Predictive Automatic Relevance Determination by Expectation Propagation

Yuan (Alan) Qi YUANQI@MEDIA.MIT.EDU
MIT Media Laboratory, Cambridge, MA, 02139 USA

Thomas P. Minka MINKA@MICROSOFT.COM
Microsoft Research, 7 J J Thomson Ave, Cambridge, CB3 0FB, UK

Rosalind W. Picard PICARD@MEDIA.MIT.EDU
MIT Media Laboratory, Cambridge, MA, 02139 USA

Zoubin Ghahramani ZOUBIN@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit, UCL, 17 Queen Square, London, WC1N 3AR, UK

Abstract

In many real-world classification problems the input contains a large number of potentially irrelevant features. This paper proposes a new Bayesian framework for determining the relevance of input features. This approach extends one of the most successful Bayesian methods for feature selection and sparse learning, known as Automatic Relevance Determination (ARD). ARD finds the relevance of features by optimizing the model marginal likelihood, also known as the evidence. We show that this can lead to overfitting. To address this problem, we propose *Predictive ARD* based on estimating the predictive performance of the classifier. While the actual leave-one-out predictive performance is generally very costly to compute, the expectation propagation (EP) algorithm proposed by Minka provides an estimate of this predictive performance as a side-effect of its iterations. We exploit this in our algorithm to do feature selection, and to select data points in a sparse Bayesian kernel classifier. Moreover, we provide two other improvements to previous algorithms, by replacing Laplace's approximation with the generally more accurate EP, and by incorporating the fast optimization algorithm proposed by Faul and Tipping. Our experiments show that our method based on the EP estimate of predictive performance is more accurate on test data than relevance determination by optimizing the evidence.

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.

1. Introduction

In many real-world classification and regression problems the input consists of a large number of features or variables, only some of which are relevant. Inferring which inputs are relevant is an important problem. It has received a great deal of attention in machine learning and statistics over the last few decades (Guyon & Elisseeff, 2003).

This paper focuses on Bayesian approaches to determining the relevance of input features. One of the most successful methods is called *Automatic Relevance Determination* (ARD) (MacKay, 1992; Neal, 1996). This is a hierarchical Bayesian approach where there are hyperparameters which explicitly represent the relevance of different input features. These relevance hyperparameters determine the range of variation for the parameters relating to a particular input, usually by modelling the width of a zero-mean Gaussian prior on those parameters. If the width of that Gaussian is zero, then those parameters are constrained to be zero, and the corresponding input cannot have any effect on the predictions, therefore making it irrelevant. ARD optimizes these hyperparameters to discover which inputs are relevant. [1]

Automatic Relevance Determination optimizes the

[1] From a strictly Bayesian point of view, hard decisions about whether an input feature should be selected or not are not warranted unless there is a loss function which explicitly associates a cost to the number of features. However, in practice it is often desirable to have an easily interpretable model with a sparse subset of relevant features, and ARD methods can achieve this while closely approximating the full Bayesian average which does no feature selection.

model evidence, also known as the *marginal likelihood*, which is the classic criterion used for Bayesian model selection. In this paper we show that while this is often effective, in cases where there are a very large number of input features it can lead to overfitting. We instead propose a different approach, called *predictive ARD*, which is based on the estimated predictive performance. We show that this estimated predictive performance can be computed efficiently as a side effect of the expectation propagation algorithm for approximate inference and that it performs better that the evidence-based ARD on a variety of classification problems.

Although the framework we present can be applied to many Bayesian classification and regression models, we focus our presentation and experiments on classification problems in the presence of irrelevant features as well as in sparse Bayesian learning for kernel methods.

Compared to the traditional ARD classification, this paper presents three specific enhancements: (1) an approximation of the integrals via Expectation Propagation, instead of Laplace's method or Monte Carlo; (2) an ARD procedure which minimizes an estimate of the predictive leave-one-out generalization error (obtained directly from EP); (3) a fast sequential update for the hyperparameters based on Faul and Tipping (2002)'s recent work. These enhancements improve classification performance.

The rest of this paper is organized as follows. Section 2 reviews the ARD approach to classification and its properties. Section 3 presents predictive ARD by EP, followed by experiments and discussions in section 4.

2. Automatic Relevance Determination

A linear classifier classifies a point $\mathbf{x}$ according to $t = \text{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x})$ for some parameter vector $\mathbf{w}$ (the two classes are $t = \pm 1$). Given a training set $D = \{(\mathbf{x}_1, t_1), ..., (\mathbf{x}_N, t_N)\}$, the likelihood for $\mathbf{w}$ can be written as

$$p(\mathbf{t}|\mathbf{w}, X) = \prod_i p(t_i|\mathbf{x}_i, \mathbf{w}) = \prod_i \Psi(t_i \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_i)) \tag{1}$$

where $\mathbf{t} = \{t_i\}_{i=1}^N$, $X = \{\mathbf{x}_i\}_{i=1}^N$, $\Psi(\cdot)$ is the cumulative distribution function for a Gaussian. One can also use the step function or logistic function as $\Psi(\cdot)$. The basis function $\boldsymbol{\phi}^T(\mathbf{x}_i)$ allows the classification boundary to be nonlinear in the original features. This is the same likelihood used in logistic regression and in Gaussian process classifiers. Given a new input $\mathbf{x}_{N+1}$, we approximate the predictive distribution:

$$p(t_{N+1}|\mathbf{x}_{N+1}, \mathbf{t}) = \int p(t_{N+1}|\mathbf{x}_{N+1}, \mathbf{w}) p(\mathbf{w}|\mathbf{t}) d\mathbf{w} \tag{2}$$

$$\approx p(t_{N+1}|\mathbf{x}_{N+1}, \langle \mathbf{w} \rangle) \tag{3}$$

where $\langle \mathbf{w} \rangle$ denotes the posterior mean of the weights, called the Bayes Point (Herbrich et al., 1999).

