Query Learning with Gaussian Processes

Simon Rogers
Department of Engineering Mathematics
University of Bristol
Bristol BS8 1TR, United Kingdom
Simon.Rogers@bristol.ac.uk

Colin Campbell
Department of Engineering Mathematics
University of Bristol
Bristol BS8 1TR, United Kingdom
C.Campbell@bristol.ac.uk

June 18, 2003

Abstract

The active selection of examples can significantly improve the efficiency of the learning process. For many tasks, compared to passive selection, a comparable test error can be attained using fewer examples which are correspondingly more informative. For classification a number of active learning algorithms have been proposed. However, there has been little progress in the development of corresponding active learning procedures for regression. Gaussian Processes have an explicit probabilistic formulation enabling the determination of posterior class probabilities for classification and Bayesian confidence intervals for regression. Using this probabilistic framework we outline an active learning algorithmic framework for regression. We demonstrate the advantages of using active learning with several classification and regression datasets.

1 Introduction

The ability to pose queries and to actively interogate the environment is the hallmark of an intelligent system. It can improve the efficiency of the learning process since, by posing maximally informative queries, a task could be learnt just as efficiently but using fewer examples compared to passive learning. For this reason query learning [2] has been well studied within machine learning. Two approaches are possible: firstly, the learning machine may be able to create a query and ask for the label. Secondly, the learning machine may be able to select an unlabelled datapoint and ask for the label. The creation of queries only makes sense in certain contexts. For example, for a dataset consisting of handwritten characters, it is possible to ask for the label of an unlabelled

character. However, if we create characters it would be easy to create instances which have no meaningful label. In this paper we will only consider the latter task if which the learning machine has the ability to ask the label of selected datapoints, a problem is known as *instance selection* or *selective sampling*.

Instance selection has been considered by learning theorists from several viewpoints including PAC learning and statistical mechanics (SM) calculations of the generalization error curves [16]. In an early paper Rivest and Eisenberg [12] showed that it is possible to create malicious distributions in which there is no advantage for active learning over passive learning. On the other hand, for more typical distributions, SM calculations [16, 17] indicated that there is a clear advantage to active learning. For perceptrons these SM calculations also showed that the best unlabelled point to query is that point closest to the current hyperplane (this hyperplane being derived from those points already queried and labelled). This idea was successfully transferred [4, 14, 13] to active learning using Support Vector Machines (SVMs) and it was demonstrated to be effective for both artificial and real life examples. For example, it has been successfully used to substantially improve detection of biologically active samples in drug discovery [15]. Since we never need to ask for the labels of the eventual non-support vectors, instance selection makes sense in this context and an active learning strategy can be viewed as a heuristic for finding support vectors. The theorem of Rivest and Eisenberg applies since it is possible to invent problems in which every point will be a support vector: but this is not typical of most problems encountered. Campbell et al. [4] argued that there is always a gain to be made by querying a point in the margin band between the support vector hyperplanes. Finally, since the sparsity ratio of SVMs (the ratio of number of support vectors to dataset size) typically decreases as a dataset size increases, active learning is expected to be most efficient for large datasets.

A different viewpoint on active learning is provided by considering version space. Maximally informative examples (maximising the entropy) are those examples which bisect version space into equal halves. These maximally informative examples can be found using a geometric billiard as in the Bayes Point Machine [8]. However, a much simpler approach is to populate version space at random locations with a set of 'voters' using a learning rule without a convex cost function (e.g. perceptrons). This committee approach to query learning [7] can be used to select the most informative examples efficiently - the points with maximally ambiguous predicted labels according to the 'voters'.

These approaches have worked well for active selection using classification. However, margin or version space concepts do not transfer readily to the problem of active learning for regression. In this paper we will consider active selection using Gaussian Processes (GPs). Since GPs give confidence measures for regression and posterior class probabilities for classification we can readily build an active learning strategy. This gives a further route to active learning for classification but also a means for tackling regession tasks. There are a number of possible algorithmic approaches to active learning for regression and we will present a more detailed study elsewhere. In this paper we will extend a sparse online learning algorithm for GPs recently proposed by Csato and Opper [6]. Since this algorithm is fully described elsewhere [6] we need only summarise the approach and list pseudo-code for the active learning variants in section 2. In section 3 we demonstrate the advantages of active learning with artificial

and real-life examples.

