Bayesian Online Classifiers for Text Classification and Filtering

Kian Ming Adam Chai
DSO National Laboratories
20 Science Park Drive
Singapore 118230
ckianmin@dso.org.sg

Hwee Tou Ng
Department of Computer Science
National University of Singapore
3 Science Drive 2
Singapore 117543
nght@comp.nus.edu.sg

Hai Leong Chieu
DSO National Laboratories
20 Science Park Drive
Singapore 118230
chaileon@dso.org.sg

ABSTRACT

This paper explores the use of Bayesian online classifiers to classify text documents. Empirical results indicate that these classifiers are comparable with the best text classification systems. Furthermore, the online approach offers the advantage of continuous learning in the batch-adaptive text filtering task.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval—*Information filtering*

General Terms

Algorithms, Experimentation

Keywords

Text Classification, Text Filtering, Bayesian, Online, Machine Learning

1. INTRODUCTION

Faced with massive information everyday, we need automated means for classifying text documents. Since handcrafting text classifiers is a tedious process, machine learning methods can assist in solving this problem[15, 7, 27].

Yang & Liu[27] provides a comprehensive comparison of supervised machine learning methods for text classification. In this paper we will show that certain Bayesian classifiers are comparable with *Support Vector Machines*[23], one of the best methods reported in [27]. In particular, we will evaluate the *Bayesian online perceptron*[17, 20] and the *Bayesian online Gaussian process*[3].

For text classification and filtering, where the initial training set is large, online approaches are useful because they allow continuous learning without storing all the previously seen data. This continuous learning allows the utilization of information obtained from subsequent data after the initial training. Bayes' rule allows online learning to be performed in a principled way[16, 20, 17]. We will evaluate the Bayesian online perceptron, together with information gain considerations, on the batch-adaptive filtering task[18].

2. CLASSIFICATION AND FILTERING

For the text classification task defined by Lewis[9], we have a set of predefined categories and a set of documents. For each category, the document set is partitioned into two mutually exclusive sets of relevant and irrelevant documents. The goal of a text classification system is to determine whether a given document belongs to any of the predefined categories. Since the document can belong to zero, one, or more categories, the system can be a collection of binary classifiers, in which one classifier classifies for one category.

In *Text REtrieval Conference* (TREC), the above task is known as batch filtering. We will consider a variant of batch filtering called the batch-adaptive filtering[18]. In this task, during testing, if a document is retrieved by the classifier, the relevance judgement is fed back to the classifier. This feedback can be used to improve the classifier.

2.1 Corpora and Data

For text classification, we use the ModApte version of the Reuters-21578 corpus[1], where unlabelled documents are removed. This version has 9,603 training documents and 3,299 test documents. Following [7, 27], only categories that have at least one document in the training and test set are retained. This reduces the number of categories to 90.

For batch-adaptive filtering, we attempt the task of TREC-9[18], where the OHSUMED collection[6] is used. We will evaluate on the OHSU topic-set, which consists of 63 topics. The training and test material consist of 54,710 and 293,856 documents respectively. In addition, there is a topic statement for each topic. For our purpose, this is treated as an additional training document for that topic. We will only use the title, abstract, author, and source sections of the documents for training and testing.

2.2 Representation

There are various ways to transform a document into a representation convenient for classification. We will use the

SIGIR'02, August 11-15, 2002, Tampere, Finland.
Copyright 2002 ACM 1-58113-561-0/02/0008 ...$5.00.

[1] Available via `http://www.daviddlewis.com/resources/testcollections/reuters21578`.

bag-of-words approach, where we only retain frequencies of words after tokenisation, stemming, and stop-words removal. These frequencies can be normalized using various schemes[19, 6]; we use the *ltc* normalization:

$$\begin{aligned} l_{i,d} &= 1 + \log_2 TF_{i,d} \\ t_i &= \log_2 \frac{N}{n_i} \\ ltc_{i,d} &= \frac{l_{i,d} \cdot t_i}{\sqrt{\sum_{j \in \{\text{terms in } d\}} \left(l_{j,d} \cdot t_j\right)^2}}, \end{aligned}$$

where the subscripts i and d denote the ith term and the dth document respectively, $TF_{i,d}$ is the frequency of the ith term in the dth document, n_i is the document-frequency of the ith term, and N is the total number of documents.

2.3 Feature Selection Metric

Given a set of candidate terms, we select features from the set using the *likelihood ratio* for *binomial distribution* advocated by Dunning[5]:

$$\lambda = \frac{\left(\frac{R_t+R_{\bar{t}}}{N}\right)^{R_t+R_{\bar{t}}} \left(\frac{N_t+N_{\bar{t}}}{N}\right)^{N_t+N_{\bar{t}}}}{\left(\frac{R_t}{R_t+N_t}\right)^{R_t} \left(\frac{N_t}{R_t+N_t}\right)^{N_t} \left(\frac{R_{\bar{t}}}{R_{\bar{t}}+N_{\bar{t}}}\right)^{R_{\bar{t}}} \left(\frac{N_{\bar{t}}}{R_{\bar{t}}+N_{\bar{t}}}\right)^{N_{\bar{t}}}},$$

where R_t (N_t) is the number of relevant (non-relevant) training documents which contain the term, $R_{\bar{t}}$ ($N_{\bar{t}}$) is the number of relevant (non-relevant) training documents which do not, and N is the total number of training documents.