The basic idea in ARD is to give the feature weights independent Gaussian priors:

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_i \mathcal{N}(w_i|0, \alpha_i^{-1}),$$

where $\boldsymbol{\alpha} = \{\alpha_i\}$ is a hyperparameter vector that controls how far away from zero each weight is allowed to go. The hyperparameters $\boldsymbol{\alpha}$ are trained from the data by maximizing the Bayesian 'evidence' $p(\mathbf{t}|\boldsymbol{\alpha})$, which can be done using a fixed point algorithm or an EM algorithm treating $\mathbf{w}$ as a hidden variable (MacKay, 1992). The outcome of this optimization is that many elements of $\boldsymbol{\alpha}$ go to infinity such that $\mathbf{w}$ would have only a few nonzero weights w_j. This naturally prunes irrelevant features in the data. Later we will discuss why ARD favors sparse models (section 2.2).

2.1. ARD-Laplace

Both the fixed point and EM algorithms require the posterior moments of $\mathbf{w}$. These moments have been approximated by second-order expansion, i.e. Laplace's method (MacKay, 1992), or approximated by Monte Carlo (Neal, 1996). ARD with Laplace's method (**ARD-Laplace**) was used in the Relevance Vector Machine (RVM) (Tipping, 2000). The RVM is a linear classifier using basis functions $\boldsymbol{\phi}(\mathbf{x}) = [k(\mathbf{x}, \mathbf{x}_1), k(\mathbf{x}, \mathbf{x}_2), \cdots, k(\mathbf{x}, \mathbf{x}_N)]$. Specifically, Laplace's method approximates the evidence by a Gaussian distribution around the *maximum a posteriori* (MP) value of $\mathbf{w}$, $\mathbf{w}_{MP}$, as follows:

$$\Sigma = -\mathbf{H}^{-1} = -\frac{d^2 \log p(\mathbf{w}, \mathbf{t}|\boldsymbol{\alpha})}{d\mathbf{w} d\mathbf{w}^T}\Big|_{\mathbf{w}=\mathbf{w}_{MP}}^{-1}$$

$$p(\mathbf{w}, \mathbf{t}|\boldsymbol{\alpha}) \approx p(\mathbf{t}, \mathbf{w}_{MP}) \exp\big(-\frac{1}{2}(\mathbf{w} - \mathbf{w}_{MP})^T \Sigma^{-1} (\mathbf{w} - \mathbf{w}_{MP})\big)$$

$$p(\mathbf{t}|\boldsymbol{\alpha}) = \int p(\mathbf{w}, \mathbf{t}|\boldsymbol{\alpha}) d\mathbf{w} \approx p(\mathbf{t}, \mathbf{w}_{MP}) |2\pi\Sigma|^{1/2}$$

$$p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}) = \frac{p(\mathbf{w}, \mathbf{t}|\boldsymbol{\alpha})}{p(\mathbf{t}|\boldsymbol{\alpha})} \approx \mathcal{N}(\mathbf{w}|\mathbf{w}_{MP}, \Sigma)$$

If we use a logistic model for $\Psi(\cdot)$, then the Hessian matrix $\mathbf{H}$ has the following form: $H = -(\Phi B \Phi^{\mathrm{T}} + A)$, where $\Phi = \big(\phi(\mathbf{x}_i), \ldots, \phi(\mathbf{x}_N)\big)$ is a d by N matrix, $A = \text{diag}(\boldsymbol{\alpha})$, and B is a diagonal matrix with $B_{ii} = \Psi\big(\mathbf{w}_{MP}^{\mathrm{T}} \mathbf{x}_i\big)\big(1 - \Psi(\mathbf{w}_{MP}^{\mathrm{T}} \mathbf{x}_i)\big)$.

Laplace's method is a simple and powerful approach for approximating a posterior distribution. But it does not really try to approximate the posterior mean; instead it simply approximates the posterior mean by the posterior mode. The quality of the approximation for the posterior mean can be improved by using EP as shown by Minka (2001).

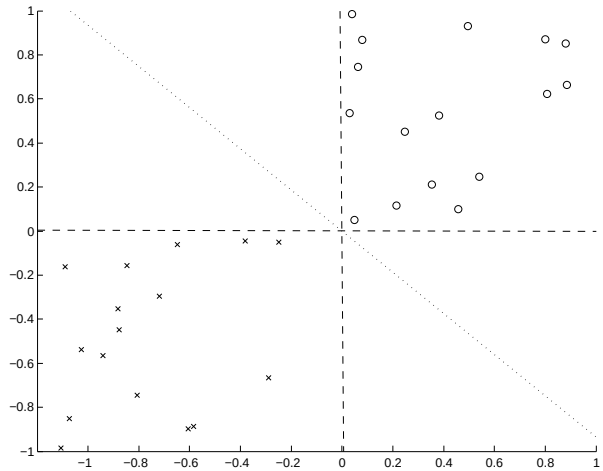


Figure 1. Illustration of overfitting of ARD.

