS database (see e.g. [3]), which consists of 7291 training examples and 2007 test examples, where each example is a 16×16 pixelated image of a digit in the range 0–9. For all these experiments, the following kernel was used

$$\mathcal{K}(\mathbf{x}, \mathbf{y}) = \frac{(\mathbf{x} \cdot \mathbf{y})^3}{256}.$$

Although this kernel does not give the best possible performance on the data set, it is comparable and is only meant to ensure that a comparison between the techniques presented here is a fair one.

4.1 Efficiency Comparison

In order to compare this method to the one presented in [7], we conducted an experiment on a subset of the USPS data set. All examples of the digits 2 and 7 were extracted from the data set creating a two-class pattern recognition problem with 1376 training examples and 345 test examples. Table 1 shows the timings and error rates for both methods[2]. Note that a normal Support Vector machine also has 3 errors on this data set (when trained with the same kernel function). Also in this case, the 3 errors produced by the SV machine and the two transductive methods were the same 3 examples. For the new method the range of values which the hash function can produce (h), can be changed. The value of h determines how many subsets each class in the training set is split into, and results are shown for h = 2, 3, and 4. Even though the data set in this experiment would not normally be considered to be large, the previous method suffers a heavy time penalty. The table clearly shows that the method proposed in this paper is more efficient, whilst retaining the same level of performance. In order to interpret the last column of the table, notice that a -log p-value of 2 indicates a p-value of 1%. The gap in efficiency between the two methods is due to the fact that the new method does not have to run two optimisation problems for each test point. If the number of test examples is increased, the time taken by the hashing method does not alter significantly. The old method however, scales badly with any such increase. In order to illustrate this in practice we used a subset of the data described above. A total of 400 examples were used for training, and two test set sizes were used: 100 examples and 345 examples. Table 4.1 shows the error rates and timings of the old method, and the hashing method with 3 hash sets. Notice the time penalty incurred by the old method as the test set is expanded.

Method	Time	Errors	ave -log p-value
Old	5 hrs 20 mins	3	3.06
2 Splits	39 secs	4	2.51
3 Splits	50 secs	3	2.33
4 Splits	1 min 4 secs	3	2.20

Table 1. Timings, errors (out of 345), and average -log (base 10) p-values for the different methods, on a 2-class subset of the USPS data set. Note that large average p-values are preferable (see section 4.2)

4.2 Predictive Performance of the Algorithm

Experiments were also conducted on the full USPS data set, and the performance of the algorithm was measured when each class was split into different numbers of subsets. Table 2 summarises these results. In the case of having 5 splits, the performance of the algorithm deteriorated. This could be due to the fact that although by having 5 splits the training set was larger and therefore one would

[2] Note that for the experiments we used the SVM implementation from Royal Holloway. See [8] for details.

Method	Time (100 examples)	Time (345 examples)
Old	11 mins 37 secs (0 errors)	39 mins 16 secs (5 errors)
3 Splits	12 secs (0 errors)	13 secs (6 errors)

Table 2. Timings and error rates for the two methods. The training set size was 400, and two test sets of size 100 and 345 were used. The old algorithm suffers a heavy time penalty with the increase in test set size.

expect a better decision function, the working set is greatly reduced in size. This led to the p-values for many classes being of the same magnitude and would therefore result in more misclassifications. As a point of comparison for the results shown in table 2, note that the Support Vector Machine

No of Splits	Error Rate	ave -log p-value
2	5.7%	2.46
3	5.5%	2.23
4	5.4%	2.04
5	6.0%	1.91

Table 3. Error rates for different numbers of splits of each class; the last column gives the average minus log p-value over all incorrect classifications. The data set used was the 10-class USPS data set.

when using the same kernel has an error rate of 4.3%. Although for the smaller data set used in the previous section the performance of the new method, the original transductive method, and the Support Vector machine was identical, our quest for efficiency on a large data set has resulted in a small loss in performance in this case. Our aim though is to produce valid confidence and credibility values whilst retaining good performance, we are not necessarily trying to outperform all other methods. The table shows that the performance of the algorithm does not suffer to a large extent, even though it provides the extra measures. The last column in the table shows the average minus log of p-values calculated for the incorrect classifications of the new example. For relatively noise-free data sets we expect this figure to be high, and our predictive performance to be good. This can also be interpreted as a measure of the quality of our approximation to the actual level of randomness, the higher the number, the better our approximation. This is our main aim: to improve the p-values produced by the algorithm. We believe that good predictive performance will be achieved as our p-values improve. This can already be seen in the progression from the algorithm presented in [1]. Our algorithm provides better confidence and credibility[3] values, and our predictive performance is also higher. When comparing p-values in the tables it is important to note that there is an upper bound on the ave -log p-value which can be obtained. This stems from the fact that even if every incorrect classification is highlighted by the algorithm as the strangest possible, then the p-value is restricted by the sample size from which it is obtained. As an example, consider the p-values obtained in table 1. For the old method, the strangeness measure was taken over the whole training set (approx. 1300 examples). This would yield a maximum average (-log p-value) of 3.11. For hashing however, we are restricted to computing p-typicalness functions over the hash set. For 3 splits, each hash set contains roughly 225 examples. This would yield a maximum average of 2.34. For larger data sets, we would therefore hope that this figure qould improve (as the hash set size would increase).