Asymptotically, $-2\ln\lambda$ is χ^2 distributed with 1 degree of freedom. We choose terms with $-2\ln\lambda$ more than 12.13, i.e. at 0.05% significance level. More details on the feature selection procedures will be given in section 4.

2.4 Performance Measures

To evaluate a text classification system, we use the F_1 measure introduced by van Rijsbergen[22]. This measure combines recall and precision in the following way:

$$\begin{aligned} \text{Recall} &= \frac{\text{number of correct positive predictions}}{\text{number of positive examples}} \\ \text{Precision} &= \frac{\text{number of correct positive predictions}}{\text{number of positive predictions}} \\ F_1 &= \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}. \end{aligned}$$

For ease of comparison, we summarize the F_1 scores over the different categories using the micro- and macro-averages of F_1 scores[11, 27]:

$$\begin{aligned} \text{Micro-avg } F_1 &= F_1 \text{ over categories and documents} \\ \text{Macro-avg } F_1 &= \text{average of within-category } F_1 \text{ values.} \end{aligned}$$

The micro- and macro-average F_1 emphasize the performance of the system on common and rare categories respectively. Using these averages, we can observe the effect of different kinds of data on a text classification system.

In addition, for comparing two text classification systems, we use the *micro sign-test* (s-test) and the *macro sign-test* (S-test), which are two significance tests first used for comparing text classification systems in [27]. The s-test compares all the binary decisions made by the systems, while the S-test compares the within-category F_1 values. Similar to the F_1 averages, the s-test and S-test compare the performance of two systems on common and rare categories respectively.

To evaluate a batch-adaptive filtering system, we use the `T9P` measure of TREC-9[18]:

$$\texttt{T9P} = \frac{\text{number of correct positive predictions}}{\text{Max(50, number of positive predictions)}},$$

which is precision, with a penalty for not retrieving 50 documents.

3. BAYESIAN ONLINE LEARNING

Most of this section is based on work by Opper[17], Solla & Winther[20], and Csató & Opper[3].

Suppose that each document is described by a vector $\mathbf{x}$, and that the relevance indicator of $\mathbf{x}$ for a category is given by label $y \in \{-1, 1\}$, where -1 and 1 indicates irrelevant and relevant respectively. Given m instances of past data $\mathcal{D}_m = \{(y_t, \mathbf{x}_t), t = 1...m\}$, the *predictive probability* of the relevance of a document described by $\mathbf{x}$ is

$$p(y|\mathbf{x}, \mathcal{D}_m) = \int d\mathbf{a}\; p(y|\mathbf{x}, \mathbf{a}) p(\mathbf{a}|\mathcal{D}_m),$$

where we have introduced the classifier $\mathbf{a}$ to assist us in the prediction. In the Bayesian approach, $\mathbf{a}$ is a random variable with probability density $p(\mathbf{a}|\mathcal{D}_m)$, and we integrate over all the possible values of $\mathbf{a}$ to obtain the prediction.

Our aim is to obtain a reasonable description of $\mathbf{a}$. In the Bayesian online learning framework[16, 20, 17], we begin with a *prior* $p(\mathbf{a}|\mathcal{D}_0)$, and perform incremental Bayes' updates to obtain the *posterior* as data arrives:

$$p(\mathbf{a}|\mathcal{D}_{t+1}) \quad = \quad \frac{p(y_{t+1}|\mathbf{x}_{t+1}, \mathbf{a}) p(\mathbf{a}|D_t)}{\int d\mathbf{a}\; p(y_{t+1}|\mathbf{x}_{t+1}, \mathbf{a}) p(\mathbf{a}|D_t)}.$$

To make the learning online, the explicit dependence of the posterior $p(\mathbf{a}|\mathcal{D}_{t+1})$ on the past data is removed by approximating it with a distribution $p(\mathbf{a}|A_{t+1})$, where A_{t+1} characterizes the distribution of $\mathbf{a}$ at time $t+1$. For example, if $p(\mathbf{a}|A_{t+1})$ is a Gaussian, then A_{t+1} refers to its mean and covariance.

Hence, starting from the *prior* $p_0(\mathbf{a}) = p(\mathbf{a}|A_0)$, learning from a new example $(y_{t+1}, \mathbf{x}_{t+1})$ comprises two steps:

Update the posterior using Bayes rule

$$p(\mathbf{a}|A_t, (y_{t+1}, \mathbf{x}_{t+1})) \propto p(y_{t+1}|\mathbf{x}_{t+1}, \mathbf{a})\; p(\mathbf{a}|A_t)$$

Approximate the updated posterior by parameterisation

$$p(\mathbf{a}|A_t, (y_{t+1}, \mathbf{x}_{t+1})) \to p(\mathbf{a}|A_{t+1}),$$

where the approximation step is done by minimizing the *Kullback-Leibler distance* between the the approximating and approximated distributions.