2.2. Overfitting of ARD

Overfitting can be caused not only by over-complicated classifiers, but also by just picking one from many simple classifiers that can correctly classify the data. Consider the example plotted in figure 1. The data labelled with 'x' and 'o' are in class 1 and 2 respectively. Every line through the origin with negative slope separates the data. If you apply the regular Bayes Point linear classifier, you get a classifier of angle 135° (shown); if you apply ARD to maximize evidence, you end up with a horizontal line which is sparse, becaues it ignores one input feature, but seemingly very dangerous. Both horizontal and vertical sparse classifiers have larger evidence values than the one of 135°, though both of them are intuitively more dangerous. Having effectively pruned out one of the two parameter dimensions (by taking one of the αs to infinity), the evidence for the horizontal and vertical classifiers involves computing an integral over only one remaining dimension. However, the evidence for the classifier which retains both input dimensions is an integral over two parameter dimensions. In general, a more complex model will have lower evidence than a simpler model if they can both classify the data equally well. Thus ARD using evidence maximization chooses the "simpler" model, which in this case is more dangerous because it uses only one relevant dimension.

ARD is a Type II maximum likelihood method and thus subject to overfitting as the somewhat dramatic albeit contrived example above illustrates. However, it is important to point out that the overfitting resulting from fitting the relevance hyperparameters $\boldsymbol{\alpha}$ is not the same kind of overfitting as one gets from fitting the parameters $\mathbf{w}$ as in classical maximum likelihood methods. By optimizing $\mathbf{w}$ one can directly fit noise in the data. However, optimizing $\boldsymbol{\alpha}$ only corresponds to making decisions about which variables are irrelevant, since $\mathbf{w}$ is integrated out. In the simple case where α can take only two values, very large or small, and the input is d-dimensional, choosing $\boldsymbol{\alpha}$ corresponds to model selection by picking one out of 2^d subsets of input features. This is only d bits of information in the training data that can be overfit, far fewer than what can be achieved by precisely tuning a single real-valued parameter w. However, when d is large, as in the practical examples in our experimental section, we find that even this overfitting can cause problems. This motivates our proposed predictive measure for ARD based on estimating leave-one-out performance.

2.3. Computational Issues

To compute the posterior moments of $\mathbf{w}$ required by ARD-Laplace, we need to invert the Hessian matrix $\mathbf{H}$ for each update. This takes $O(d^3)$ time, which is quite expensive when the dimension d is large.

3. Predictive-ARD-EP

In this section, we improve ARD-Laplace in three ways: replacing Laplace's method by more accurate EP, estimating the predictive performance based on leave-one-out estimate without actually carrying out the expensive cross-validation, and incorporating a fast sequential optimization method into ARD-EP.

3.1. EP for Probit Model

The algorithm described in this section is a variant of the one in (Minka, 2001), and we refer the reader to that paper for a detailed derivation. Briefly, Expectation Propagation (EP) exploits the fact that the likelihood is a product of simple terms. If we approximate each of these terms well, we can get a good approximation to the posterior. Expectation Propagation chooses each approximation such that the posterior using the term exactly and the posterior using the term approximately are close in KL-divergence. This gives a system of coupled equations for the approximations which are iterated to reach a fixed-point.

Denote the exact terms by $g_i(\mathbf{w})$ and the approximate terms by $\tilde{g}_i(\mathbf{w})$:

$$\begin{aligned} p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha}) &\propto p(\mathbf{w}|\boldsymbol{\alpha}) \prod_i p(t_i|\mathbf{w}) \\ &= p(\mathbf{w}|\boldsymbol{\alpha}) \prod_i g_i(\mathbf{w}) \approx p(\mathbf{w}|\boldsymbol{\alpha}) \prod_i \tilde{g}_i(\mathbf{w}) \end{aligned}$$

The approximate terms are chosen to be Gaussian, parameterized by (m_i, v_i, s_i): $\tilde{g}_i = s_i \exp(-\frac{1}{2v_i}(t_i \boldsymbol{\phi}_i^T \mathbf{w} -$

$m_i)^2$). This makes the approximate posterior distribution also Gaussian: $p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha}) \approx q(\mathbf{w}) = \mathcal{N}(\mathbf{m}_w, \mathbf{V}_w)$.

To find the best term approximations, proceed as follows: (to save notation, $t_i\boldsymbol{\phi}_i$ is written as $\boldsymbol{\phi}_i$)

1. Initialization Step: Set $\tilde{g}_i = 1$: $v_i = \infty$, $m_i = 0$, and $s_i = 1$. Also, set $q(\mathbf{w}) = p(\mathbf{w}|\boldsymbol{\alpha})$.

2. Loop until all (m_i, v_i, s_i) converge:

   Loop $i = 1, \ldots, N$:

   (a) Remove the approximation $\tilde{g}_i$ from $q(\mathbf{w})$ to get the 'leave-one-out' posterior $q^{\setminus i}(\mathbf{w})$, which is also Gaussian: $\mathcal{N}(\mathbf{m}_w^{\setminus i}, \mathbf{V}_w^{\setminus i})$. From $q^{\setminus i}(\mathbf{w}) \propto q(\mathbf{w})/\tilde{g}_i$, this implies

$$\mathbf{V}_w^{\setminus i} = \mathbf{V}_w + \frac{(\mathbf{V}_w\boldsymbol{\phi}_i)(\mathbf{V}_w\boldsymbol{\phi}_i)^T}{v_i - \boldsymbol{\phi}_i^T\mathbf{V}_w\boldsymbol{\phi}_i} \tag{4}$$

$$\mathbf{m}_w^{\setminus i} = \mathbf{m}_w + (\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i)v_i^{-1}(\boldsymbol{\phi}_i^T\mathbf{m}_w - m_i) \tag{5}$$