2 The Algorithms

2.1 Gaussian Process Models

Gaussian Processes (GP) can be viewed as Bayesian kernel machines [10, 11]. Let us consider a training set consisting of input vectors $\mathbf{X}_N \equiv \{\mathbf{x}^{(n)}\}_{n=1}^N$ with corresponding labels $\mathbf{t}_N$, produced by some underlying function $y(\mathbf{x})$ which is parameterized by a set of parameters $\mathbf{w}$. We can use a Bayesian approach to infer the function given the data. This inference is described by the posterior probabiliy distribution:

$$P(y(\mathbf{x}|\mathbf{t}_N, \mathbf{X}_N)) = \frac{P(\mathbf{t}_N|y(\mathbf{x}), \mathbf{X}_N)P(y(\mathbf{x}))}{P(\mathbf{t}_N|\mathbf{X}_N)} \tag{1}$$

Now, $P(y(\mathbf{x}))$ is the prior probability distribution on the family of functions assumed by the parameterization of $y(\mathbf{x})$. For example, for a feed-forward neural network, this would be a prior over the weights which implies a prior distribution over the overall function. In GPs, no such parameterization takes place and the prior $P(y(\mathbf{x}))$ is placed over the space of *all* functions. The simplest type of prior over such a function space is called a Gaussian Process which can be viewed as a generalization of a Gaussian Distribution over a finite space to an infinite dimensional function space. The GP is defined by its mean and a covariance function $C(\mathbf{x}_1, \mathbf{x}_2)$ which gives the expected covariance of $y(\mathbf{x})$ at points $\mathbf{x}_1$ and $\mathbf{x}_2$.

2.2 An Active Learning Algorithm for Regression and Classification

One approach to active learning with GPs is to adapt an online algorithm for training GPs with a suitable criterion for selecting the most informative examples. Several approaches are possible, though here we will only adapt one sparse online algorithm and refer the reader to [6] for further details.

2.2.1 Regression

In the pseudo-code listing of the algorithm (below) we give an outline of the sparse online GP algorithm of Csató and Opper ([6, 5]) for regression using an RBF kernel $K_0(\mathbf{x}_1, \mathbf{x}_2) = \exp(-||\mathbf{x}_1 - \mathbf{x}_2||^2/2d\sigma_k^2)$ where d is the dimensionality of the input data and σ_k^2 is the user-specified kernel parameter. RBF kernels were chosen as they ensure that input points only have influence on their locality in the input space. This is beneficial for active selection as it will lead to a larger degree of uncertainty in unexplored areas of the input space. For passive learning the examples are presented in a random order. For active selection the examples are presented in such an order as to enhance the learning process. For the case of regression we propose the following method. The current estimates of the mean and covariance of the gaussian process are

used to predict the output distribution for each of the remaining training points in $\mathbf{X}$, given by:

$$p(y|\mathbf{x}, \boldsymbol{\alpha}, \mathbf{C}) = \left(\frac{1}{2\pi\sigma_x^2}\right)^2 \exp\left\{-\frac{||y - \boldsymbol{\alpha}^T\mathbf{k}_x||^2}{2\sigma_x^2}\right\} \tag{2}$$

with $\sigma_x^2 = \sigma_0^2 + \mathbf{k}_x^T\mathbf{C}\mathbf{k}_x + k_x^*$ and $k_x^* = K_0(\mathbf{x}, \mathbf{x})$, and the output distribution for each point is assumed to be $y \sim \mathcal{N}(\boldsymbol{\alpha}^T\mathbf{k}_x, \sigma_x^2)$. Thus our selection procedure is to pick the point from the training set with the maximum variance σ_x as this is expected to be most uncertain and hence the point that would provide us with the maximum information. For a detailed description of the algorithm see [6] or [5]. In brief our main parametric quantities, $\boldsymbol{\alpha}_t$ and $\mathbf{C}_t$ (used to calculate our approximation of the mean and covariance of the GP at time t respectively) begin as null matrices. $\mathbf{Q}$ stores a representation of the inverse of $\mathbf{C}$ which can be updated iteratively and thus avoids inverting $\mathbf{C}$ at each step. $\mathbf{B}$ will hold basis vectors (training vectors that are used in out current approximation). Initially, this is empty and the number of basis vectors N_b is zero. Having selected the next point (as described above) we calculate $\mathbf{k}_{x_p}$ - this holds the prior covariance between the new point and each of the current basis points. This is then used to calculate σ_x^2, $q^{(t+1)}$, $r^{(t+1)}$ and γ_{t+1}. γ_{t+1} can be thought of as the *novelty* of the current input. If this does not exceed some pre-determined threshold ε_{tol} then the point is not included as a basis vector and we perform a reduced update on our GP parameters - i.e., their size is not increased. If however, the *novelty* does exceed our threshold, we perform a full update of our GP parameters and add this current point to $\mathbf{B}$. Note, the function $T_{t+1}(\mathbf{a})$ increases the dimension of the vector $\mathbf{a}$ by adding an empty element. Similarly, $U_{t+1}(\mathbf{Q})$ increases the dimension of a square matrix by adding an empty row and column. To give a sparse solution we can impose a maximum value on the number of basis vectors in $\mathbf{B}$. Every time a vector is added to $\mathbf{B}$, we would check to see if we have not exceeded this number. If so, the lowest scoring basis vector is removed from the basis and the GP parameters are updated. The definitions of $\mathbf{C}^{(t)}$, $\mathbf{C}^*$ and c^* used in the update are shown in figure 1.