The amount of information gained about $\mathbf{a}$ after learning from a new example can be expressed as the Kullback-Leibler distance between the posterior and prior distributions[25]:

$$\begin{aligned} IG(y_{t+1}, \mathbf{x}_{t+1}|\mathcal{D}_t) &= \int d\mathbf{a}\; p(\mathbf{a}|\mathcal{D}_{t+1}) \log_2 \frac{p(\mathbf{a}|\mathcal{D}_{t+1})}{p(\mathbf{a}|\mathcal{D}_t)} \\ &\approx \int d\mathbf{a}\; p(\mathbf{a}|A_{t+1}) \log_2 \frac{p(\mathbf{a}|A_{t+1})}{p(\mathbf{a}|A_t)}, \end{aligned}$$

where instances of the data $\mathcal{D}$ are replaced by the summaries A in the approximation.

To simplify notation henceforth, we use $p_t(\mathbf{a})$ and $\langle \ldots \rangle_t$ to denote $p(\mathbf{a}|A_t)$ and averages taken over $p(\mathbf{a}|A_t)$ respectively. For example, the predictive probability can be rewritten as

$$p(y|\mathbf{x}, \mathcal{D}_t) \approx p(y|\mathbf{x}, A_t) = \int d\mathbf{a}\, p(y|\mathbf{x}, \mathbf{a}) p_t(\mathbf{a}) = \langle p(y|\mathbf{x}, \mathbf{a}) \rangle_t\,.$$

In the following sections, the scalar field $h = \mathbf{a} \cdot \mathbf{x}$ will also be used to simplify notation and calculation.

3.1 Bayesian Online Perceptron

Consider the case where $\mathbf{a}$ describes a *perceptron*. We then define the *likelihood* as a *probit* model

$$p(y|\mathbf{x}, \mathbf{a}) = \Phi\left(\frac{y\mathbf{a} \cdot \mathbf{x}}{\sigma_0}\right),$$

where σ_0^2 is a fixed noise variance, and Φ is the cumulative Gaussian distribution

$$\Phi(u) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{u} d\xi\; e^{-\xi^2/2}.$$

If $p_0(\mathbf{a})$ is the spherical unit Gaussian, and $p_t(\mathbf{a})$ is the Gaussian approximation, Opper[16, 17] and Solla & Winther[20] obtain the following updates by equating the means and covariances of $p(\mathbf{a}|A_{t+1})$ and $p(\mathbf{a}|A_t, (y_{t+1}, \mathbf{x}_{t+1}))$:

$$\begin{aligned} \langle \mathbf{a} \rangle_{t+1} &= \langle \mathbf{a} \rangle_t + \mathbf{s}_{t+1} \frac{\partial}{\partial \langle h \rangle_t} \ln \langle p(y_{t+1}|h) \rangle_t \\ \mathbf{C}_{t+1} &= \mathbf{C}_t + \mathbf{s}_{t+1} \mathbf{s}_{t+1}^{\mathrm{T}} \frac{\partial^2}{\partial \langle h \rangle_t^2} \ln \langle p(y_{t+1}|h) \rangle_t\,, \end{aligned}$$

where

$$\begin{aligned} \mathbf{s}_{t+1} &= \mathbf{C}_t \mathbf{x}_{t+1}, \\ \langle p(y_{t+1}|h) \rangle_t &= \Phi\left(\frac{y_{t+1} \langle h \rangle_t}{\sigma_{t+1}}\right), \\ \sigma_{t+1}^2 &= \sigma_0^2 + \mathbf{x}_{t+1}^T \mathbf{C}_t \mathbf{x}_{t+1} \quad \text{and} \\ \langle h \rangle_t &= \langle \mathbf{a} \rangle_t^{\mathrm{T}} \mathbf{x}_{t+1}. \end{aligned}$$

3.1.1 Algorithm

Training the Bayesian online perceptron on m data involves successive calculation of the means $\langle \mathbf{a} \rangle_t$ and covariances $\mathbf{C}_t$ of the posteriors, for $t \in \{1, ..., m\}$:

1. Initialize $\langle \mathbf{a} \rangle_0$ to be $\mathbf{0}$ and $\mathbf{C}_0$ to be $\mathbf{1}$ (identity matrix), i.e. a spherical unit Gaussian centred at origin.
2. For $t = 0, 1, ..., m-1$
3. y_{t+1} is the relevance indicator for document $\mathbf{x}_{t+1}$
4. Calculate $\mathbf{s}_{t+1}$, σ_{t+1}, $\langle h \rangle_t$ and $\langle p(y_{t+1}|h) \rangle_t$
5. Calculate $u = \frac{y_{t+1} \langle h \rangle_t}{\sigma_{t+1}}$ and $\Box = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}u^2)$
6. Calculate $\frac{\partial}{\partial \langle h \rangle_t} \ln \langle p(y_{t+1}|h) \rangle_t = \frac{y_{t+1}}{\sigma_{t+1}} \cdot \frac{1}{\langle p(y_{t+1}|h) \rangle_t} \cdot \Box$
7. Calculate $\frac{\partial^2}{\partial \langle h \rangle_t^2} \ln \langle p(y_{t+1}|h) \rangle_t =$
$$-\frac{1}{\sigma_{t+1}^2} \left[\frac{u \cdot \Box}{\langle p(y_{t+1}|h) \rangle_t} + \left(\frac{\Box}{\langle p(y_{t+1}|h) \rangle_t} \right)^2 \right]$$
8. Calculate $\langle \mathbf{a} \rangle_{t+1}$ and $\mathbf{C}_{t+1}$

The prediction for datum $(y, \mathbf{x})$ simply involves the calculation of $\langle p(y|\mathbf{x}, \mathbf{a}) \rangle_m = \langle p(y|h) \rangle_m$.