   (b) Putting the posterior without i together with term i gives $\hat{p}(\mathbf{w}) \propto g_i(\mathbf{w})q^{\setminus i}(\mathbf{w})$. Choose $q(\mathbf{w})$ to minimize $KL(\hat{p}(\mathbf{w}) \mid\mid q(\mathbf{w}))$. Let Z_i be the normalizing factor.

$$\mathbf{m}_w = \mathbf{m}_w^{\setminus i} + \mathbf{V}_w^{\setminus i}\rho_i\boldsymbol{\phi}_i$$

$$\mathbf{V}_w = \mathbf{V}_w^{\setminus i} - (\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i)\Big(\frac{\rho_i(\boldsymbol{\phi}_i^T\mathbf{m}_w + \rho_i)}{\boldsymbol{\phi}_i^T\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i + 1}\Big)(\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i)^T$$

$$Z_i = \int_{\mathbf{w}} g_i(\mathbf{w})q^{\setminus i}(\mathbf{w})\mathrm{d}\mathbf{w} = \Psi(z_i) \tag{6}$$

   where

$$z_i = \frac{(\mathbf{m}_w^{\setminus i})^T\boldsymbol{\phi}_i}{\sqrt{\boldsymbol{\phi}_i^T\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i + 1}} \qquad \rho_i = \frac{1}{\sqrt{\boldsymbol{\phi}_i^T\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i + 1}}\frac{\mathcal{N}(z_i;0,1)}{\Psi(z_i)} \tag{7}$$

   (c) From $\tilde{g}_i = Z_i\frac{q(\mathbf{w})}{q^{\setminus i}(\mathbf{w})}$, update the term approximation:

$$v_i = \boldsymbol{\phi}_i^T\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i\Big(\frac{1}{\rho_i(\boldsymbol{\phi}_i^T\mathbf{m}_w + \rho_i)} - 1\Big) + \frac{1}{\rho_i(\boldsymbol{\phi}_i^T\mathbf{m}_w + \rho_i)} \tag{8}$$

$$m_i = \boldsymbol{\phi}_i^T\mathbf{m}_w^{\setminus i} + (v_i + \boldsymbol{\phi}_i^T\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i)\rho_i \tag{9}$$

$$s_i = \Psi(z_i)\sqrt{1 + v_i^{-1}\boldsymbol{\phi}_i^T\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i}\exp\Big(\frac{\boldsymbol{\phi}_i^T\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i + 1}{\boldsymbol{\phi}_i^T\mathbf{m}_w + \rho_i}\rho_i\Big) \tag{10}$$

3. Finally, compute the normalizing constant and the evidence:

$$B = (\mathbf{m}_w)^T\mathbf{V}_w\mathbf{m}_w - \sum_i \frac{m_i^2}{v_i}$$

$$p(D|\boldsymbol{\alpha}) \approx \int \prod_i p(\mathbf{w}|\boldsymbol{\alpha})\tilde{g}_i(\mathbf{w})\mathrm{d}\mathbf{w} = \frac{|\mathbf{V}_w|^{1/2}}{(\prod_j \alpha_j)^{1/2}}\exp(B/2)\prod_i s_i \tag{11}$$

The time complexity of this algorithm is $O(d^2)$ for processing each term, and therefore $O(Nd^2)$ per iteration.

3.2. Estimate of Predictive Performance

A nice property of EP is that it can easily offer an estimate of leave-one-out error without any extra computation. At each iteration, EP computes in (4) and (5) the parameters of the approximate leave-one-out posterior $q^{\setminus i}(\mathbf{w})$ that does not depend on the ith data point. So we can use the mean $\mathbf{m}_w^{\setminus i}$ to approximate a classifier trained on the other $(N-1)$ data points. Thus an estimate of leave-one-out error can be computed as

$$\epsilon_{loo} = \frac{1}{N}\sum_{i=1}^{N}\Theta(-t_i(\mathbf{m}_w^{\setminus i})^T\boldsymbol{\phi}(\mathbf{x}_i)) \tag{12}$$

where $\Theta(\cdot)$ is a step function. An equivalent estimate was given by Opper and Winther (2000) using the TAP method for training Gaussian processes, which is equivalent to EP (Minka, 2001).

Furthermore, we can provide an estimate of leave-one-out error probability. Since Z_i in (6) is the posterior probability of the ith data label, we propose the following estimator:

$$\epsilon_{pred} = \frac{1}{N}\sum_{i=1}^{N}(1 - Z_i) = \frac{1}{N}\sum_{i=1}^{N}\Psi(-z_i) \tag{13}$$

where Z_i and z_i is defined in (6) and (7). In (7), $\boldsymbol{\phi}_i$ is the product of t_i and $\phi(\mathbf{x}_i)$. By contrast, Opper and Winther estimate the error probability by $\frac{1}{N}\sum_{i=1}^{N}\Psi(-|z_i|)$, which ignores the information of the label t_i. Notice that ϵ_{pred} utilizes the variance of $\mathbf{w}$ given the data, not just the mean. However, it assumes that the posterior for $\mathbf{w}$ has Gaussian tails. In reality, the tails are lighter so we compensate by prescaling z_i by 50, a number tuned by simulations.

3.3. Fast Optimization of Evidence

This section combines EP with a fast sequential optimization method (Faul & Tipping, 2002) to efficiently update the hyperparameters $\boldsymbol{\alpha}$.