4.3 Confidence and Credibility Values

For the experiments, the confidence in our predictions was typically very high, 85–99%. This was due to the data set being relatively noise free. In a data set corrupted by noise, we would expect the prediction not to be so clear cut. That is, the noise in the data may make another classification

[3] In the paper, the measure of credibility was referred to as possibility.

(other than correct one) appear to be random. The correct classification may have a large p-value (95%), and therefore may clearly be one we predict. The confidence in the prediction however, will be lower. Our intuition behind the measure of credibility was that it should reflect the "quality" of our predictions. If credibility is low, then the example looks strange for every possible classification, and so our prediction is not as reliable. It is therefore expected that the credibility associated with a prediction which is later found to be incorrect, should be low in a majority of cases. This has been observed experimentally and is illustrated by Figure 1, which displays histograms showing the number of incorrect predictions which have credibility within a certain range for 2,3 and 4 splits.

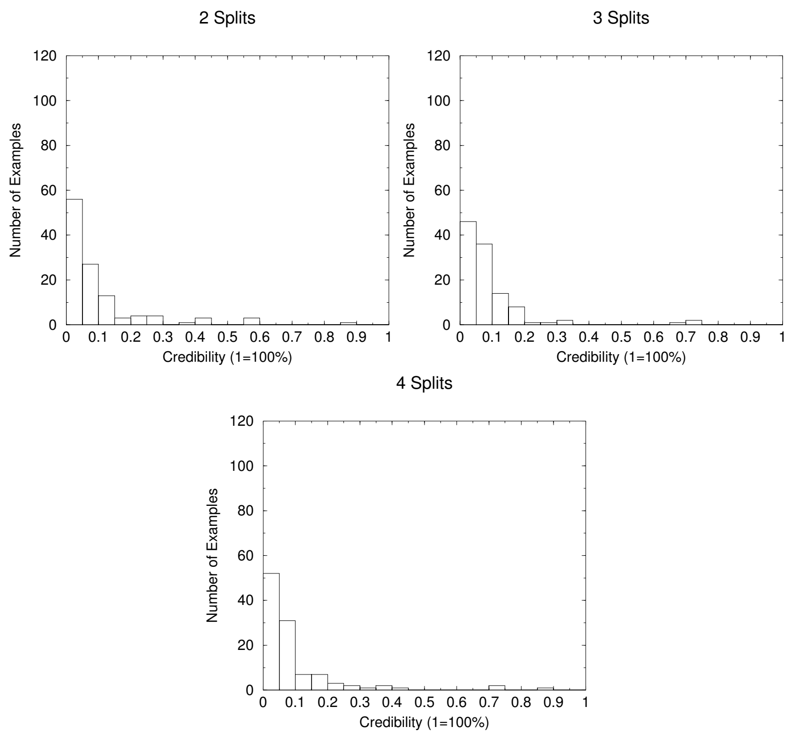

Fig. 1. Credibility values for incorrectly predicted examples, when run with different numbers of splits.

4.4 Rejecting Examples

It is possible to use the measures of confidence and credibility to obtain a rejection criteria for difficult examples. Suppose we pick a specific confidence threshold, say 95%, and reject all predictions which fall below this level. We can then expect that the error rate on the remaining predictions will not deviate significantly from at most 5%. Note that over randomisations of the training set and the test example, and over time, we would expect the error rate to be $\leq 5\%$ (over all examples). In this scenario however, we have a fixed (but large) training set. Also, we are measuring the error over the non-rejected examples and not the whole set. If a small number of examples are rejected however, we would not expect the error rate to deviate significantly from 5%. Unfortunately, it is not possible to say a-priori how many examples will be rejected. For our experiments have selected four possible rejection criteria, these are : Confidence, Credibility, Confidence $\times$ Credibility and $(1 -$ Confidence$) -$ Credibility. The first measure is obvious - we want to reject all classifications which do not achieve a certain confidence value, therefore capping the generalisation error. The other measures however, also control generalisation error. We may wish to reject examples with low credibility; that is, those examples which look unlikely given any classification. Thirdly, by simply taking the product of the two measures, we end up with a single measure which is only high when both values are high. Finally, the difference between typicalness values of the two likeliest classifications can be used. Again, this is an attempt to reject samples which do not have a clear leading candidate for the correct classification. The rejection rate vs. generalisation error on non-rejected examples is plotted for hash sizes 2,3,4 and 5, and are shown in figure 2.

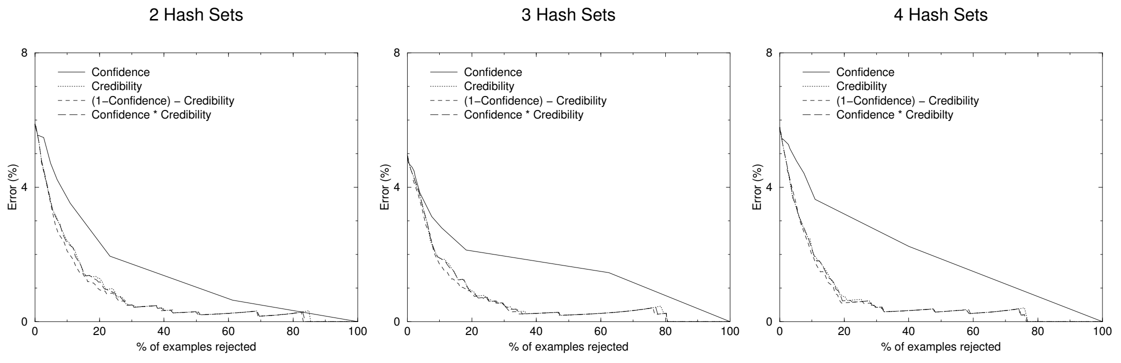

Fig. 2. Generalisation error on non-rejected examples vs. rejection rate.

5 Discussion

In this paper we have presented an algorithm which gives both confidence and credibility values for its predictions, on a multi-class pattern recognition problem. This method overcomes the time penalties suffered by a previously proposed algorithm, whilst retaining a comparable level of performance. This allows the method to be used on large real-world data sets. Empirical evidence has been presented which indicates that the confidence and credibility values produced by the algorithm correctly reflect confidence in the prediction and the quality of the data upon which it was based. Furthermore, in addition to providing confidence and credibility values, the performance of the algorithm has been shown to be comparable to that of Support Vector machines. The work here concentrates on pattern recognition problems, but can easily be extended to regression estimation. Both Support Vector Machine regression, and methods such as Ridge Regression (see e.g. [2], or [6] for the kernel-based version) can be extended to incorporate the ideas in this paper.

Acknowledgements

This work was partially supported by EPSRC GR/L35812 and GR/M15972, and EU INTAS-93-725-ext grants. In addition we are indebted to the support provided by IFR Ltd.

References

1. A. Gammerman, V. Vapnik, and V. Vovk. Learning by transduction. In *Uncertainty in Artificial Intelligence*, pages 148–155, 1998.
2. A. Hoerl and R.W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.
3. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. J. Jackel. "Handwritten digit recognition with back-propagation network". *Advances in Neural Information Processing Systems*, pages 396–404, 1990.
4. M. Li and P. Vitanyi. *An Introduction to Kolmogorov Compexity and Its Applications.* Springer, 1997.

5. P. Martin-Löf. The definition of random sequences. *Information and Control*, 1966.
6. C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *ICML '98. Proceedings of the 15th International Conference on Machine Learning*, pages 515–521. Morgan Kaufmann, 1998.
7. C. Saunders, A. Gammerman, and V. Vovk. Transduction with confidence and credibility. In *Proceedings of IJCAI'99*, volume 2, pages 722–726, 1999.
8. C. Saunders, M.O. Stitson, J. Weston, L. Bottou, B. Schölkopf, and A. Smola. Support Vector machine - reference manual. Technical Report CSD-TR-98-03, Royal Holloway, University of London, 1998.
9. B. Schölkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In *Proceedings, First International Conference on Knowledge Discovery and Data Mining*, pages 252–257. AAAI Press, 1995.
10. V. N. Vapnik. *Statistical Learning Theory.* Wiley, 1998.
11. V. Vovk and A. Gammerman. Algorithmic randomness theory and its applications in computer learning. Technical Report CLRC-TR-00-02, Royal Holloway, University of London, 1999.
12. V. Vovk, A. Gammerman, and C. Saunders. Machine-learning applications of algorithmic randomness. In *Proceedings of ICML '99*, pages 444–453, 1999.