3.2 Bayesian Online Gaussian Process

Gaussian process (GP) has been constrained to problems with small data sets until recently when Csató & Opper[3] and Williams & Seeger[24] introduced efficient and effective approximations to the full GP formulation. This section will outline the approach in [3].

In the GP framework, $\mathbf{a}$ describes a function consisting of function values $\{a(\mathbf{x})\}$. Using the probit model, the likelihood can be expressed as

$$p(y|\mathbf{x}, \mathbf{a}) = \Phi\left(\frac{y a(\mathbf{x})}{\sigma_0}\right),$$

where σ_0 and Φ are described in section 3.1.

In addition, $p_0(\mathbf{a})$ is a GP prior which specifies a Gaussian distribution with zero mean function and *covariance/kernel* function $K_0(\mathbf{x}, \mathbf{x}')$ over a function space. If $p_t(\mathbf{a})$ is also a Gaussian process, then Csató & Opper obtain the following updates by equating the means and covariances of $p(\mathbf{a}|A_{t+1})$ and $p(\mathbf{a}|A_t, (y_{t+1}, \mathbf{x}_{t+1}))$:

$$\begin{aligned} \langle \mathbf{a} \rangle_{t+1} &= \langle \mathbf{a} \rangle_t + \mathbf{s}_{t+1} \frac{\partial}{\partial \langle h \rangle_t} \ln \langle p(y_{t+1}|h) \rangle_t \\ \mathbf{C}_{t+1} &= \mathbf{C}_t + \mathbf{s}_{t+1} \mathbf{s}_{t+1}^{\mathrm{T}} \frac{\partial^2}{\partial \langle h \rangle_t^2} \ln \langle p(y_{t+1}|h) \rangle_t\,, \end{aligned}$$

where

$$\begin{aligned} \mathbf{s}_{t+1} &= \mathbf{C}_t \mathbf{k}_{t+1} + \mathbf{e}_{t+1}, \\ \langle p(y_{t+1}|h) \rangle_t &= \Phi\left(\frac{y_{t+1} \langle h \rangle_t}{\sigma_{t+1}}\right), \\ \sigma_{t+1}^2 &= \sigma_0^2 + k_{t+1}^* + \mathbf{k}_{t+1}^T \mathbf{C}_t \mathbf{k}_{t+1} \quad \text{and} \\ \langle h \rangle_t &= \langle a(\mathbf{x}_{t+1}) \rangle_t \;=\; \langle \mathbf{a} \rangle_t^{\mathrm{T}} \mathbf{k}_{t+1} \end{aligned}$$

Notice the similarities to the updates in section 3.1. The main difference is the 'kernel trick' introduced into the equations through

$$\begin{aligned} k_{t+1}^* &= K_0(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) \quad \text{and} \\ \mathbf{k}_{t+1} &= (K_0(\mathbf{x}_1, \mathbf{x}_{t+1}), \ldots, K_0(\mathbf{x}_t, \mathbf{x}_{t+1}))^T \end{aligned}$$

New inputs $\mathbf{x}_{t+1}$ are added sequentially to the system via the $(t+1)$th unit vector $\mathbf{e}_{t+1}$. This results in a quadratic increase in matrix size, and is a drawback for large data sets, such as those for text classification. Csató & Opper overcome this by introducing sparseness into the GP. The idea is to replace $\mathbf{e}_{t+1}$ by the projection

$$\hat{\mathbf{e}}_{t+1} = \mathbf{K}_t^{-1} \mathbf{k}_{t+1},$$

where

$$\mathbf{K}_t = \{K_0(\mathbf{x}_i, \mathbf{x}_j), i, j = 1 \ldots t\}.$$

This approximation introduces an error

$$\epsilon_{t+1} = (k_{t+1}^* - \mathbf{k}_{t+1}^{\mathrm{T}} \mathbf{K}_t^{-1} \mathbf{k}_{t+1}) \frac{\partial}{\partial \langle h \rangle_t} \ln \langle p(y_{t+1}|h) \rangle_t\,,$$

which is used to decide when to employ the approximation.

Hence, at any time the algorithm holds a set of *basis vectors*. It is usually desirable to limit the size of this set. To accommodate this, Csató & Opper describe a procedure for removing a basis vector from the set by reversing the process of adding new inputs.

For lack of space, the algorithm for the Bayesian Online Gaussian Process will not be given here. The reader is referred to [3] for more information.

4. EVALUATION

4.1 Classification on Reuters-21578

In this evaluation, we will compare Bayesian online perceptron, Bayesian online Gaussian process, and Support Vector Machines (SVM)[23]. SVM is one of the best performing learning algorithms on the Reuters-21578 corpus[7, 27].