As mentioned before, EP approximates each classification likelihood term $g_i(\mathbf{w}) = \Psi(\mathbf{w}^{\mathrm{T}}\phi_i)$ by $\tilde{g}_i = s_i\exp(-\frac{1}{2v_i}(\boldsymbol{\phi}_i^T\mathbf{w} - m_i)^2)$. Here ϕ_i is short hand for $t_i\phi(\mathbf{x}_i)$ as in the previous section. Notice that $\tilde{g}_i$ has the same form as a regression likelihood term in a regression problem. Therefore, EP actually maps a classification problem into a regression problem where (m_i, v_i) defines the virtual observation data point with mean m_i and variance v_i. Based on this interpretation, it is easy to see that for the approximate posterior $q(\mathbf{w})$, we have

$$\mathbf{V}_w = \big(\mathbf{A} + \Phi\Lambda^{-1}\Phi^{\mathrm{T}}\big)^{-1} \qquad \mathbf{m}_w = \mathbf{V}_w\Phi\Lambda^{-1}\mathbf{m}_o \tag{14}$$

where we define

$$\mathbf{m}_o = (m_1, \ldots, m_N)^{\mathrm{T}} \qquad \mathbf{A} = \mathrm{diag}(\boldsymbol{\alpha})$$
$$\mathbf{v}_o = (v_1, \ldots, v_N)^{\mathrm{T}} \qquad \Lambda = \mathrm{diag}(\mathbf{v}_o)$$

and $\Phi = (\phi_i, \ldots, \phi_N)$ is a d by N matrix.

To have a sequential update on α_j, we can explicitly decompose $p(D|\boldsymbol{\alpha})$ into two parts, one part denoted by $p(D|\boldsymbol{\alpha}_{\setminus j})$, that does not depend on α_j and another that does, i.e.,

$$p(D|\boldsymbol{\alpha}) = p(D|\boldsymbol{\alpha}_{\setminus j}) + \frac{1}{2}\big(\log \alpha_j - \log(\alpha_j + r_j) + \frac{u_j^2}{\alpha_j + r_j}\big)$$

where $r_j = \phi_j \mathbf{C}_{\setminus j}^{-1} \phi_j^{\mathrm{T}}$, $u_j = \phi_j \mathbf{C}_{\setminus j}^{-1} \mathbf{m}_o$, and $\mathbf{C}_{\setminus j} = \Lambda^{-1} + \sum_{m \neq j} \phi_m^{\mathrm{T}} \phi_m$. Here ϕ_j and ϕ_m are j^{th} and m^{th} rows of the data matrix Φ respectively.

Using the above equation, Faul and Tipping (2002) show $p(D|\boldsymbol{\alpha})$ has a maximum with respect to α_j:

$$\alpha_j = \frac{r_j^2}{u_j^2 - r_j}, \qquad \text{if} \quad \eta_j > 0 \qquad (15)$$
$$\alpha_j = \infty, \qquad \text{if} \quad \eta_j \leq 0 \qquad (16)$$

where $\eta_j = u_j^2 - r_j$. Thus, in order to maximize the evidence, we introduce the jth feature when $\alpha_j = \infty$ and $\eta_j > 0$, exclude the jth feature when $\alpha_j < \infty$ and $\eta_j \leq 0$, and reestimate α_j according to (15) when $\alpha_j < \infty$ and $\eta_j > 0$. To further save computation, we can exploit the following relations:

$$r_j = \frac{\alpha_j R_j}{\alpha_j - R_j}, \quad u_j = \frac{\alpha_j U_j}{\alpha_j - R_j} \qquad (17)$$

where $R_j = \phi_j \Lambda^{-1} \phi_j^{\mathrm{T}} - \phi_j \Lambda^{-1} \hat{\Phi}^{\mathrm{T}} \hat{\mathbf{m}}_w$ and $U_j = \phi_j \Lambda^{-1} \mathbf{m}_o - \phi_j \Lambda^{-1} \hat{\Phi}^{\mathrm{T}} \hat{\mathbf{V}}_w \hat{\Phi} (\phi_j \Lambda^{-1})^{-1}$. where $\hat{\Phi}$ contains only the features that are currently included in the model, and $\hat{m}_w$ and $\hat{\mathbf{V}}_w$ are obtained based on these features. This observation allows us to efficiently compute the EP approximation and update $\boldsymbol{\alpha}$, since in general there are only a small set of the features in the model during the updates.

3.4. Algorithm Summary

To summarize the **predictive-ARD-EP** algorithm:

1. First, initialize the model so that it only contains a small fraction of features.

2. Then, sequentially update $\boldsymbol{\alpha}$ as in section 3.3 and calculate the required statistics by EP as in section 3.1 until the algorithm converges.

3. Finally, choose the classifier from the sequential updates with minimum leave-one-out error estimate (12). The leave-one-out error is discrete, so in case of a tie, choose the first classifier in the tie, i.e., the one with the smaller evidence.

A variant of this algorithm uses the error probability (13) and is called **predictive$_{Prob}$-ARD-EP**. Choosing the classifier with the maximum evidence (approximated by EP) is called **evidence-ARD-EP**.

