Comparing the Bayes and typicalness frameworks

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

Computer Science, Royal Holloway, University of London, Egham, Surrey, TW20 0EX

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[1] 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 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% predictive intervals can contain

[1] Published in Proceedings of ECML 01, Lecture Notes in Computer Science, Springer-Verlag, 2001

the true label with in much less than 95% of the time. 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 is 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 sketch algorithms for regression estimation and pattern recognition in both the Bayesian and typicalness frameworks. A more comprehensive treatment is given in [4]. In Section 7 we present some experimental results and our conclusions are in Section 8.

2 The typicalness framework

Here we give a brief outline of the typicalness framework. For more details, see [6, ?,11]. 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 $\tilde{y}_{l+1}$ and for each one we examine how likely it is that all elements of the extended sequence $((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l), (\mathbf{x}_{l+1}, \tilde{y}_{l+1}))$ 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 $\tilde{y}_{l+1}$.

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 or less 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, e.g. we can exclude all labels for the new example which would occur only 10% of the time or less.

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

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

with each example and define the following typicalness function

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

which can 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 [8, 11].

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 [3] 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 $Y = (y_1, \ldots, y_l)'$ and X be the matrix formed from the training examples $X = (\mathbf{x}_1, \ldots, \mathbf{x}_l)'$. We now wish to find $\mathbf{w}$ such that

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

Taking derivatives in $\mathbf{w}$ and rearranging we find this is achieved when

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

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 $\mathbf{w}_{\text{post}}$ 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

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

where μ is given by the right-hand side of (5), and the probability distribution of the multivariate Normal distribution is

$$P(\mathbf{x}) \propto e^{-\frac{1}{2}(\mathbf{x}-\mu)'\Lambda^{-1}(\mathbf{x}-\mu)}$$

where μ is the mean of the distribution and Λ is the covariance matrix. Therefore the weight vector's posterior distribution is

$$P(\mathbf{w}|(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)) \sim N((X'X + aI)^{-1}X'Y, (XX' + aI)^{-1}) \tag{11}$$

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

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

Ridge Regression has a well known dual formulation [5], also known as Kriging, which allows the 'kernel trick' to be applied to find non-linear decision rules. The posterior for a new label's classification in the dual form is

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

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 [9], $\mathbf{k}$ is the vector defined by $\mathbf{k}_i = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_{l+1})$ and $\mathbf{b}$ is the $l+1$-vector $(0, \ldots, 0, 1)'$. 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 $\hat{y}_{l+1}$. 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 its postulated label $((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l), (\mathbf{x}_{l+1}, \hat{y}_{l+1}))$. We use the residuals to those predictions to find strangeness values:

$$\alpha_i = |y_i - \tilde{y}_i| \tag{12}$$

and then use equation (4) (with $n = l+1$) to find the typicalness of $\hat{y}_{l+1}$.

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 confidence level. That is, we consider a set of labels $R \subseteq \mathcal{Y}$ such that

$$t((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l), (\mathbf{x}_{l+1}, \hat{y})) \leq r \;\; \forall \hat{y} \notin R, r \in [0,1] \tag{13}$$

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

There exists an efficient application of the typicalness framework to ridge regression, the Ridge Regression Confidence Machine (RRCM), which allows tolerance regions to be found for any particular confidence level without considering all possible labelings of the new example. For details of the algorithm see [**?**].

The algorithm works by partitioning the real line into a set of intervals, each of which has uniform typicalness. This avoids the problem of having to explicitly consider all possible values for $\hat{y}$. For any particular confidence level $r\%$, the algorithm returns a union of these intervals that have typicalness $> 1 - \frac{r}{100}$.

6 Pattern Recognition

In this section we briefly describe two algorithms which provide confidence values for pattern recognition tasks. One is Bayesian Transduction [1] which can be shown to approximate the Bayes optimal decision; the other uses the typicalness framework in conjunction with a kernel perceptron (which is equivalent to a Support Vector Machine with no threshold).

6.1 Bayesian Transduction

The Bayesian Transduction (BT) algorithm [1] is a transductive algorithm which uses Bayes Point Machines [2] 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

$$f_{\mathbf{w}}(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \phi(\mathbf{x})) = \text{sign}\left(\sum_{i=1}^{l} \alpha_i \mathcal{K}(\mathbf{x}_i, \mathbf{x})\right) \tag{14}$$

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

$$V(S) = \{\mathbf{w} | f_{\mathbf{w}}(\mathbf{x}_i) = y_i; (\mathbf{x}_i, y_i) \in S; i = 1, \ldots, l\}^2 \tag{15}$$

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 [1] 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.