The Bayesian methods are as described in section 3, while for SVM we will use the SVM^{light} package by Joachims[8]. Since SVM is a batch method, to have a fair comparison, the online methods are iterated through the training data 3 times before testing.[2]

4.1.1 Feature Selection

For the Reuters-21578 corpus, we select as features for each category the set of all words for which $-2 \ln \lambda > 12.13$. We further prune these by using only the top 300 features. This reduces the computation time required for the calculation of the covariances of the Bayesian classifiers.

Since SVM is known to perform well for many features, for the SVM classifiers we also use the set of words which occur in at least 3 training documents[7]. This gives us 8,362 words. Note that these words are non-category specific.

4.1.2 Thresholding

The probabilistic outputs from the Bayesian classifiers can be used in various ways. The most direct way is to use the *Bayes decision rule*, $p(y = 1|\mathbf{x}, \mathcal{D}_m) > 0.5$, to determine the relevance of the document described by $\mathbf{x}$.[3] However, as discussed in [10, 26], this is not optimal for the chosen evaluation measure.

Therefore, in addition to 0.5 thresholding, we also empirically optimise the threshold for each category for the F_1 measure on the training documents. This scheme, which we shall call `MaxF1`, has also been employed in [27] for thresholding kNN and LLSF classifiers. The difference from our approach is that the threshold in [27] is calculated over a validation set. We do not use a validation set because we feel that, for very rare categories, it is hard to obtain a reasonable validation set from the training documents.

For the Bayesian classifiers, we also perform an analytical threshold optimisation suggested by Lewis[10]. In this scheme, which we shall call `ExpectedF1`, the threshold for each category is selected to optimise the expected F_1:

$$\mathrm{E}\left[F_1\right]_\theta \approx \begin{cases} \prod_{i \in \mathcal{D}} (1 - p_i) & \text{if } |\mathcal{D}_+| = 0 \\ \frac{2 \sum_{i \in \mathcal{D}_+} p_i}{|\mathcal{D}_+| + \sum_{i \in \mathcal{D}} p_i} & \text{otherwise,} \end{cases}$$

where θ is the threshold, p_i is the probability assigned to document i by the classifier, $\mathcal{D}$ is the set of all test documents, and $\mathcal{D}_+$ is the set of test documents with probabilities higher than the threshold θ.

Note that `ExpectedF1` can only be applied after the probabilities for all the test documents are assigned. Hence the classification can only be done in batch. This is unlike the first two schemes, where classification can be done online.

4.1.3 Results and Discussion

Table 1: Description of Methods

	Description[4]
SVM-1	$K_0 = \mathbf{x}_i \cdot \mathbf{x}_j + 1$
SVM-2	$K_0 = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2$
SVM-R1	$K_0 = \exp(-\frac{1}{2}\lvert\mathbf{x}_i - \mathbf{x}_j\rvert^2)$
Perceptron	$\sigma_0 = 0.5$, one fixed feature (for bias)
GP-1	$\sigma_0 = 0.5,\ K_0 = \mathbf{x}_i \cdot \mathbf{x}_j + 1 + 10^{-4}\delta_{ij}$
GP-2	$\sigma_0 = 0.5,\ K_0 = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2 + 10^{-4}\delta_{ij}$
GP-R1	$\sigma_0 = 0.5,\ K_0 = \exp(-\frac{1}{2}\lvert\mathbf{x}_i - \mathbf{x}_j\rvert^2) + 10^{-4}\delta_{ij}$

Table 2: Micro-/Macro-average F_1

	`0.5`	`MaxF1`	`ExpectedF1`
SVM$_a$-1	86.15 / 42.63	86.35 / 56.92	
SVM$_a$-2	85.44 / 40.13	86.19 / 56.42	
SVM$_a$-R1	84.99 / 37.61	86.63 / 53.14	
SVM$_b$-1	85.60 / 52.03	85.05 / 52.43	
SVM$_b$-2	85.60 / 50.53	84.50 / 50.49	
SVM$_b$-R1	85.75 / 50.52	84.65 / 51.27	
Perceptron	85.12 / 45.23	86.69 / 52.16	86.44 / 53.08
GP-1	85.08 / 45.20	86.73 / 52.12	86.54 / 53.12
GP-2	85.58 / 47.90	86.60 / 52.19	86.77 / 55.04
GP-R1	85.18 / 44.88	86.76 / 52.61	86.93 / 53.35

Table 1 lists the parameters for the algorithms used in our evaluation, while Table 2 and 3 tabulate the results. There are two sets of results for SVM, and they are labeled SVM$_a$ and SVM$_b$. The latter uses the same set of features as the Bayesian classifiers (i.e. using the $-2 \ln \lambda$ measure), while the former uses the set of 8,362 words as features.

Table 2 summarizes the results using F_1 averages. Table 3 compares the classifiers using s-test and S-test. Here, the `MaxF1` thresholds are used for the classification decisions. Each row in these tables compares the method listed in the first column with the other methods. The significance levels from [27] are used.