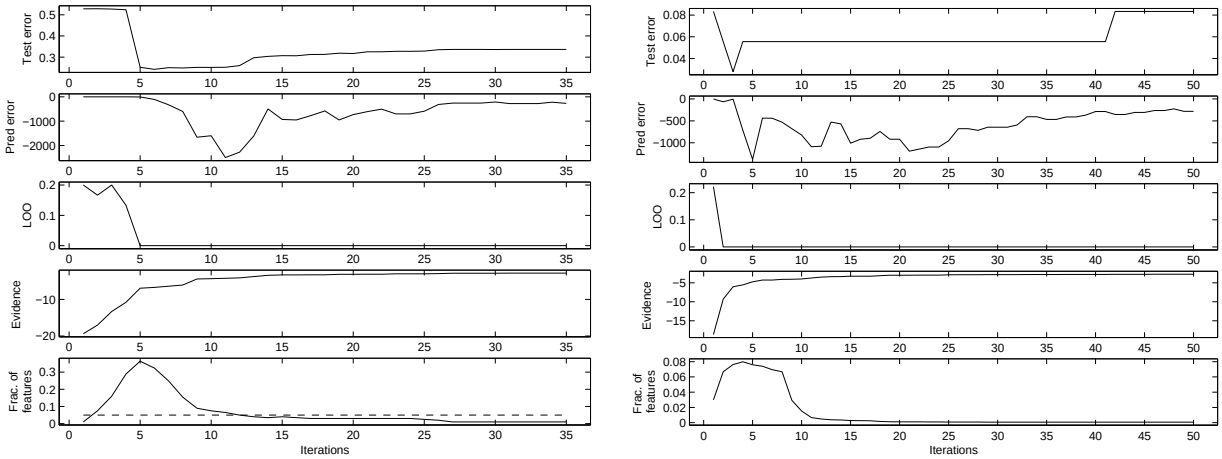
4. Experiments and Discussion

This section compares evidence-ARD-EP and predictive-ARD-EP on synthetic and real-world data sets. The first experiment has 30 random training points and 5000 random test points with dimension 200. True classifier weights consist of 10 Gaussian weights, sampled from $\mathcal{N}(-0.5, 1)$ and 190 zero weights. The true classifier is then used as the ground truth to label the data. Thus the data is guaranteed to be separable. The basis functions are simply $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}$. The results over 50 repetitions of this procedure are visualized in figure 3-(a). Both predictive-ARD-EP and predictive$_{Prob}$-ARD-EP outperform evidence-ARD-EP by picking the model with the smallest estimate of the predictive error, rather than choosing the most probable model.

Figure 2-(a) shows a typical run. As shown in the figure, the estimates of the predictive performance based on leave-one-out error count (12) and (log) error probability (13) are better correlated with the true test error than evidence and the fraction of features. The evidence is computed as in equation (11) and the fraction of features is defined as $\frac{||\mathbf{w}||_0}{d}$ where d is the dimension of the classifier $\mathbf{w}$. Also, since the data is designed to be linearly separable, we always get zero training error along the iterations. While the (log) evidence keeps increasing, the test error rate first decreases and then increases. This demonstrates the overfitting problem associated with maximizing evidence. As to the fraction of features, it first increases by adding new useful features into the model and then decreases by deleting old features. Notice that the fraction of features converges to a lower value than the true one, $\frac{10}{200} = 0.05$, which is plotted as the dashed line in figure2-(a). Moreover, choosing the sparsest model, i.e., the one with the smallest fraction of features, leads to overfitting here even though there is zero training error.

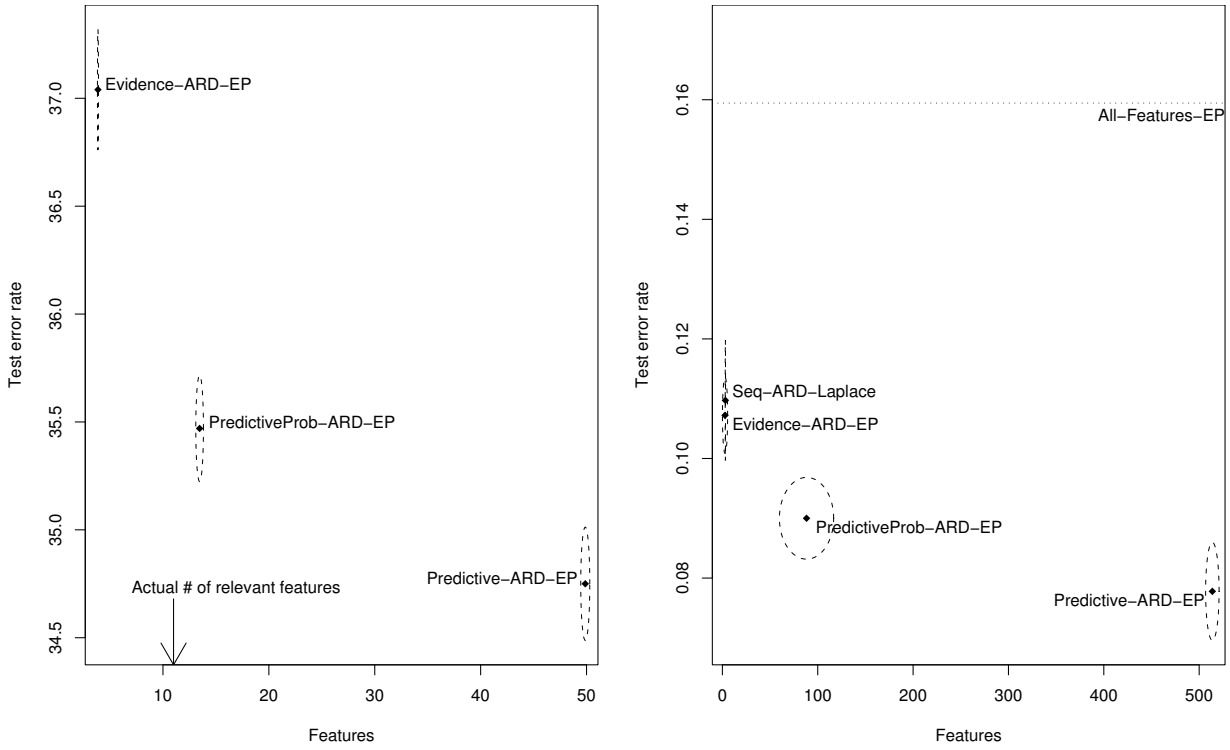
Next, the algorithms are applied to high-dimensional gene expression datasets: leukaemia and colon cancer.