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

6.2 Perceptron with typicalness

The typicalness framework has recently been successfully applied to pattern-recognition Support Vector Machines [10, 7]. As a comparison with BT, we will use the typicalness framework in conjunction with a kernel perceptron.

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 [2].
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

- $\{-1, 1\}$ if $t^+ > r$ and $t^- > r$
- $\{1\}$ if $t^+ > r$ and $t^- \leq r$ (and vice versa)
- $\emptyset$ if $t^+ \leq r$ and $t^- \leq r$

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 a similar data set, 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.

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

This data precisely meets the prior for Bayesian ridge regression with the ridge factor $a = 1$. We also experimented on two benchmark datasets, 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

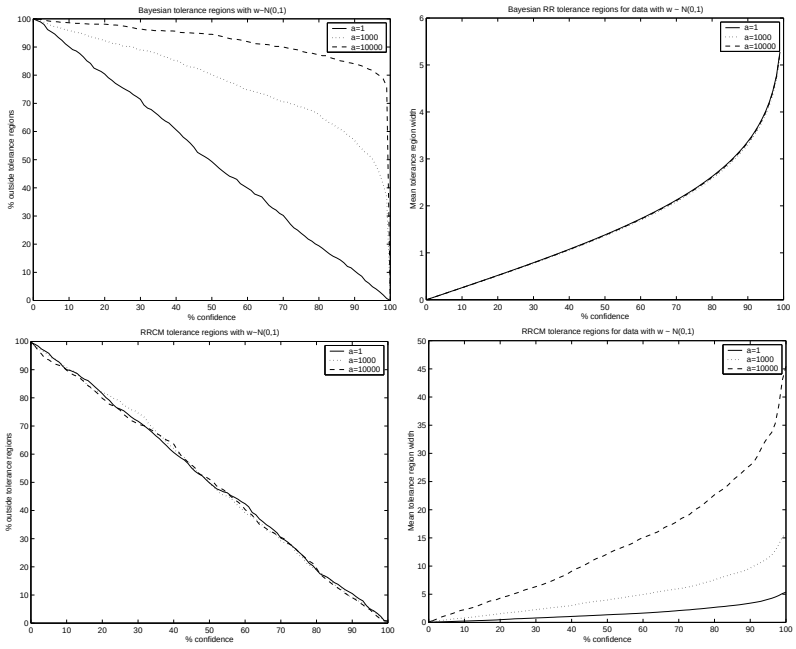

Fig. 1. Bayesian RR and RRCM on data generated with $\mathbf{w} \sim N(0, 1)$

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 RRCM's performance (bottom graphs in Figure 1) we can see that the tolerance regions contain almost precisely r% of the labels for every confidence level r, independent of the setting 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.

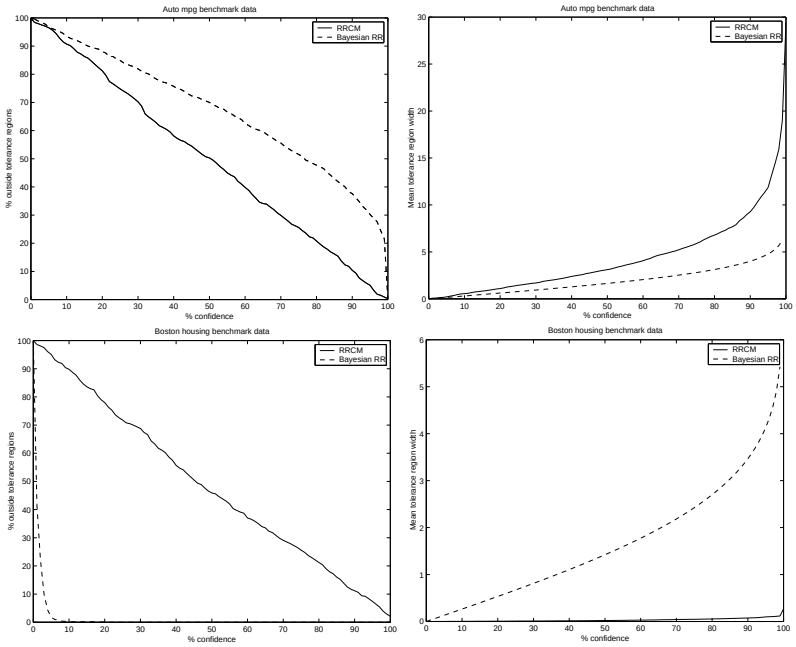

Fig. 2. Bayesian RR and RRCM applied to Auto mpg and Boston housing benchmarks

Figure 2 shows results from applying the algorithms to two real world datasets, with the ridge coefficient a chosen so that a reasonable mean square error is obtained. 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 RRCM 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 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 RRCM. As the RRCM'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 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 BT). We generated 100 train and 100 test points uniformly in the range $[-10, 10]^5$ and then labeled 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 BT (implemented in C++) and the typicalness method took just over 7 minutes (in MATLAB). All experiments were run on a 233Mhz Dec Alpha. Both algorithms however have a naive implementation, and improvements in performance could be made.

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 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 predicting both labels (this corresponds to an infinitely wide tolerance region in the regression case), however we wish our 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 the better. We are less interested in empty predictions, since there cannot be many of them (at most $100r\%$ at any confidence level $100(1-r)\%$, 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 BT is over confident and produces too many errors for many confidence levels; whereas the values given by typicalness are valid.

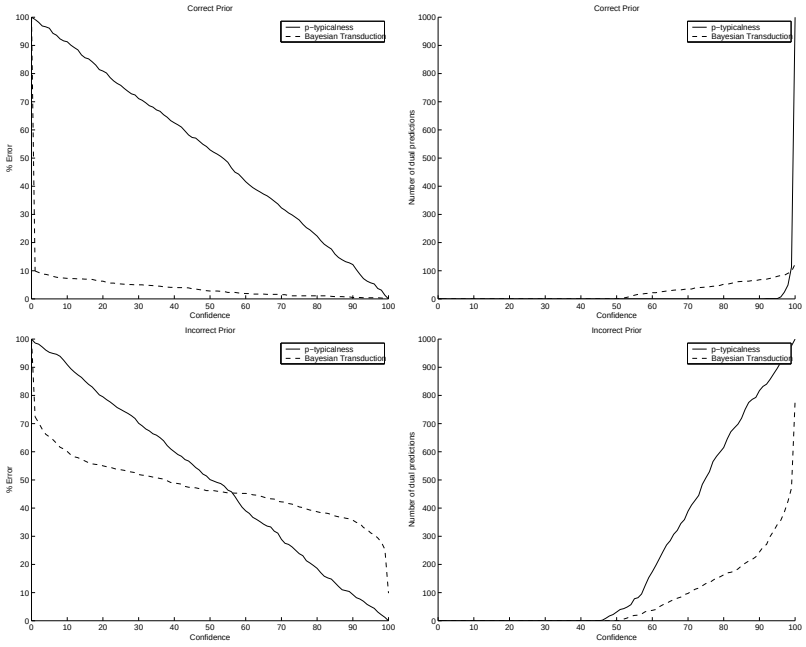

Fig. 3. Bayesian Transduction (BT) and typicalness perceptron on data with a uniform prior (top), and a non-uniform prior (bottom).

Figure 4 presents results on the heart data set. Once again BT 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 presented a comparison of the Bayesian and typicalness frameworks. We 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 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

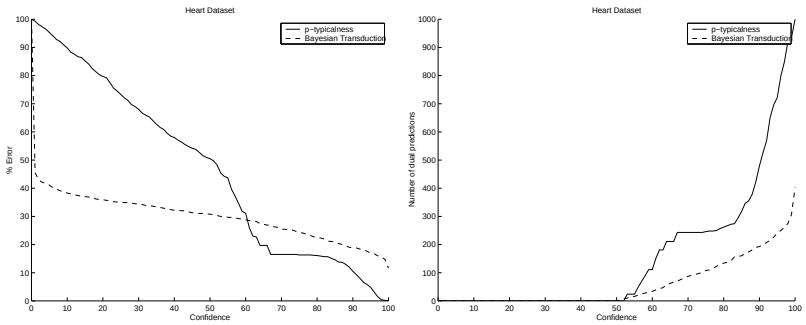

Fig. 4. Bayesian transduction and typicalness perceptron on the heart benchmark

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 levels. Indeed, in almost all our experiments typicalness methods outperform their Bayesian counterparts in either validity or tightness of confidence intervals.

References

1. 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.
2. 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.
3. A. Hoerl and R.W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.
4. T. Melluish, C. Saunders, I. Nouretdinov, and V Vovk. The typicalness framework: a comparison with the Bayesian approach. Technical Report CLRC-TR-01-05, Royal Holloway University of London, 2001.
5. 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.
6. C. Saunders, A. Gammerman, and V. Vovk. Transduction with Confidence and Credibility. In *Proceedings of IJCAI '99*, volume 2, 1999.
7. 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.
8. 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.
9. Vladimir Vapnik. *The Nature of Statistical Learning Theory.* Springer, 1995.

10. 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.
11. Volodya Vovk and Alex Gammerman. *Algorithmic theory of randomness and its applications in computer learning*. Unpublished manuscript, 2000.