Several observations can be made:

- Generally, `MaxF1` thresholding increases the performance of all the systems, especially for rare categories.
- For the Bayesian classifiers, `ExpectedF1` thresholding improves the performance of the systems on rare categories.
- Perceptron implicitly implements the kernel used by GP-1, hence their similar results.
- With `MaxF1` thresholding, feature selection impedes the performance of SVM.
- In Table 2, SVM with 8,362 features have slightly lower micro-average F_1 to the Bayesian classifiers. However, the s-tests in Table 3 show that Bayesian classifiers outperform SVM for significantly many common categories. Hence, in addition to computing average F_1 measures, it is useful to perform sign tests.
- As shown in Table 3, for limited features, Bayesian classifiers outperform SVM for both common and rare categories.
- Based on the sign tests, the Bayesian classifiers outperform SVM (using 8,362 words) for common categories, and vice versa for rare categories.

[2] See section A.2 for discussion on the number of passes.

[3] For SVM, to minimise structural risks, we would classify the document as relevant if $\mathbf{w} \cdot \mathbf{x} + b > 0$, where $\mathbf{w}$ is the hyperplane, and b is the bias.

[4] See section A.3 for discussion on the jitter terms δ_{ij}.

Table 3: s-test/S-test using `MaxF1` thresholding

	SVM_a-1	SVM_a-2	SVM_a-R1	SVM_b-1	SVM_b-2	SVM_b-R1	Pptron	GP-1	GP-2	GP-R1
SVM_a-1		$\sim / \sim$	$< / \sim$	$\gg / \gg$	$\gg / \gg$	$\gg / \gg$	$\ll / \gg$	$\ll / \gg$	$\ll / \gg$	$\ll / \gg$
SVM_a-2	$\sim / \sim$		$\ll / \sim$	$> / \gg$	$\gg / \gg$	$\gg / \gg$	$\ll / \gg$	$\ll / \gg$	$\ll / \sim$	$\ll / >$
SVM_a-R1	$> / \sim$	$\gg / \sim$		$\gg / \sim$	$\gg / \gg$	$\gg / \gg$	$< / >$	$< / >$	$\sim / \sim$	$< / \sim$
SVM_b-1	$\ll / \ll$	$< / \ll$	$\ll / \sim$		$\gg / \sim$	$> / >$	$\ll / <$	$\ll / \sim$	$\ll / <$	$\ll / <$
SVM_b-2	$\ll / \ll$	$\ll / \ll$	$\ll / \ll$	$\ll / \sim$		$\sim / \sim$	$\ll / <$	$\ll / <$	$\ll / \ll$	$\ll / \ll$
SVM_b-R1	$\ll / \ll$	$\ll / \ll$	$\ll / \ll$	$< / <$	$\sim / \sim$		$\ll / <$	$\ll / \ll$	$\ll / \ll$	$\ll / \ll$
Perceptron	$\gg / \ll$	$\gg / \ll$	$> / <$	$\gg / >$	$\gg / >$	$\gg / >$		$\sim / \sim$	$\sim / \sim$	$\sim / \sim$
GP-1	$\gg / \ll$	$\gg / \ll$	$> / <$	$\gg / \sim$	$\gg / >$	$\gg / \gg$	$\sim / \sim$		$\sim / \sim$	$\sim / \sim$
GP-2	$\gg / \ll$	$\gg / \sim$	$\sim / \sim$	$\gg / >$	$\gg / \gg$	$\gg / \gg$	$\sim / \sim$	$\sim / \sim$		$\sim / \sim$
GP-R1	$\gg / \ll$	$\gg / <$	$> / \sim$	$\gg / >$	$\gg / \gg$	$\gg / \gg$	$\sim / \sim$	$\sim / \sim$	$\sim / \sim$	

"$\gg$" or "$\ll$" means P-value ≤ 0.01; "$>$" or "$<$" means $0.01 <$ P-value ≤ 0.05; "$\sim$" means P-value > 0.05.

The last observation suggests that one can use Bayesian classifiers for common categories, and SVM for rare ones.

4.2 Filtering on OHSUMED

In this section, only the Bayesian online perceptron will be considered. In order to avoid numerical integration of the information gain measure, instead of the probit model of section 3.1, here we use a simpler likelihood model in which the outputs are flipped with fixed probability κ:

$$p(y|\mathbf{x}, \mathbf{a}) = \kappa + (1 - 2\kappa)\Theta\left(y\mathbf{a} \cdot \mathbf{x}\right),$$

where

$$\Theta(x) = \begin{cases} 1 & x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

The update equations will also change accordingly, e.g.

$$\begin{aligned} \left\langle p(y_{t+1}|h)\right\rangle_t &= \kappa + (1 - 2\kappa)\Phi\left(\frac{y_{t+1}\left\langle h\right\rangle_t}{\sigma_{t+1}}\right), \\ \sigma_{t+1}^2 &= \mathbf{x}_{t+1}^T \mathbf{C}_t \mathbf{x}_{t+1} \quad \text{and} \\ \left\langle h\right\rangle_t &= \left\langle \mathbf{a}\right\rangle_t^{\mathrm{T}} \mathbf{x}_{t+1}. \end{aligned}$$