For the leukaemia dataset, the task is to distin-



(a) Synthetic data classification (b) Leukaemia data classification

Figure 2. Comparison of different model selection criteria during ARD training. The second and third rows are computed via (13) and (12), respectively. The estimated predictive performance is better correlated with the test errors than evidence and sparsity.



(a) Synthetic data classification (b) Leukaemia data classification

Figure 3. Test errors and sizes of selected features. (a) synthetic dataset with 30 training and 5000 test data points. Each data point has 201 features, among which only 11 are useful. 50 random repetitions of the data were used. (b) leukaemia microarray dataset with 36 training and 36 test data points. Each data point has 7129 features. The results are averaged over 100 random partitions. Ellipses indicate standard errors over repetitions.

guish acute myeloid leukaemia (AML) from acute lymphoblastic leukaemia (ALL). The dataset has 47 and 25 samples of type ALL and AML respectively with 7129 features per sample. The dataset was randomly split 100 times into 36 training and 36 test samples, with evidence-ARD-EP and predictive-ARD-EP run

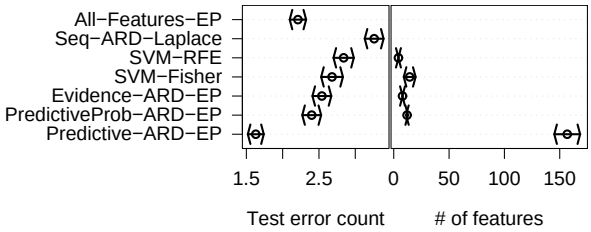


Figure 4. Test errors and sizes of selected feature sets on colon cancer microarray dataset with 50 training and 12 test data points. Each data point has 2000 features. 100 random partitionings of the data were used.

on each. Figure 2-(b) shows a typical run that again illustrates the overfitting phenomenon as shown in figure 2-(a). On most of runs including the one shown in figure 2-(b), there is zero training error from the first to the last iterations. The test performance is visualized in figure 3-(b). The error counts of evidence-ARD-EP and predictive-ARD-EP are 3.86 ± 0.14 and 2.80 ± 0.18, respectively. The numbers of the chosen features of these two methods are 2.78 ± 1.65 and 513.82 ± 4.30, respectively.

For the colon cancer dataset, the task is to discriminate tumour from normal tissues using microarray data. The whole dataset has 22 normal and 40 cancer samples with 2000 features per sample. We randomly split the dataset into 50 training and 12 test samples 100 times and run evidence-ARD-EP and predictive-ARD-EP on each partition. The test performance is visualized in figure 4. For comparison, we show the results from Li et al. (2002). The methods tested by Li et al. (2002) include ARD-Laplace with fast sequential updates on a logistic model (Seq-ARD-Laplace), Support Vector Machine (SVM) with recursive feature elimination (SVM-RFE), and SVM with Fisher score feature ranking (SVM-Fisher Score). The error counts of evidence-ARD-EP and predictive-ARD-EP are 2.54 ± 0.13 and 1.63 ± 0.11, respectively. The sizes of chosen feature sets for these two methods are 7.92 ± 0.14 and 156.76 ± 11.68, respectively.

As shown in figures 3-(b) and 4, pruning irrelevant features by maximizing evidence helps to reduce test errors. But aggressive pruning will overfit the model and therefore increase the test errors. For both colon cancer and leukaemia datasets, predictive-ARD-EP with a moderate number of features outperforms all the other methods including EP without feature pruning as well as the evidence-ARD-EP with only a few features left in the model.

Finally, RBF-type feature expansion can be combined with predictive-ARD-EP to obtain sparse nonlinear Bayesian classifiers. Specifically, we use the following

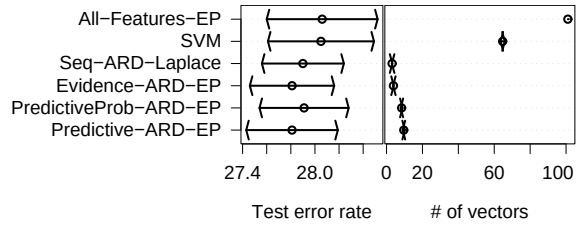


Figure 5. Test error rates and numbers of relevance or support vectors on breast cancer dataset. 50 partitionings of the data were used. All these methods use the same Gaussian kernel with kernel width $\sigma = 5$. The trade-off parameter C in SVM is chosen via 10-fold cross-validation for each partition.

Gaussian basis function $\phi(\mathbf{x}_i)$

$$\phi(\mathbf{x}_i) = [\mathbf{1}, k(\mathbf{x}_i, \mathbf{x}_1), \ldots, k(\mathbf{x}_i, \mathbf{x}_N)]^{\mathrm{T}}$$

where $k(\mathbf{x}_i, x_j) = \exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2})$. Unlike the ARD feature selection in the previous examples, here ARD is used to choose relevance vectors $\phi(\mathbf{x}_i)$, i.e., to select data points instead of features. The algorithms are tested on two UCI datasets: breast cancer and diabetes.

For the breast cancer dataset provided by Zwitter and Soklic (1998), the task is to distinguish no-recurrence-events from recurrence-events. We split the dataset into 100 training and 177 test samples 50 times and run evidence-ARD-EP and predictive-ARD-EP on each partition. The test performance is visualized in figure 5 and summarized in table 1.