2.2.2 Classification

For classification, the algorithm listed in the pseudo-code is used with the equations for $q^{(t+1)}$ and $r^{(t+1)}$ given by (3) to (6) below, where erf' and erf'' are the first and second derivatives of the error function defined in (6).

$$q^{(t+1)} = \frac{y_{t+1}}{\sigma_x}\frac{erf'(z)}{erf(z)} \tag{3}$$

$$r^{(t+1)} = \frac{1}{\sigma_x^2}\left\{\frac{erf''(z)}{erf(z)} - \left(\frac{erf'(z)}{erf(z)}\right)^2\right\} \tag{4}$$

$$z = \frac{y\boldsymbol{\alpha}_t^T\mathbf{k}_{x_p}}{\sigma_x} \tag{5}$$

$$erf(z) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{z} \exp\left(\frac{-t^2}{2}\right) dt \tag{6}$$

For the classification case, the next point is chosen as that example which is maximally informative. The class-conditional probability for each of the remaining training points is calculated from:

$$p(y|\mathbf{x}, \boldsymbol{\alpha}, \mathbf{C}) = erf\left(\frac{y\boldsymbol{\alpha}^T\mathbf{k}_x}{\sigma_x}\right) \tag{7}$$

where y can be either ± 1 for the binary case. This probability is then used to calculate the cross-entropy for that point, given by:

$$e = -p_{-1}\log_2(p_{-1}) - (1-p_{-1})\log_2(1-p_{-1}) \tag{8}$$

The chosen point is that which has the highest value of e, though this amounts to choosing the point with posterior probability closest to 0.5 in (7) during the active learning process.

3 Experiments

3.1 Classification

As our first example for classification we will consider the majority rule since this dataset has a low sparsity ratio and hence has been shown to perform well with a SVM query learning algorithm ([4]). The training set is made up of vectors consisting of ± 1. The target class is $+1$ if there are more $+1$ than -1 in the input vector and vice versa. The data set comprised 400 points, that was partitioned 100 times into training and test sets of 200 points each. In each experiment, the upper curve shows the performance obtained by choosing the next point at random and the lower curve is for active learning. The entropy plot (figure 2(d)) indicates the gain in information return per example using active learning. Parameters values used were $\sigma_k^2 = 400$, $\sigma_0^2 = 0.0$, $\varepsilon_{tol} = 1e-6$ and $M_{BV} = 50$. In figure 2(c) we give results for classification using the Cleveland Heart dataset [1]: here the task is to detect the presence or absence of heart disease. The learning curve in figure 2(c) was generated using 100 partitions of the 297 instances into 150 training points and 147 test points. Parameter values were $\sigma_k^2 = 0.06$, $\sigma_0^2 = 0.0$, $\varepsilon_{tol} = 1e-3$ and $M_{BV} = 150$. The final classification study is from the UCI repository [3] and the task is to distinguish between two different types of wine. The learning curve in 2(d) is created from 100 partitions of the 130 instances into 80 train and 50 test points. Parameter values were $\sigma_k^2 = 10.0$, $\sigma_0^2 = 0.0$, $\varepsilon_{tol} = 1e-6$ and $M_{BV} = 80$. In each learning curve plot, the lower curve corresponds to active selection and the y axis shows the number of mis-classified points. For the heart and wine datasets we notice that prototypical examples are learnt first resulting in a sharp fall in the learning curve. Thereafter, the learning machine learns outliers and the test error can climb. This phenomenon is not apparent with passive learning.

3.2 Stopping Criteria

To complete the algorithm we also need a stopping criterion. For SVM classifiers, Campbell et al [4] proposed stopping criteria based on successive agreement between predicted label and the actual label (noiseless datasets) or the absense of points in the

Algorithm 1 Sparse Online Gaussian Process

Require: Training Data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, Targets $\mathbf{y}$, kernel parameter σ_k^2, noise parameter σ_0^2, Maximum number of Basis Vectors M_{BV}, Tolerance ε_{tol}, Basis vector matrix $\mathbf{B} \leftarrow [\,]$,$\alpha \leftarrow [\,]$, $\mathbf{C} \leftarrow [\,]$, $\mathbf{Q} \leftarrow [\,]$, $N_t =$ Number of points in training set, $N_b = 0$ Number of basis vectors.

for $t = 0$ to $N_t - 1$ **do**
  Choose Next Point ($\mathbf{x}_p$) *Randomly or Actively*
  $\mathbf{k}_{x_p} \leftarrow [K_0(\mathbf{b}_1, \mathbf{x}_p), \ldots, K_0(\mathbf{b}_{N_b}, \mathbf{x}_p)]^T$
  $\sigma_x^2 \leftarrow \sigma_0^2 + \mathbf{k}_{x_p}\mathbf{C}_t\mathbf{k}_{x_p}^T + k_{x_p}^*$
  $q^{(t+1)} \leftarrow (y_{t+1} - \boldsymbol{\alpha}_t^T\mathbf{k}_{x_p})/\sigma_x^2$
  $r^{(t+1)} \leftarrow -1/\sigma_x^2$
  $\gamma_{t+1} \leftarrow k_{x_p}^* - \mathbf{k}_{x_p}^T\mathbf{Q}_t\mathbf{k}_{x_p}$
  $\hat{\mathbf{e}}_{t+1} \leftarrow \mathbf{Q}_t\mathbf{k}_{x_p}$
  if $\gamma_{t+1} < \varepsilon_{tol}$ **then**
    $\hat{\mathbf{s}}_{t+1} \leftarrow \mathbf{C}_t\mathbf{k}_{x_p} + \hat{\mathbf{e}}_{t+1}$ *Perform Reduced Update*
    $\boldsymbol{\alpha}_{t+1} \leftarrow \boldsymbol{\alpha}_t + q^{(t+1)}\hat{\mathbf{s}}_{t+1}$
    $\mathbf{C}_{t+1} \leftarrow \mathbf{C}_t + r^{(t+1)}\hat{\mathbf{s}}_{t+1}\hat{\mathbf{s}}_{t+1}^T$
  else
    $\mathbf{s}_{t+1} \leftarrow T_{t+1}(\mathbf{C}_t\mathbf{k}_{t+1}) + \mathbf{e}_{t+1}$ *Perform Full Update*
    $\boldsymbol{\alpha}_{t+1} \leftarrow T_{t+1}(\boldsymbol{\alpha}_t) + q^{(t+1)}\mathbf{s}_{t+1}$
    $\mathbf{C}_{t+1} \leftarrow U_{t+1}(\mathbf{C}_t) + r^{(t+1)}\mathbf{s}_{t+1}\mathbf{s}_{t+1}^T$
    Add $\mathbf{x}_p$ to Basis Vector Matrix $\mathbf{B}$
    $N_b \leftarrow N_b + 1$
    $\mathbf{Q}_{t+1} \leftarrow U_{t+1}(\mathbf{Q}_t) + \gamma_{t+1}^{-1}(T_{t+1}(\hat{\mathbf{e}}_{t+1}) - \mathbf{e}_{t+1})(T_{t+1}(\hat{\mathbf{e}}_{t+1}) - \mathbf{e}_{t+1})^T$
    if $N_b > M_{BV}$ **then**
      $\varepsilon_i \leftarrow |\alpha_{t+1}(i)|/Q_{t+1}(i,i)$ where $i \leftarrow 1 \ldots N_b$ *Calculate BV Scores*
      $i^* \leftarrow$ Position of minimum value of ε_i
      $\hat{\boldsymbol{\alpha}} \leftarrow \boldsymbol{\alpha}^{(t)} - \alpha^* \frac{\mathbf{Q}^*}{q^*}$ *Removal Equations*
      $\hat{\mathbf{C}} \leftarrow \mathbf{C}^{(t)} + (c^*\mathbf{Q}^*\mathbf{Q}^{*T})/q^{*2} - \left[\mathbf{Q}^*\mathbf{C}^{*T} + \mathbf{C}^*\mathbf{Q}^{*T}\right]/q^*$
      $\hat{\mathbf{Q}} \leftarrow \mathbf{Q}^{(t)} - (\mathbf{Q}^*\mathbf{Q}^{*T})/q^*$ *For definitions, see figure 1*
      Remove $\mathbf{b}_{i^*}$ from $\mathbf{B}$
      $N_b \leftarrow N_b - 1$
    end if
  end if
  Remove $\mathbf{x}_p$ from training set $\mathbf{X}$
end for

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha^{(t)} \\ \hline \alpha^* \\ \hline \alpha^{(t)} \end{bmatrix}, \mathbf{C} = \left[\begin{array}{c|c|c} \mathbf{C}^{(t)} & \mathbf{C}^* & \mathbf{C}^{(t)} \\ \hline \mathbf{C}^* & c^* & \mathbf{C}^* \\ \hline \mathbf{C}^{(t)} & \mathbf{C}^* & \mathbf{C}^{(t)} \end{array}\right], \mathbf{Q} = \left[\begin{array}{c|c|c} \mathbf{Q}^{(t)} & \mathbf{Q}^* & \mathbf{Q}^{(t)} \\ \hline \mathbf{Q}^* & q^* & \mathbf{Q}^* \\ \hline \mathbf{Q}^{(t)} & \mathbf{Q}^* & \mathbf{Q}^{(t)} \end{array}\right]$$

Figure 1: Decomposition of $\boldsymbol{\alpha}$,$\mathbf{Q}$,$\mathbf{C}$ matrices for point removal

margin band (noisy datasets). Here, we propose using the entropy of the selected point. For binary classification, for example, it is apparent that the maximally informative unlabeled example is that point with posterior probability closest to 0.5. From (8), it can be seen that this gives us the maximum possible cross entropy value of $-\log_2 0.5$ or 1. Similarly, assuming that we are learning the underlying function correctly, training points that we classify with a probability approaching 1 provide us with little information (they have a cross-entropy value approaching 0). Figure 2(b) shows how the entropy of the chosen point varies as the points are learnt for the majority rule (learning curves can be seen in 2(a)). In figure 2(b), the dotted curve corresponds to the entropy of the points chosen by active selection whilst the solid curve corresponds to the points being chosen randomly. Both curves are averaged over 100 numerical experiments. When the entropy curve reaches the x axis no further gain is being made by learning more points. This is supported by a levelling of the active learning curve in figure 2(a) in the same region. The passive learning entropy curve falls more slowy than the active learning curve indicating that the passive learner is still learning from examples well after the active learner has finished.

3.3 Regression

The first regression task we shall consider is for learning the $sin(x_1 x_2)$ function: given the values of x_1 and x_2 (this is our input $\mathbf{x}$ vector) the task is to predict a new value of $sin(x_1 x_2)$. The results for this dataset can be seen in figure 3(a). 100 partitions of the 441 training points into 200 training and 220 test examples were used with the following parameter values: $M_{BV} = 50$, $\sigma_k^2 = 1$, $\sigma_0^2 = 0.02$, $\varepsilon_{tol} = 1e-4$. The second regression problem is based on the Mackey-Glass time series - a classic benchmark regression problem. The task is to predict the $(n+1)$'th value given as an input of n consecutive values from the time series. In this example, $n = 5$ and the curves in figure 3(b) were created using 100 different partitions of the total 400 points into 200 training and test points. The other parameter values are as follows: $M_{BV} = 50$, $\sigma_K^2 = 1$, $\sigma_0^2 = 0.02$ and $\varepsilon_{tol} = 1e-6$. Our final regression data set is a QSAR dataset (for determining quantitative structure activity relationships), see [9]. It involves inferring the relationship between the physical structure of a chemical compound and its associated activity. 100 partitions of the training data into 100 training points and 86 test points were performed with the following parameter values: $M_{BV} = 100$, $\sigma_k^2 = 5$, $\sigma_0^2 = 5$, $\varepsilon_{tol} = 1e-6$. For all these cases, the lower curve corresponds to active selection and the upper curve to passive selection. The performance measure is the average mean square difference between actual and predicted values.

4 Conclusion.

We have outlined a framework for active learning using Gaussian Processes. In the introduction we mentioned various schemes for active learning using classifiers but corresponding approaches for regression appear much less studied. On the other hand active selection for regression is potentially important in many real-life applications. For example, for QSAR the measurements are generally continuous-valued, but each

experiment is costly and time-consuming. In this paper we have adapted an online algorithm for training GPs to perform active learning, both for regression and classification. Other approaches involving GPs and further methods can also be used for these active learning tasks and in a subsequent paper we expect to present a more detailed analysis.

Acknowledgements: We would like to thank Manfred Opper and Lehel Csato for discussions.

References

[1] J. Andras. Cleveland heart database: http://www.ics.uci.edu/ mlearn/mlrepository.html.

[2] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

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

[4] C Campbell, N Christianini, and A Somola. Query learning with large margin classifiers. *Proceedings of the 17th International Conference on Machine Learning*, pages 111–118, 2000.

[5] L. Csató. *Gaussian Processes - Iterative Sparse Approximations*. PhD thesis, Aston University, Neural Computation Research Group, 2002.

[6] L. Csató and M. Opper. Sparse on-line gaussian processes. *Neural Computation*, 14:641–668, 2002.

[7] Y. Freund, S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168, 1997.

[8] R. Herbrich, T. Graepel, and C. Campbell. Bayes point machines. *Journal of Machine Learning Research*, 1:245–279, 2001.

[9] J.D. King. A comparison of artificial intelligence methods for modelling QSARs. *Applied Artificial Intelligence*, 1994.

[10] D.J.C. Mackay. Introduction to gaussian processes. In Christopher M. Bishop, editor, *Neural Networks and Machine Learning*, pages 133–164. NATO Advanced Study Institute, Springer, 1997.

[11] R.M. Neal. *Bayesian Learning for Neural Networks (Lecture Notes in Statistics 118)*. Springer, 1996.

[12] R.L. Rivest and B. Eisenberg. On the sample complexity of pac-learning using random and chosen examples. In *Proceedings of the 1990 Workshop on Computational Learning Theory. San Mateo, CA*. Morgan Kauffman, 1990.

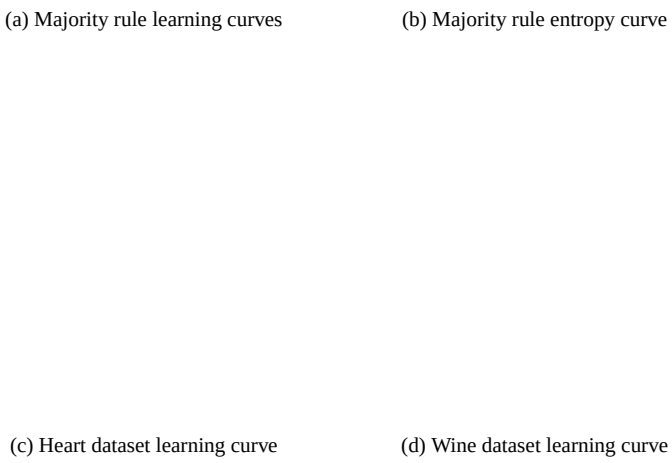
[13] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning. Stanford, CA.*, 2000.

[14] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

[15] M.K. Warmuth, J. Liao, G. Rätsch, M. Mathieson, S. Putta, and C. Lemmen. Active learning with svms in the drug discovery process. *Chemical Information and Computer Sciences*, accepted, 2003.

[16] T L H Watkin, A Rau, and M Biehl. The statistical mechanics of learning a rule. *Rev. Mod. Phys.*, 65:499–556, 1993.

[17] T.L.H. Watkin and A. Rau. Selecting examples for perceptrons. *J. Phys.*, A25:113–121, 1992.



(a) Majority rule learning curves (b) Majority rule entropy curve

(c) Heart dataset learning curve (d) Wine dataset learning curve

Figure 2: Learning curves and an entropy plot. In plots (a), (c) and (d) the number of errors (y-axis) is plotted against number of examples (x-axis). The upper curve gives the performance with passive learning, the lower gives active learning. For the real-life noisy datasets in (c) and (d) prototypical examples are learnt first, then the outliers in the data, resulting in the test error curve climbing as outliers are learnt. The entropy plot (b) shows that the information gain is more efficient with active learning (dashed curve) rather than passive learning (solid mean curve), for the majority rule in (a).

(a) $sin(x_1 x_2)$ (b) Mackey Glass

(c) QSAR

Figure 3: Regression learning curves. The error (averaged mean squared difference) on test points (y-axis) versus number of examples (x-axis) for three regression tasks. The upper curves are for passive learning and the lower are for active learning.