Using this likelihood measure, we can express the information gained from datum $(y_{t+1}, \mathbf{x}_{t+1})$ as

$$IG(y_{t+1}, \mathbf{x}_{t+1}|\mathcal{D}_t) \approx$$
$$\log_2 \kappa + \Phi\left(\frac{y_{t+1}\widehat{\langle h\rangle}_{t+1}}{\widehat{\sigma}_{t+1}}\right)\log_2\left(\frac{1-\kappa}{\kappa}\right) - \log_2\left\langle p(y_{t+1}|h)\right\rangle_t,$$

where

$$\begin{aligned} \widehat{\sigma}_{t+1}^2 &= \mathbf{x}_{t+1}^T \mathbf{C}_{t+1} \mathbf{x}_{t+1} \quad \text{and} \\ \widehat{\langle h\rangle}_{t+1} &= \left\langle \mathbf{a}\right\rangle_{t+1}^{\mathrm{T}} \mathbf{x}_{t+1}. \end{aligned}$$

We use $\kappa = 0.1$ in this evaluation. The following sections will describe the algorithm in detail. To simplify presentation, we will divide the batch-adaptive filtering task into batch and adaptive phases.

4.2.1 Feature Selection and Adaptation

During the batch phase, words for which $-2\ln\lambda > 12.13$ are selected as features.

During the adaptive phase, when we obtain a feedback, we update the features by adding any new words with $-2\ln\lambda > 12.13$. When a feature is added, the distribution of the perceptron $\mathbf{a}$ is extended by one dimension:

$$\langle \mathbf{a}\rangle \to \left(\begin{array}{c} \langle \mathbf{a}\rangle \\ \hdashline 0 \end{array}\right) \qquad \mathbf{C} \to \left(\begin{array}{c:c} \mathbf{C} & 0 \\ \hdashline 0 & 1 \end{array}\right).$$

4.2.2 Training the classifier

During the batch phase, the classifier is iterated through the training documents 3 times. In addition, the relevant documents are collected for use during the adaptive phase.

During the adaptive phase, retrieved relevant documents are added to this collection. When a document is retrieved, the classifier is trained on that document and its given relevance judgement.

The classifier will be trained on irrelevant documents most of the time. To prevent it from "forgetting" relevant documents due to its limited capacity, whenever we train on an irrelevant document, we would also train on a past relevant document. This past relevant document is chosen successively from the collection of relevant documents.

This is needed also because new features might have been added since a relevant document was last trained on. Hence the classifier would be able to gather new information from the same document again due to the additional features.

Note that the past relevant document does not need to be chosen in successive order. Instead, it can be chosen using a probability distribution over the collection. This will be desirable when handling topic-drifts.

We will evaluate the effectiveness of this strategy of retraining on past retrieved relevant documents, and denote its use by `+rel`. Though its use means that the algorithm is no longer online, asymptotic efficiency is unaffected, since only one past document is used for training at any instance.

4.2.3 Information Gain

During testing, there are two reasons why we retrieve a document. The first is that it is relevant, i.e. $p(y = 1|\mathbf{x}, \mathcal{D}_t) > 0.5$, where $\mathbf{x}$ represents the document. The second is that, although the document is deemed irrelevant by the classifier, the classifier would gain useful information from the document. Using the measure $IG(y, \mathbf{x}|\mathcal{D}_t)$, we calculate the expected information gain

$$\langle IG(\mathbf{x}|\mathcal{D}_t)\rangle = \sum_{\upsilon \in \{-1,1\}} p(y = \upsilon|\mathbf{x}, \mathcal{D}_t) \cdot IG(y = \upsilon, \mathbf{x}|\mathcal{D}_t).$$

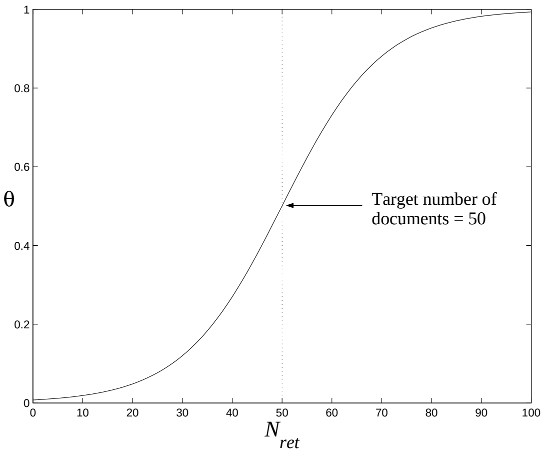

Figure 1: θ versus N_{ret} tuned for T9P

A document is then deemed useful if its expected information gain is at least θ. Optimizing for the T9P measure (i.e. targeting 50 documents), we choose θ to be

$$\theta = 0.999 \left(1 + \exp\left(-\frac{N_{\mathrm{ret}} - 50.0}{10}\right)\right)^{-1} + 0.001,$$

where N_{ret} is the total number of documents that the system has retrieved. Figure 1 plots θ against N_{ret}. Note that this is a kind of *active learning*, where the willingness to tradeoff precision for learning decreases with N_{ret}. The use of this information gain criteria will be denoted by `+ig`.