Table 1. Test error rates and numbers of relevance or support vectors on breast cancer dataset.

Algorithm	Test error rate (%)	Size of feature set
SVM	28.05 ± 0.44	64.70 ± 1.17
EP	28.06 ± 0.46	101 ± 0
Seq-ARD-Laplace	27.90 ± 0.34	3.10 ± 0.13
Evd-ARD-EP	$\mathbf{27.81 \pm 0.35}$	3.84 ± 0.19
Pred_{Prob}-ARD-EP	27.91 ± 0.37	8.40 ± 0.54
Pred-ARD-EP	$\mathbf{27.81 \pm 0.38}$	9.60 ± 0.62

In this experiment, evidence-ARD-EP and predictive-ARD-EP marginally outperform the other alternatives, but give much simpler models. They only have about 4 or 10 relevance vectors while SVM uses about 65 support vectors.

Finally evidence-ARD-EP and predictive-ARD-EP are tested on the UCI diabetes dataset. Each is run on 100 random partitions of 468 training and 300 test samples. The partitions are the same as in Rätsch et al. (2001), so that we can directly compare our

results with theirs. The test performance is summarized in figure 4. The error rates of the evidence-ARD-EP and predictive-ARD-EP are $23.91 \pm 0.21\%$ and $23.96 \pm 0.20\%$, respectively.

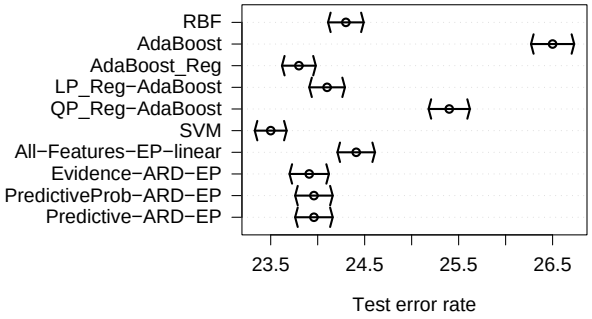


Figure 6. Test errors on diabetes dataset with 468 training and 300 test data points. The results are averaged over 100 partitions.

On the diabetes dataset, predictive-ARD-EP performs comparably or outperforms most of the other state-of-the-art methods; only SVM performs better. Note that the SVM's kernel has been optimized using the test data points, which is not possible in practice (Rätsch et al., 2001).

5. Conclusions

Predictive-ARD-EP is an efficient algorithm for feature selection and sparse learning. Predictive-ARD-EP chooses the model with the best estimate of the predictive performance instead of choosing the one with the largest marginal likelihood. On high-dimensional micorarray datasets, Predictive-ARD-EP outperforms other state-of-the-art algorithms in test accuracy. On UCI benchmark datasets, it results in sparser classifiers than SVMs with comparable test accuracy. The resulting sparse models can be used in applications where classification time is critical. To achieve a desired balance between test accuracy and testing time, one can choose a classifier that minimizes a loss function trading off the leave-one-out error estimate and the number of features.

The success of this algorithm argues against a few popular principles in learning theory. First, it argues against the evidence framework in which the evidence is maximized by tuning hyperparameters. Maximizing evidence is useful for choosing among a small set of distinct models, but can overfit if used with a large continuum of similar models, as in ARD. Second, our findings show that larger fraction of nonzero features (or lower sparsity) can lead to better generalization performance, even when the training error is zero. This is against the sparsity principles as well as Occam's razor. If what we care about is generalization performance, then it is better to minimize some measure of predictive performance as predictive ARD does.

Acknowledgements

Y. Qi was supported by the Things That Think consortium at the MIT Media laboratory and the work was partially performed during his visit to the Gatsby Unit, University Colledge London. Thanks to W. Chu for providing the gene expression datasets, J. Kandola for offering the sequential ARD-Laplace matlab codes, and F. Pérez-Cruz for useful discussions on SVMs. Z. Ghahramani was partially supported from CMU by DARPA under the CALO project.

References

Faul, A. C., & Tipping, M. E. (2002). Analysis of sparse bayesian learning. *Advances in Neural Information Processing Systems 14* (pp. 383–389).

Guyon, I., & Elisseeff, A. (2003). Special issue on variable and feature selection. *Journal of Machine Learning Research.* http://www.jmlr.org/papers/special/feature.html.

Herbrich, R., Graepel, T., & Campbell, C. (1999). Bayes point machine: Estimating the Bayes point in kernel space. *IJCAI Workshop SVMs* (pp. 23–27).

Li, Y., Campbell, C., & Tipping, M. E. (2002). Bayesian automatic relevance determination algorithms for classifying gene expression data. *Bioinformatics*, *18*, 1332–1339.

MacKay, D. J. (1992). Bayesian interpolation. *Neural Computation*, *4*, 415–447.

Minka, T. P. (2001). Expectation propagation for approximate Bayesian inference. *Uncertainty in AI'01.* `http://www.stat.cmu.edu/~minka/papers/ep/`.

Neal, R. M. (1996). *Bayesian learning for neural networks.* No. 118 in Lecture Notes in Statistics. New York: Springer.

Opper, M., & Winther, O. (2000). Gaussian processes for classification: Mean field algorithms. *Neural Computation.*

Rätsch, G., Onoda, T., & Müller, K.-R. (2001). Soft margins for AdaBoost. *Machine Learning*, *42*, 287–320. also NeuroCOLT Technical Report NC-TR-1998-021. In press.

Tipping, M. E. (2000). The relevance vector machine. *NIPS* (pp. 652–658). The MIT Press.

Zwitter, M., & Soklic, M. (1998). This breast cancer domain was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia.