The typicalness framework: a comparison with the Bayesian approach

Thomas Melluish Craig Saunders Ilia Nouretdinov
Volodya Vovk
{T.Melluish, C.Saunders, I.Nouretdinov, V.Vovk}@cs.rhul.ac.uk

Technical Report

CLRC-TR-01-05

November 2, 2001

Royal Holloway
University of London

Department of Computer Science
Egham, Surrey TW20 0EX, England

Abstract

When correct priors are known, Bayesian algorithms give optimal decisions, and accurate confidence values for predictions can be obtained. If the prior is incorrect however, these confidence values have no theoretical base – even though the algorithms' predictive performance may be good. There also exist many successful learning algorithms which only depend on the iid assumption. Often however they produce no confidence values for their predictions. Bayesian frameworks are often applied to these algorithms in order to obtain such values, however they can rely on unjustified priors. In this paper we outline the typicalness framework which can be used in conjunction with many other machine learning algorithms. The framework provides confidence information based only on the standard iid assumption and so is much more robust to different underlying data distributions. We show how the framework can be applied to existing algorithms. We also present experimental results which show that the typicalness approach performs close to Bayes when the prior is known to be correct. Unlike Bayes however, the method still gives accurate confidence values even when different data distributions are considered.

1 Introduction

In many real-world applications (such as risk-sensitive applications or those which rely on human-computer interaction) it is often desirable to obtain confidence values for any predictions that are given. In most cases these confidence values are used as a filter mechanism, whereby only those predictions in which the algorithm has a certain confidence are predicted; other examples are rejected or possibly simply abstained from and passed on to a human for judgement. Many machine learning algorithms for the problems of both pattern recognition and regression estimation give confidence levels, and the Bayesian framework is often used to obtain such values. When applying the Bayesian framework one has to assume the existence of a (often strong) prior, which for real-world data sets is often chosen arbitrarily. If an incorrect prior is assumed an algorithm may give 'incorrect' confidence levels, for example 95% confidence predictive regions may contain much less than 95% of the true labels. For real-world applications this is a major failure, as one would wish confidence levels to bound the number of expected errors.

We therefore desire confidence values to be valid in the following sense. Given some possible label space $\mathcal{Y}$, if an algorithm predicts some set of labels $R \subseteq \mathcal{Y}$ with confidence t for a new example which is truly labelled $y \in \mathcal{Y}$, then we would expect the following to hold over randomisations of the training set and the new example:

$$P(y \notin R) \leq 1 - t \tag{1}$$

Moreover, we prefer algorithms which give 'nearly precise' confidence values, that is values such that (1) approaches equality.

In this paper we outline the typicalness framework which can produce nearly precise confidence values for data which is independently and identically distributed. We compare methods within this framework to their Bayesian counterparts and show experimentally that when the correct prior is known the difference in performance of the two approaches can be negligible; when an incorrect prior is given (or benchmark data is used) however, the Bayesian algorithms give inaccurate confidences, whereas those using typicalness remain valid and even nearly precise.

The rest of this paper is laid out as follows. In Section 2 we describe the typicalness framework. Sections 3, 4, 5 and 6 detail algorithms for regression estimation and pattern recognition in both the Bayesian and typicalness frameworks. In Section 7 we present some experimental results and our conclusions are in Section 8.

It is often the case that many people have questions regarding the typicalness framework. In order therefore to try and clarify some points we include an appendix which contains some frequently asked questions and their answers.

2 The typicalness framework

Here we give a brief outline of the typicalness framework. For more details, see [22, 16, 13]. Consider a sequence of examples $(z_1, \ldots, z_l) = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l))$ drawn independently from the same distribution over $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ where $\mathcal{Y}$ is some label space. We can use the typicalness framework to gain confidence information for possible labelings for a new example $\mathbf{x}_{l+1}$. The idea is that we postulate some labels $\hat{y}_{l+1}$ and for each one we examine how likely it is that all elements of the extended sequence $((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l), (\mathbf{x}_{l+1}, \hat{y}_{l+1}))$ might have been drawn independently from the same distribution, or how typically iid the sequence is. The more typical the sequence, the more confident we are in $\hat{y}_{l+1}$.

To measure the typicalness of sequences, we define, for every $n = 1, 2, \ldots$, a function $t : \mathcal{Z}^n \to [0, 1]$ which has the property

$$P((z_1, \ldots, z_n) : t((z_1, \ldots, z_n)) \leq r) \leq r \tag{2}$$

If such a function returns 0.1 for a particular sequence, we know that the sequence is unusual because such values will be produced at most 10% of the time by any iid process. This means that we can exclude labels that we consider to be unlikely at some particular significance level, i.e. we can exclude all labels for the new example such that such a sequence would occur only 10% of the time or less.

It turns out that we can construct such functions by considering how 'strange' individual examples are. If we have some family of functions $f : \mathcal{Z}^n \times \{1, 2, \ldots, n\} \to \mathbb{R}$, $n = 1, 2, \ldots$, such that we can associate some strangeness value α with each example:

$$\alpha_i = f(\{z_1, \ldots, z_n\}, i) \qquad i = 1, \ldots, n \tag{3}$$

then we can define the following typicalness function

$$t((z_1, \ldots, z_n)) = \frac{\#\{\alpha_i : \alpha_i \geq \alpha_n\}}{n} \tag{4}$$

This can easily be proven to satisfy (2), provided that the strangeness function returns the same value for each example regardless of the order in which they are presented. Note that the inequalities in (2) and (3) are essential properies of a strangeness function.

In order to show this, we shall use an adapted version of a proof given in ([22, 18]). The typicalness test we use here is based on a test for randomness which was originally formulated by Martin-Löf. The cited references give more details on the relationship between typicalness and randomness tests.

First of all though, we need to define an individual strangeness measure. A family of functions $\{A_n : n \in \mathbb{N}\}$, where $A_n : \mathcal{Z}^n \to \mathbb{R}^n$ for all n, is called an *individual strangeness measure* if, for any n, any permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, any $(z_1, \ldots, z_n) \in \mathcal{Z}^n$ and any $(\alpha_1, \ldots, \alpha_n) \in \mathbb{R}^n$,

$$(\alpha_1, \ldots, \alpha_n) = A_n(z_1, \ldots, z_n) \Rightarrow (\alpha_{\pi(1)}, \ldots, \alpha_{\pi(n)}) = A_n(z_{\pi(1)}, \ldots, z_{\pi(n)}).$$

We tend to use positive individual strangeness measures, which take values in the range $[0, \infty)^*$. It is easy to see that function (3) can be used to construct an individual strangeness measure. Once we have an individual strangeness measure we can use the strangeness values produced in conjunction with equation (4).

Theorem 1 [22, 17] *Every function (4) obtained by a computable individual strangeness measure will satisfy equation (2).*

Proof: Let us fix some n. Without loss of generality we assume that r is of the form j/n, where $j \in \{1, \ldots, n\}$, and that P is the uniform distribution in $\Xi(z_0)$ (where $\Xi(z_0)$ is the set of all possible permutations of the sequence z_0) for some sequence $z_0 \in \mathcal{Z}^n$. If α is the sequence, $\{\alpha_1, \ldots, \alpha_n\}$, α_0 stands for $A_n(z_0)$ and $t(z)$ is defined as the right hand side of eq. 4, we have:

$$P\{z \in \mathcal{Z}^n : t(z) \leq j/n\} = \frac{\#\{z \in \Xi(z_0) : t(z) \leq j/n\}}{|\Xi(z_0)|}$$
$$= \frac{\#\{\alpha \in \Xi(\alpha_0) : \frac{\#\{i : \alpha_i \geq \alpha_n\}}{n} \leq j/n\}}{|\Xi(\alpha_0)|} = \frac{\#\{\alpha \in \Xi(\alpha_0) : \#\{i : \alpha_i \geq \alpha_n\} \leq j\}}{|\Xi(\alpha_0)|}$$
$$= \frac{\#\{\alpha \in \Xi(\alpha_0) : \alpha_n \in S_j(\alpha_0)\}}{|\Xi(\alpha_0)|} = \frac{|S_j(\alpha_0)|}{n} \leq j/n,$$

where $S_j(s_0)$ are the indices of the j smallest elements in the sequence s_0 (if there are any ties, then $S_j(s_0)$ is defined to be the largest set S of size at most j such that for some threshold h, S is the set of indices i for which $s_{0,i} \leq h$). □

3 Regression estimation

We will now consider some algorithms for regression estimation in the context of the Bayesian and typicalness frameworks before moving on to the pattern recognition setting.

Given a training sequence $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ we choose to model the dependency $y_i = f(\mathbf{x}_i)$ as $y_i = \mathbf{x}_i \cdot \mathbf{w}$ where $\mathbf{w} \in \mathbb{R}^d$. The well known ridge regression procedure [8] recommends choosing $\mathbf{w}$ to achieve

$$a||\mathbf{w}||^2 + \sum_{i=1}^{l} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 \to \min$$

where a is a positive constant (the ridge factor).

The method has a natural matrix representation: let Y be the vector of labels

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_l \end{pmatrix}$$

and X be the matrix formed from the training examples

$$X = \begin{pmatrix} \mathbf{x}_1' \\ \vdots \\ \mathbf{x}_l' \end{pmatrix}$$

We now wish to find $\mathbf{w}$ such that

$$a||\mathbf{w}||^2 + ||Y - X\mathbf{w}||^2 \to \min$$

or

$$Y'Y - 2\mathbf{w}'X'Y + \mathbf{w}'(X'X + aI)\mathbf{w} \to \min$$

Taking the derivative in $\mathbf{w}$ we obtain

$$2(X'X + aI)\mathbf{w} - 2X'Y = 0$$

or

$$\mathbf{w} = (X'X + aI)^{-1}X'Y \tag{5}$$

so our ridge regression estimate of the label for some new point $\mathbf{x}_{l+1}$ is

$$\tilde{y}_{l+1} = \mathbf{x}_{l+1}'(X'X + aI)^{-1}X'Y$$

4 Bayesian ridge regression

We now give a Bayesian derivation of the ridge regression estimator.

Suppose we have some data points $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$ and an unlabeled example $\mathbf{x}_{l+1}$. We assume that the unlabelled examples $x_1, \ldots, x_{l+1}$ are fixed (deterministic) and the labels $y_1, \ldots, y_l$ were generated by the rule

$$y_i = \mathbf{w} \cdot \mathbf{x}_i + \xi_i \tag{6}$$

where $\mathbf{w} \sim N(0, \frac{1}{a}I)$ and each $\xi_i \sim N(0, 1)$. We would like to predict y_{l+1} under these assumptions. We should therefore predict

$$y_{l+1} = \mathbf{x}_{l+1} \cdot \mathbf{w}_{\text{post}} + N(0, 1)$$

where $\mathbf{w}_{\mathrm{post}}$ is chosen according to the distribution $P(\mathbf{w}|(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l))$. Bayes rule gives us

$$P(\mathbf{w}|(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)) \propto P(\mathbf{w})P((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)|\mathbf{w}) \tag{7}$$

Recalling that the Normal distribution's density is

$$\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{(x-m)^2}{2\sigma^2}} \tag{8}$$

equations (6), (7) and the multivariate form of (8) give us

$$\begin{aligned} P(\mathbf{w}|(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)) &\propto e^{-\frac{1}{2}a||\mathbf{w}||^2} \prod_{i=1}^{l} e^{-\frac{1}{2}(y_i - \mathbf{x}_i \cdot \mathbf{w})^2} \\ &\propto e^{-\frac{1}{2}(a||\mathbf{w}||^2 + \sum_{i=1}^{l}(y_i - \mathbf{x}_i \cdot \mathbf{w})^2)} \end{aligned} \tag{9}$$

If we choose $\mathbf{w}$ to maximise (9), which is equivalent to choosing $\mathbf{w}$ such that

$$a||\mathbf{w}||^2 + \sum_{i=1}^{l}(y_i - \mathbf{x}_i \cdot \mathbf{w})^2 \to \min$$

we obtain exactly the same ridge regression estimator as in section 3.

Formula (9) can be written as

$$e^{-\frac{1}{2}(\mathbf{w}'(X'X+aI)\mathbf{w} - 2Y'X\mathbf{w} + Y'Y)} \tag{10}$$

The probability distribution function of the multivariate Normal distribution is

$$P(Z = \mathbf{z}) \propto e^{-\frac{1}{2}(\mathbf{z}-\mu)'\Lambda^{-1}(\mathbf{z}-\mu)} \tag{11}$$

where μ is the mean of the distribution and Λ is the covariance matrix. Substituting the right-hand side of (5) into (11) as μ (as the maximum of a Normal probability distribution function always occurs at the distribution's mean) and equating with (10) we obtain

$$\begin{aligned} & \mathbf{w}'(X'X + aI)\mathbf{w} - 2Y'X\mathbf{w} + Y'Y \\ = \; & (\mathbf{w} - (X'X + aI)^{-1}X'Y)'\Lambda^{-1}(\mathbf{w} - (X'X + aI)^{-1}X'Y) \\ \propto \; & \mathbf{w}'\Lambda^{-1}\mathbf{w} - \mathbf{w}'\Lambda^{-1}(X'X + aI)^{-1}X'Y - ((X'X + aI)^{-1}X'Y)'\Lambda^{-1}\mathbf{w} \end{aligned}$$

Setting $\Lambda = (X'X + aI)^{-1}$ gives

$$\begin{aligned} & \mathbf{w}'(X'X + aI)\mathbf{w} - 2Y'X\mathbf{w} + Y'Y \\ \propto \; & \mathbf{w}'(X'X + aI)\mathbf{w} - \mathbf{w}'X'Y - \mathbf{w}'((X'X + aI)^{-1}X'Y)(X'X + aI) \\ \propto \; & \mathbf{w}'(X'X + aI)\mathbf{w} - \mathbf{w}'X'Y - \mathbf{w}'(X'X + aI)'((X'X + aI)^{-1}X'Y) \end{aligned}$$

Using the above and noticing that $(X'X + aI)$ is symmetric we can write

$$\mathbf{w}'(X'X + aI)\mathbf{w} - 2Y'X\mathbf{w} + Y'Y \quad \propto \quad \mathbf{w}'(X'X + aI)\mathbf{w} - 2\mathbf{w}'X'Y$$

The weight vector's posterior distribution is therefore

$$P(\mathbf{w}|(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)) \sim N((X'X + aI)^{-1}X'Y, (X'X + aI)^{-1}) \tag{12}$$

and so the predictive distribution for a new example's label y_{l+1} is

$$y_{l+1} \sim N(\mathbf{x}'_{l+1}(X'X + aI)^{-1}X'Y, \mathbf{x}'_{l+1}(X'X + aI)^{-1}\mathbf{x}_{l+1} + 1)$$

Ridge Regression has a well known dual formulation [15], also known as Kriging, which allows the 'kernel trick' to be applied to find non-linear decision rules. In order to apply the kernel trick, we need to be able to find both μ and Λ for a new example using only dot products. The dual form of the mean is known [15] to be

$$Y'(K + aI)^{-1}\mathbf{k}$$

K is the kernel matrix defined by $K_{i,j} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$, $i, j = 1 \ldots l$ where $\mathcal{K}$ is some kernel function [20] and $\mathbf{k}$ is the vector defined by $\mathbf{k}_i = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_{l+1})$. To find such a representation for the covariance matrix, we need to introduce some new notation:

$$X_1 = (\mathbf{x}'_1),\ \ldots,\ X_l = \begin{pmatrix} \mathbf{x}'_1 \\ \vdots \\ \mathbf{x}'_l \end{pmatrix}, X_{l+1} = \begin{pmatrix} \mathbf{x}'_1 \\ \vdots \\ \mathbf{x}'_l \\ \mathbf{x}'_{l+1} \end{pmatrix}$$

Now, we can write our primal form for the variance as

$$\mathbf{x}'_{l+1}(X'_lX_l + aI)^{-1}\mathbf{x}_{l+1} + 1 \tag{13}$$

We can use the Sherman-Morrison formula [2]

$$(A + uv')^{-1} = A^{-1} + \frac{(A^{-1}u)(v'A^{-1})}{1 + v'A^{-1}u}$$

to express (13) in terms of X_{l+1}. First notice that

$$(X'_{l+1}X_{l+1} + aI)^{-1} \quad = \quad (X'_lX_l + aI + \mathbf{x}_{l+1}\mathbf{x}'_{l+1})^{-1} \tag{14}$$

$$A = (X'_lX_l + aI), u = \mathbf{x}_{l+1}, v = \mathbf{x}_{l+1}$$

For convenience we will also write

$$B = (X'_{l+1}X_{l+1} + aI)$$

$$B^{-1} = A^{-1} - \frac{(A^{-1} \cdot \mathbf{x}_{l+1}) \cdot (\mathbf{x}'_{l+1} \cdot A^{-1})}{1 + \mathbf{x}'_{l+1} \cdot A^{-1} \cdot \mathbf{x}_{l+1}}$$

Substituting into (13) yields

$$\mathbf{x}'_{l+1}B^{-1}\mathbf{x}_{l+1} \quad = \quad \mathbf{x}'_{l+1}A^{-1}\mathbf{x}_{l+1} - \frac{\mathbf{x}'_{l+1}(A^{-1} \cdot \mathbf{x}_{l+1}) \cdot (\mathbf{x}'_{l+1} \cdot A^{-1})\mathbf{x}_{l+1}}{1 + \mathbf{x}'_{l+1} \cdot A^{-1} \cdot \mathbf{x}_{l+1}}$$

$$\begin{aligned}
\mathbf{x}'_{l+1}B^{-1}\mathbf{x}_{l+1} &= \frac{\mathbf{x}'_{l+1}A^{-1}\mathbf{x}_{l+1}}{1+\mathbf{x}'_{l+1}A^{-1}\mathbf{x}_{l+1}} \\
\mathbf{x}'_{l+1}A^{-1}\mathbf{x}_{l+1} &= \mathbf{x}'_{l+1}B^{-1}\mathbf{x}_{l+1} + (\mathbf{x}'_{l+1}B^{-1}\mathbf{x}_{l+1})(\mathbf{x}'_{l+1}A^{-1}\mathbf{x}_{l+1}) \\
\mathbf{x}'_{l+1}A^{-1}\mathbf{x}_{l+1}(1-\mathbf{x}'_{l+1}B^{-1}\mathbf{x}_{l+1}) &= \mathbf{x}'_{l+1}B^{-1}\mathbf{x}_{l+1} \\
\mathbf{x}'_{l+1}A^{-1}\mathbf{x}_{l+1} &= \frac{\mathbf{x}'_{l+1}B^{-1}\mathbf{x}_{l+1}}{1-\mathbf{x}'_{l+1}B^{-1}\mathbf{x}_{l+1}} \\
\mathbf{x}'_{l+1}(X'_lX_l+aI)^{-1}\mathbf{x}_{l+1}+1 &= \frac{\mathbf{x}'_{l+1}(X'_{l+1}X_{l+1}+aI)^{-1}\mathbf{x}_{l+1}}{1-\mathbf{x}'_{l+1}(X'_{l+1}X_{l+1}+aI)^{-1}\mathbf{x}_{l+1}}+1 \qquad (15)
\end{aligned}$$

(15) allows us to apply the following identity:

$$\begin{aligned}
(XX'+aI)^{-1}X(X'X+aI) &= (XX'+aI)^{-1}(XX'+aI)X \\
(XX'+aI)^{-1}X(X'X+aI)(X'X+aI)^{-1} &= X(X'X+aI)^{-1} \\
X(X'X+aI) &= (XX'+aI)X \qquad (16)
\end{aligned}$$

Now

$$\begin{aligned}
\mathbf{x}'_{l+1}(X'_{l+1}X_{l+1}+aI)^{-1}\mathbf{x}_{l+1} &= \mathbf{b}'_{l+1}X_{l+1}(X'_{l+1}X_{l+1}+aI)^{-1}\mathbf{x}_{l+1} \\
&= \mathbf{b}'_{l+1}(X_{l+1}X'_{l+1}+aI)(X_{l+1}\mathbf{x}_{l+1}) \qquad (17)
\end{aligned}$$

where $\mathbf{b}_i$ is the $l+1$-vector $(0,\ldots,0,1)'$. Substituting the above into (15) gives the dual form of the covariance matrix:

$$\begin{aligned}
\Lambda &= \mathbf{x}'_{l+1}(X'_lX_l+aI)^{-1}\mathbf{x}_{l+1}+1 \\
&= \frac{\mathbf{b}'_{l+1}(X_{l+1}X'_{l+1}+aI)^{-1}(X_{l+1}\mathbf{x}_{l+1})}{1-\mathbf{b}'_{l+1}(X_{l+1}X'_{l+1}+aI)^{-1}(X_{l+1}\mathbf{x}_{l+1})}+1
\end{aligned}$$

The posterior for a new label's classification in the dual form is therefore

$$\hat{y}_{l+1} \sim N(Y'(K+aI)^{-1}\mathbf{k}, \frac{\mathbf{b}'_{l+1}(M+aI)^{-1}\mathbf{m}}{1-\mathbf{b}'_{l+1}(M+aI)^{-1}\mathbf{m}}+1)$$

where K, $\mathbf{k}$ and $\mathbf{b}$ remain defined as above and M and $\mathbf{m}$ are equivalent to K and $\mathbf{k}$ constructed using $\mathbf{x}_1$ to $\mathbf{x}_{l+1}$. A $t\%$ confidence tolerance region for a label will lie between the $\frac{1-t}{2}\%$ and $\frac{1+t}{2}\%$ quantiles of the label's posterior distribution.

5 Typicalness tests for regression estimation

Now we consider applying the typicalness framework to the particular case of regression estimation. In this case our sample space is $\mathcal{Z} = \mathbb{R}^d \times \mathbb{R}$. If we have some training sequence $((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l))$ and a new example $\mathbf{x}_{l+1}$ it is easy to find the typicalness of any postulated label $\hat{y}$. We use some regression algorithm whose predictions are independent of the order of training examples (e.g. ridge regression) to make predictions $\tilde{y}_1, \ldots, \tilde{y}_l, \tilde{y}_{l+1}$ on the basis of the training sequence extended with the new example and it's postulated label

$((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l), (\mathbf{x}_{l+1}, \hat{y}))$. We use the residuals to those predictions to find strangeness values:

$$\alpha_i = |y_i - \tilde{y}_i| \tag{18}$$

and then use equation (4) with $n = l$ to find the typicalness of $\hat{y}$.

In regression however, we are not interested in the typicalness of a single label, our confidence information should take the form of a set of possible labels admissible at some significance level. That is, we consider a set of labels $R \subseteq \mathcal{Y}$ such that

$$t((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l), (\mathbf{x}_{l+1}, \hat{y})) \leq r \qquad \forall \hat{y} \notin R, r \in [0, 1] \tag{19}$$

and return our confidence in the set as being at least $100(1-r)\%$. In statistics such a set is often called a tolerance region. Probably we will most often have some particular confidence level in mind (e.g. 95%) and will wish to predict some tolerance region in which we have at least that much confidence. Obviously, in order to find such a region we cannot consider all possible $\hat{y}$.

5.1 Ridge Regression Confidence Machine

The Ridge Regression Confidence Machine is an efficient algorithm for finding accurate tolerance regions with given significance levels for ridge regression estimates. It relies on the fact that the vector of residuals for a ridge regression estimate given an extended training sequence $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l), (\mathbf{x}_{l+1}, \hat{y})$ can be written as

$$\begin{aligned} \Delta &= |Y - X(X'X + aI)^{-1}X'Y| \\ &= |(I - X(X'X + aI)^{-1}X')Y| \end{aligned}$$

Noticing that $Y = (y_1, \ldots, y_l, 0) + (0, \ldots, \hat{y})$ we can see that Δ can be written as $|A + B\hat{y}|$ where

$$A = (I - X(X'X + aI)^{-1}X')(y_1, \ldots, y_l, 0)'$$

and

$$B = (I - X(X'X + aI)^{-1}X')(0, \ldots, 0, 1)'$$

Therefore each α_i varies piecewise-linearly as we change $\hat{y}$.

This, together with equation (4), means that the typicalness of the sequence can only change at points $\hat{y}$ such that $|a_i + b_i\hat{y}| = |a_{l+1} + b_{l+1}\hat{y}|$. In other words, we can partition the set of possible labels into a set of intervals $[\hat{y}_i, \hat{y}_{i+1})$, each having constant typicalness.

For each training example we can define

$$S_i = \{\hat{y} : \alpha_i \geq \alpha_{l+1}\} = \{\hat{y} : |a_i + b_i\hat{y}| \geq |a_{l+1} + b_{l+1}\hat{y}|\}$$

(the set of all possible labels for $\mathbf{x}_{l+1}$ such that z_i's residual is greater than or equal to z_{l+1}'s residual), where $i = 1, \ldots, l+1$. For any interval $[\hat{y}_i, \hat{y}_{i+1}]$ the typicalness is the number of sets S_i which cover the interval divided by $l + 1$.

To find S_i, first of all find for every training example the points where $|a_i + b_i\hat{y}| = |a_{l+1} + b_{l+1}\hat{y}|$. If $b_i \neq b_{l+1}$ then they are

$$\frac{a_i - a_{l+1}}{b_{l+1} - b_i} \quad \text{and} \quad -\frac{a_i + a_{l+1}}{b_i + b_{l+1}} \tag{20}$$

Let u_i be the minimum and v_i be the maximum of the two. If $b_i = b_{l+1} \neq 0$ then

$$u_i = v_i = -\frac{a_i + a_{l+1}}{2b_i} \tag{21}$$

unless $a_i = a_{l+1}$, in which case $S_i = \mathbb{R}$.

We can now define S_i:

$$S_i = \begin{cases} [u_i, v_i] & \text{if} \quad b_{l+1} > b_i \\ (-\infty, u_i\,] \cup [v_i, \infty) & \text{if} \quad b_{l+1} < b_i \\ [u_i, \infty) & \text{if} \quad b_{l+1} = b_i > 0 \\ & \text{and} \quad a_{l+1} < a_i \\ (-\infty, u_i] & \text{if} \quad b_{l+1} = b_i > 0 \\ & \text{and} \quad a_{l+1} > a_i \\ \mathbb{R} & \text{if} \quad b_{l+1} = b_i = 0 \\ & \text{and} \quad |a_{l+1}| \leq |a_i| \\ \emptyset & \text{if} \quad b_{l+1} = b_i = 0 \\ & \text{and} \quad |a_{l+1}| > |a_i| \end{cases} \tag{22}$$

If we take $-\infty, u_1, \ldots, u_n, v_1, \ldots, v_n, \infty$ and sort them in ascending order obtaining $\hat{y}_0, \ldots, \hat{y}_{2n+2}$ then for any interval $[\hat{y}_i, \hat{y}_{i+1}]$, the typicalness level will be the number of sets S_i which cover that interval divided by $l+1$:

$$t((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l), (\mathbf{x}_{l+1}, \hat{y})) = \frac{\#\{S_j : [\hat{y}_i, \hat{y}_{i+1}] \subseteq S_j\}}{l+1}, \ \forall \hat{y} \in [\hat{y}_i, \hat{y}_{i+1}]$$

Algorithm 1 describes how to implement the Ridge Regression Confidence Machine.

6 Pattern Recognition

In this section we briefly describe two algorithms which provide confidence values for pattern recognition tasks. One is Bayesian Transduction [5] which can be shown to approximate the Bayes optimal decision; the other uses the typicalness framework in conjunction with a kernel perceptron.

6.1 Bayesian Transduction

The Bayesian Transduction (BT) algorithm [5] is a transductive algorithm which uses Bayes Point Machines [7] as a basis. The idea behind the algorithm is as follows. Suppose we have a training sequence $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$. Assume that our hypothesis space $\mathcal{H}$ is the class of kernel perceptrons, where decision functions are given by

$$f_{\mathbf{w}}(\mathbf{x}) = \operatorname{sign}(\mathbf{w} \cdot \phi(\mathbf{x})) = \operatorname{sign}\left(\sum_{i=1}^{l} \alpha_i \mathcal{K}(\mathbf{x}_i, \mathbf{x})\right) \tag{23}$$

Algorithm 1 Ridge Regression Confidence Machine

Input training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$, new example $\mathbf{x}_{l+1}$ and significance level r
Calculate $C = I - X(X'X + aI)^{-1}X'$
Calculate $A = C(y_1, \ldots, y_l, 0)'$
Calculate $B = C(0, \ldots, 0, 1)'$
for $i = 1$ to $l + 1$ **do**
  calculate u_i and v_i as per (20) and (21)
end for
for $i = 1$ to $l + 1$ **do**
  find S_i as per (22)
end for
Sort $(-\infty, u_1, \ldots, u_n, v_1, \ldots, v_n, \infty)$ in ascending order obtaining $\hat{y}_0, \ldots, \hat{y}_{2n+2}$
Output $\cup_i [\hat{y}_i, \hat{y}_{i+1}]$, the union being over i such that $\frac{\#\{S_j : [\hat{y}_i, \hat{y}_{i+1}] \subseteq S_j\}}{l+1} > r$

where $\mathbf{w} = (\alpha_1, \ldots, \alpha_l)'$, $\phi(\mathbf{x}) = (\mathcal{K}(\mathbf{x}_1, \mathbf{x}), \ldots, \mathcal{K}(\mathbf{x}_l, \mathbf{x}))'$, and $\mathcal{K}$ is a kernel function. We define the so-called *version space* as the set of all $\mathbf{w}$ which are consistent with the training sample

$$V(S) = \{\mathbf{w} | f_{\mathbf{w}}(\mathbf{x}_i) = y_i; (\mathbf{x}_i, y_i) \in S; i = 1, \ldots, l\}^1 \tag{24}$$

Restricting $||\mathbf{w}|| = 1$ ensures uniqueness of index $\mathbf{w}$ for every $f = f_{\mathbf{w}} \in \mathcal{H}$. Note that the introduction of a test point $\mathbf{x}_{l+1}$ may bisect the volume V of version space into two sub volumes V^+ and V^-, where V^+ is the volume of version space in which any perceptron would classify the test point as +1, and V^- is the volume where perceptrons predict a negative classification. When assuming a uniform prior over $\mathbf{w}$ and the class of perceptrons it is clear that the ratio $p^+ = V^+/(V^+ + V^-)$ is the posterior probability of labelling the test point as +1. An ergodic billiard can be used to obtain estimates for the volumes of version space which produce posteriors p^+ (p^-) which do not deviate significantly from the true expectation of p^+ (p^-).

For this paper we followed the algorithm given in [5] and used $n = 1000$ bounces for the billard, which bounds the deviation from the true posterior at $e < 0.05$ with a probability of 99%. If the prior assumptions are satisfied, then we would expect the predictions and confidence values assigned by the algorithm to be approximately Bayes-optimal.

6.2 Perceptron with typicalness

The typicalness framework has recently been successfully applied to pattern-recognition Support Vector Machines [21, 17]. As a comparison with BT, we will use the typicalness framework in conjunction with a kernel perceptron.

[1] It is assumed that there is a function f^* in the space $\mathcal{H}$ such that for all $(\mathbf{x}, y) \in S$, $y = f^*(\mathbf{x})$.

In order to obtain our confidences and predictions for a test example $\mathbf{x}_{l+1}$, we do the following:

1. Add $(\mathbf{x}_{l+1}, 1)$ to our training sequence and run the kernel perceptron algorithm [7].
2. Use the resulting α_i values (from the expansion of $\mathbf{w}$) as the strangeness values, and use eq (4) to obtain t^+.
3. Repeat the above steps, but add $(\mathbf{x}_{l+1}, -1)$ to the training sequence instead, and use (4) to obtain t^-.

Once we have values for t^+ and t^- we use the following: for every confidence level $1 - r$, output as the prediction region

$$R = \begin{cases} \{-1, 1\} & \text{if} \quad t^+ > r \text{ and } t^- > r \\ \{1\} & \text{if} \quad t^+ > r \text{ and } t^- \leq r \\ \{-1\} & \text{if} \quad t^- > r \text{ and } t + - \leq r \\ \emptyset & \text{if} \quad t^+ \leq r \text{ and } t^- \leq r \end{cases}$$

7 Experimental comparisons

In order to compare the Bayesian and typicalness frameworks' performance, we generated artificial datasets from the prior distributions assumed by the Bayesian algorithms. We then generated further datasets, using a different (incorrect) prior, and compared the results. We also include some results on benchmark data sets which are taken from the UCI machine learning repository [1].

The general experimental set-up was as follows. For all experiments, 100 training and 100 test points were randomly selected a total of 10 times. Performance characteristics are then given which analyse both validity and the accuracy of the confidence values produced. For the benchmark data sets, the training and testing examples were randomly drawn from the set of all data points.

7.1 Regression estimation experiments

For the toy dataset we generated data points drawn from a uniform distribution over $[-10, 10]^5$ and for each of the 10 datasets drew a vector $\mathbf{w}$ from a five-dimensional normal distribution $N(0, I)$. That vector was then used to generate labels using the equation

$$y_i = \mathbf{w} \cdot \mathbf{x}_i + N(0, 1)$$

This data precisely meets the prior for Bayesian ridge regression with the ridge factor $a = 1$. MATLAB implementations of both algorithms take about 15 minutes to run all 10 splits on a 600Mhz DEC Alpha processor. However more efficient implementations might show a gap in performance between the algorithms.

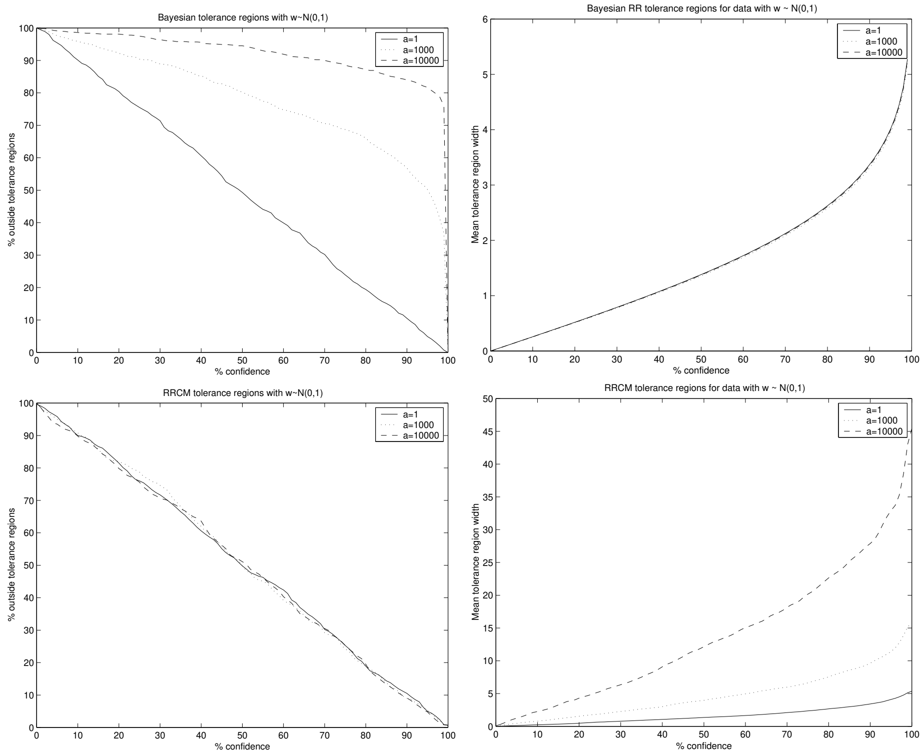

Figure 1 Bayesian RR and RRCM on data generated with $\mathbf{w} \sim N(0,1)$

We also experimented on two benchmark dataset, the auto-mpg dataset and the Boston housing dataset. For each experiment, we show the percentage confidence against the percentage of labels outside the tolerance region predicted for that confidence level. The percentage outside the tolerance region should never exceed 100 minus the percentage confidence, up to statistical fluctuations. If we have two valid algorithms, we need some qualitative measure with which to compare them. One natural comparison for regression estimation is the width of tolerance regions. We also therefore plot the percentage confidence against the mean width of the tolerance regions predicted for that confidence level. We say that algorithms giving narrower tolerance regions are more accurate.

Figure 1 shows results on the artificial data which was generated to meet Bayesian ridge regression's prior. The top left graph shows that when Bayesian RR is run with $a = 1$, fitting the prior, it generates valid tolerance regions. If we increase a however, a greater number of labels fall outside the tolerance regions. With a set to 10000, only about 15% of labels fall within a 90% tolerance region. Looking at the top right graph one can see that the tolerance region width is almost identical no matter how a is set. This causes more and more errors to be made as the regularisation is increased. If we instead look at the Ridge Regression Confidence Machine's performance (bottom graphs in 1) we can see that the tolerance regions contain contain almost precisely $r\%$ of the labels for every confidence level r, independent of the setting of a. The figure also shows that this is achieved through predicting wider tolerance intervals for

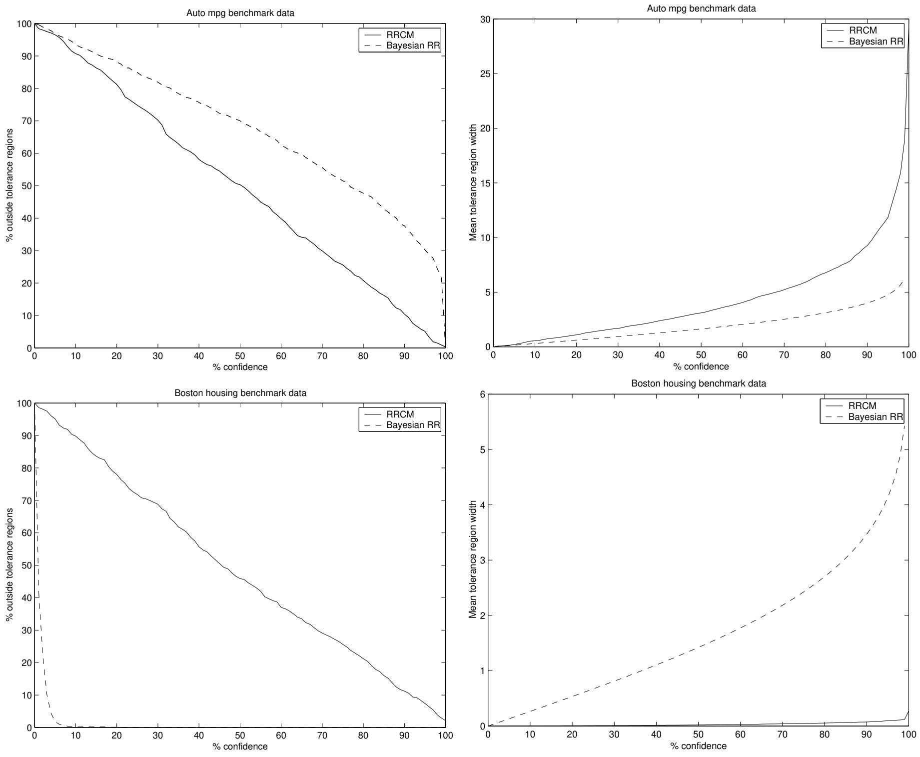

Figure 2 Bayesian RR and RRCM applied to Auto mpg and Boston housing benchmarks

higher settings of a.

These results show that the Bayesian algorithm only predicts tolerance regions valid in the sense of (1) when using the correct prior. As we change a we are effectively changing the prior, and the tolerance regions degrade in terms of validity. The typicalness algorithm however makes no assumptions about the value of the ridge parameter and so makes valid (and indeed 'nearly precise') predictions independent of its value.

Figure 2 shows results from applying the algorithms to two real world datasets, with the ridge factor a chosen such that a reasonable mean squared error is achieved. The top graphs in the figure show that Bayesian ridge regression is overconfident on the auto-mpg dataset, predicting tolerance regions that are too narrow. The Ridge Regression Confidence Machine predicts valid tolerance regions, the top right graph shows that to do so it gives wider tolerance regions than Bayesian ridge regression. On the Boston housing benchmark dataset, Bayesian ridge regression is too conservative. The bottom left graph shows that its predicted tolerance regions are always valid, however it also shows that they are much wider than those given by the Ridge Regression Confidence Machine. As the Ridge Regression Confidence Machine's tolerance regions are also valid, we prefer the more accurate RRCM's predictions.

7.2 Pattern Recognition Experiments

In this section we compare the Bayesian-Transduction (BT) algorithm and the kernel perceptron when used within the typicalness framework. We ran experiments on two toy datasets, and the well-known `heart` benchmark data set.

For the artificial data, one dataset was created using a uniform prior over $\mathbf{w}$ such that $||\mathbf{w}|| = 1$ (this is the correct prior for Bayesian Transduction). We generated 100 train and 100 test points uniformly in the range $[-10, 10]^5$ and then labelled all points with a $\mathbf{w}$ selected from the uniform prior, and repeated this process 10 times. For the second artificial data set we used a similar set-up to the one above, except that we added noise to our data from a normal distribution, and the experiments were run using an RBF kernel with $\sigma = 0.2$ (so the prior was no longer satisfied). To get an idea of the timings involved, all 10 splits took a total of 13 minutes when using Bayesian Transduction (implemented in C++) and the typicalness method took just over 7 minutes (in MATLAB). All experiments were run on a 233Mhz Dec Alpha. Please note however that both algorithms were implemented naively, and therefore possible gains in efficiency exits. In particular, a simple kernel adatron was used to initially enter the version space for the Bayesian Transduction algorithm, whereas an optimised SVM would be faster. For the typicalness framework techniques such as hashing ([17]) are known to improve performance and ability to scale well with large data sets.

In the case of pattern recognition our aim is once again to exclude labels which do not meet our confidence threshold. For a two class problem this means that we have four possible predictions; the label +1, the label -1, both labels {+1,-1} and the empty set $\{\emptyset\}$. Hopefully our algorithm will only give one prediction for an example, however the other two cases also provide us with important information. Predicting the empty set indicates that we cannot make a prediction at the required confidence level and therefore this can be used as a filter mechanism to perhaps indicate that a human should classify this example. Similarly predicting both labels indicates that it is not possible to reject a classification with that confidence level, and this could also be used as a filter. It is interesting to note that for confidence thresholds below 50%, the Bayesian method is always forced to make a non-empty prediction, whereas the typicalness method is not.

As in the regression case we plot two graphs; the first shows the percentage error achieved on the test set, and the second shows the number of "dual predictions" made for each confidence level. We want our algorithms to be valid in the sense of (1) so we would not expect any points to lie above the $y = x$ line in the first graph, again up to statistical fluctuations. Of course an algorithm can be trivially made valid by always prediction both labels (this corresponds to an infintely wide tolerance region in the regression case), however we wish are algorithms not only to be valid but also to be more accurate (have narrower tolerance regions). The second graph therefore shows the number of dual predictions made for each confidence level and the lower this value (for valid algorithms) the better. We are less interested in empty predictions, since there cannot be many of them (at most $100r\%$ for any confidence level $100(1-r)\%$,

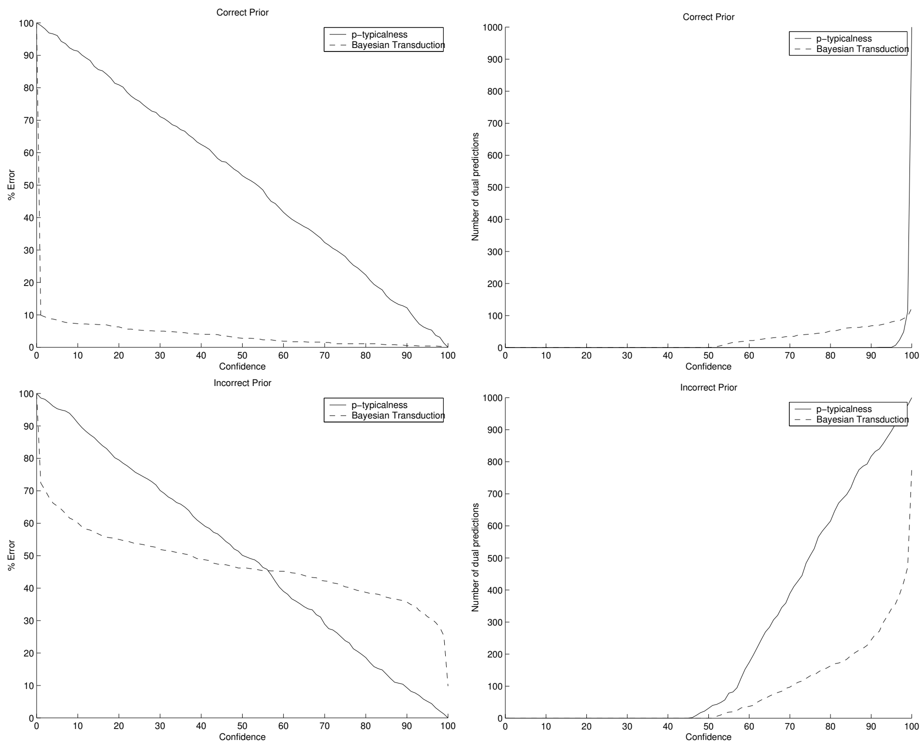

Figure 3 Bayesian Transduction (BT) and typicalness perceptron on data with a uniform prior (top), and a non-uniform prior (bottom).

up to statistical fluctuations).

The top graphs in Figure 3 show results for data generated by a correct prior. Both algorithms give valid results for this data set. The bottom two graphs show the setting when an incorrect prior is used. In this case Bayesian Transduction is over confident and produces too many errors for many confidence levels; whereas the values given by typicalness are valid.

The final graphs (figure 4) present results on the heart benchmark data set. Once again Bayesian transduction is over confident and produces a higher error rate than would be expected by the confidence rejection threshold. This is certainly not desirable in a real-world application, for a 95% threshold you would not expect the error rate to be much above 5% of errors. Notice how once again the typicalness method gives valid confidence values, with error rates at certain thresholds never in great excess of the values expected.

8 Conclusions

In this paper we have presented a comparison of the Bayesian framework and typicalness frameworks. We have highlighted the need for algorithms to produce valid (and ideally 'nearly precise') confidence values and outlined the typicalness framework which can be used in conjunction with many machine learning algorithms to achieve this goal. We have shown that the typicalness framework can

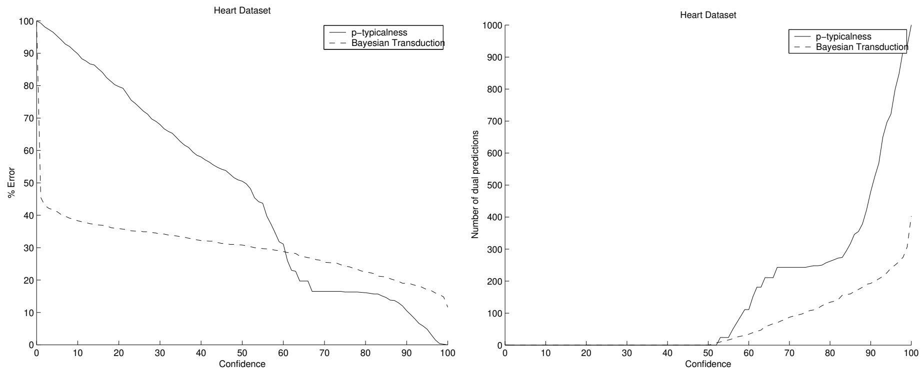

Figure 4 Bayesian transduction and typicalness perceptron on the heart benchmark

be easily applied to existing well-known algorithms for both regression and classification problems. In our experimental results we have shown that when the prior is correct Bayesian methods perform well (as expected), however the difference in performance to the methods using the typicalness framework (which does not rely on such priors) is negligible. When incorrect priors or real-world datasets are used, then Bayesian methods can be shown to produce "incorrect" confidence values for their predictions, whereas the typicalness methods still produce valid and nearly precise confidence values. Indeed, in almost all our experiments typicalness methods outperform their Bayesian counterparts in either validity or tightness of confidence intervals.

A Questions and Answers

In the process of explaining the typicalness framework and algorithms within it we have often found that friends and reviewers have some difficulties accepting the approach. Here we attempt to answer some common questions about the methods described above.

A.1 Doesn't PAC theory already give distrbution independent error bounds?

The short answer is yes, PAC theory provides distribution independent bounds which give the generalisation error for a given classifier for some confidence level δ. Unfortunately, for many algorithms – such as the Support Vector Machine – the bounds are currently not tight enough for practical use. A bound by Littlestone and Warmuth [9] has been suggested and is well known, however as shown in [14] the bound still gives probabilities of error greater than 1.

Recent interest in PAC-Bayesian bounds (see e.g. [6, 10]) for SVMs have resulted in some new, tighter bounds which are based on the normalised margin. These however are still not tight enough to be of any practical use. Also, many of these bounds conentrate on the overall generalisation error, and as yet do not give confidences for individual test examples.

A.2 Aren't you using unrealistic priors in your Bayesian algorithms?

This question along with close relatives such as "But with the correct prior the Bayes algorithm is optimal! Why use typicalness" are very common, and raise a good point. Bayesian algorithms provide optimal decisions *providing the prior is correct.* On many occasions, Bayesian methods are successful at capturing prior knowledge which is known, and often achieve very good generalisation error on tasks when the prior is either not-known, or a prior which is known to be incorrect is used. The drawback is however, that when confidence information is needed, the values given by Bayesian methods are only true if the correct prior is used. Bayesian model averaging and other techniques can give more robust results, but once again faith is being placed on the combination of priors used. When dealing with a large number of data sets, many common priors (Gaussian, Normal density) are not correct, and an attempt to provide confidence values from them will result in them being incorrect for future examples.

In this report we are trying to reinforce the view that making possibly incorrect assumptions at the begining can (and usually does) result in misleading confidence values. The typicalness approach however puts forward a different view, which makes no assumptions beyond the general iid assumption (in fact, we actually assume something weaker, namely exchangeability). This provides an alternative framework where nearly-precise confidences can be produced, irrespective of the distribution which generates the examples.

A.3 How can equation (2) hold for all iid data?

For a naive response, this question is easy; we could simple make the function $t(\mathbf{z})$ always return 1, and we are done (this actually corresponds to an individual strangeness measure assigning the same value to every $(\mathbf{x}, y)$ pair). This however is not that useful, so we want to find better functions which satisfy eq. (2). The real key to answering this question is that we are only dealing with a particular sequence of examples which are drawn iid from a fixed distribution. Because of this, all possible permutations of the sequence are equiprobable, and this is the fact which we use to obtain functions $t(\mathbf{z})$ which satisfy (2). One way to look at it is to go through the proof for typicalness tests in Section 2. From this you can see that it is only the exchangeability of the sequence which is important. It turns out that we are actually making a weaker assumption than iid, the requirement is that examples are drawn from an exchangeable distribution, and not necessarily iid. See [21] and [22] for a more detailed explanation, along with the connection between the iid and exchangeable families of distributions.

A.4 How does one choose a suitable strangeness function?

Good Question! This is currently an open topic of interest. Since the framework is very general, one can apply many different types of strangeness function to many different algorithms. Currently work has been done on applying strangeness functions to Support Vector Machines (based on lagrange and distances from the hyperplane), Ridge Regression (normal and dual form) and Nearest neighbour algorithms. Indeed, in this report we presented one test for typicalness, whereas there actually exist many. One other form of typicalness which the authors and colleagues are currently investigating is i-typicalness which is based on a test for randomness which was developed by Leonid Levin. See the papers cited in the "Further Information" section below for references.

B Further information

Gammerman et al [3] were the first to use a form of the typicalness framework to find confidence information for pattern recognition. Typicalness tests of the form used in this paper first appeared in Saunders et al [16], applied to pattern recognition support vector machines using transduction. Later, Saunders et al [17] described a more efficient form of the algorithm allowing the technique to be applied to large datasets. Surkov et al [19] have also investigated a more efficient inductive approach to the pattern recognition setting. Craig Saunders' thesis [18] describes several applications of the framework in the pattern recognition setting. Proedrou et al [14] have used the framework to add confidence information to k-nearest neighbours pattern recognition. A more detailed exposition of the theoretical background of the framework was published in a paper by Vovk et al [21]. Nouretdinov et al [12] report some further theoretical analysis of the framework, also examining the density estimation problem. A talk at the workshop on 'Using Unlabeled Data for Supervised Learning' at NIPS '99 described a method for finding tolerance intervals for regression estimates. More recently, Nouretdinov et al [13] describe the Ridge Regression Confidence Machine, an efficient algorithm for finding tolerance intervals for ridge regression estimates. Gilardi et al [4] have applied the framework to environmental risk mapping in Lake Geneva. This technical report is an extended version of a paper published at ECML'01 [11]. Finally, Vovk and Gammerman [22] are writing a book describing the framework and its roots in algorithmic randomness theory.

References

[1] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[2] B. Flannery, W. Press, S. Teukolsky, and Vettering W. *Numerical Recipies in C.* Cambridge University Press, 2nd edition edition, 1992.

[3] A. Gammerman, V. Vapnik, and V. Vovk. Learning by transduction. In *Uncertainty in Artificial Intelligence*, pages 148–155, 1998.

[4] Nicolas Gilardi, Michel Maignan, and Tom Melluish. Confidence Evaluation for Risk Prediction. In *Proceedings of IAMG2001, Cancun*, 2001.

[5] Thore Graepel, Ralf Herbrich, and Klaus Obermayer. Bayesian Transduction. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller, editors, *Advances in Neural Information Processing Systems 12: Proceedings of the 1999 Conference*, pages 456–462. The MIT Press, 1999.

[6] R. Herbrich and T. Graepel. A PAC-Bayesian Margin Bound for Linear Classifiers: Why SVMs work. In *Proceedings of NIPS*, 2001.

[7] Ralf Herbrich, Thore Graepel, and Colin Campbell. Bayesian Learning in Reproducing Kernel Hilbert Spaces. Technical report, Technical University of Berlin, Franklinstr. 28/29, 10587 Berlin, 1999.

[8] A. Hoerl and R.W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[9] N LittleStone and M. Warmuth. Relating data compression and learnability. Technical report, University of Califonia, Santa Cruz, 1986.

[10] D. A. McAllester. Some PAC Bayesian theorems. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 230–234, 1998.

[11] Thomas Melluish, Craig Saunders, Ilia Nouretdinov, and Volodya Vovk. Comparing the Bayesian and typicalness frameworks. In *Proceedings of ECML'01*, Lecture Notes in Computer Science. Springer-Verlag, 2001.

[12] I. Nouretdinov, V. Vovk, M. Vyugin, and A. Gammerman. Pattern recognition and density estimation under the general iid assumption. In *Accepted for COLT2001/EUROCOLT2001*, 2001.

[13] Ilia Nouretdinov, Tom Melluish, and Volodya Vovk. Ridge Regression Confidence Machine. In *Accepted for ICML*, 2001.

[14] Kostas Proedrou, Ilia Nouretdinov, Volodya Vovk, and Alex Gammerman. Transductive Confidence Machines for Pattern Recognition. Technical Report CLRC-TR-01-02, Royal Holloway University of London, June 2001.

[15] C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, 1998.

[16] C. Saunders, A. Gammerman, and V. Vovk. Transduction with Confidence and Credibility. In *Proceedings of IJCAI '99*, volume 2, 1999.

[17] C. Saunders, A. Gammerman, and V. Vovk. Computationally Efficient Transductive Machines. In *Proceedings of the Eleventh International Conference on Algorithmic Learning Theory 2000 (ALT '00)*, Lecture Notes in Artificial Intelligenece. Springer-Verlag, 2000.

[18] Craig Saunders. *Efficient Implementation and Experimental Testing of Transductive Algorithms for Predicting with Confidence*. PhD thesis, Royal Holloway and Bedford New College, University of London, 2000.

[19] David Surkov, Vladimir Vovk, and Alex Gammerman. Inductive Confidence Machine for pattern recognition. In *Submitted to NIPS'01*, 2001.

[20] Vladimir Vapnik. *The Nature of Statistical Learning Theory.* Springer, 1995.

[21] V. Vovk, A. Gammerman, and C. Saunders. Machine-learning applications of algorithmic randomness. In *Machine Learning, Proceedings of the Sixteenth International Conference (ICML'99)*, 1999.

[22] Volodya Vovk and Alex Gammerman. *Algorithmic theory of randomness and its applications in computer learning.* Unpublished manuscript, 2000.