We will test the effectiveness of the information gain strategy, against an alternative one. The alternative, denoted by `+rnd`, will randomly select documents to retrieve based on the probability

$$U = \begin{cases} 0 & \text{if } N_{\mathrm{ret}} >= 50 \\ \frac{50 - N_{\mathrm{ret}}}{293856} & \text{otherwise,} \end{cases}$$

where 293,856 is the number of test documents.

4.2.4 Results and Discussion

Table 4 lists the results of seven systems. The first two are of Microsoft Research Cambridge and Fudan University respectively. These are the only runs in TREC-9 for the task. The third is of the system as described in full, i.e. Bayesian online perceptron, with retraining on past retrieved relevant documents, and with the use of information gain. The rest are of the Bayesian online perceptron with different combinations of strategies.

Besides the T9P measure, for the sake of completeness, Table 4 also lists the other measures used in TREC-9. Taken together, the measures show that Bayesian online perceptron, together with the consideration for information gain, is a very competitive method.

For the systems with `+rel`, the collection of past known relevant documents is kept. Although Microsoft uses this same collection for its query reformulation, another collection of all previously seen documents is used for threshold adaptation. Fudan maintains a collection of past retrieved documents and uses this collection for query adaptation.

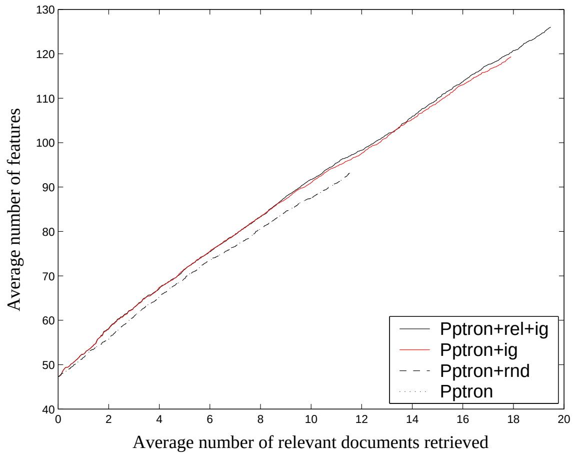

Figure 2: Variation of the number of features as relevant documents are retrieved. The plots for Pptron+rel+ig and Pptron+ig are very close. So are the plots for Pptron+rnd and Pptron.

In a typical operational system, retrieved relevant documents are usually retained, while irrelevant documents are usually discarded. Therefore `+rel` is a practical strategy to adopt.

Figure 2 plots the average number of features during the adaptive phase. We can see that features are constantly added as relevant documents are seen. When the classifier is retrained on past documents, the new features enable the classifier to gain new information from these documents. If we compare the results for `Pptron+rel` and `Pptron` in Table 4, we find that not training on past documents causes the number of relevant documents retrieved to drop by 5%. Similarly, for `Pptron+rel+ig` and `Pptron+ig`, the drop is 8%.

Table 5 breaks down the retrieved documents into those that the classifier deems relevant and those that the classifier is actually *querying* for information, for `Pptron+ig` and `Pptron+rnd`. The table shows that none of the documents randomly queried are relevant documents. This is not surprising, since only an average of 0.017% of the test documents are relevant. In contrast, the information gain strategy is able to retrieve 313 relevant documents, which is 26.1% of the documents queried. This is a significant result.

Consider `Pptron+ig`. Table 4 shows that for `Pptron`, when the information gain strategy is removed, only 731 relevant documents will be retrieved. Hence, although most of the documents queried are irrelevant, information gained from these queries helps recall by the classifier (i.e. 815 documents versus 731 documents), which is important for reaching the target of 50 documents.

MacKay[13] has noted the phenomenon of querying for irrelevant documents which are at the edges of the input space, and suggested maximizing information in a defined region of interest instead. Finding this region for batch-adaptive filtering remains a subject for further research.

Comparing the four plots in Figure 2, we find that, on average, the information gain strategy causes about 3% more features to be discovered for the same number of relevant documents retrieved. A consequence of this is better recall.

[5] [18] reports results from run ok9bfr2po, while we report results from the slightly better run ok9bf2po.

Table 4: Results for Batch-adaptive filtering optimized for `T9P` measure.

	Microsoft[5]	Fudan	`Pptron+rel+ig`	`Pptron+ig`	`Pptron+rnd`	`Pptron+rel`	`Pptron`
Total retrieved	3562	3251	2716	2391	2533	1157	1057
Relevant retrieved	1095	1061	1227	1128	732	772	731
Macro-average recall	39.5	37.9	36.2	33.3	20.0	20.8	20.0
Macro-average precision	30.5	32.2	35.8	35.8	21.6	61.9	62.3
Mean T9P	**30.5**	**31.7**	**31.3**	**29.8**	**19.2**	**21.5**	**20.8**
Mean Utility	-4.397	-1.079	15.318	15.762	-5.349	18.397	17.730
Mean T9U	-4.397	-1.079	15.318	15.762	-5.349	18.397	17.730
Mean scaled utility	-0.596	-0.461	-0.025	0.016	-0.397	0.141	0.138
Zero returns	0	0	0	0	0	8	0

Table 5: Breakdown of documents retrieved for `Pptron+ig` and `Pptron+rnd`. The numbers for the latter are in brackets.

	Relevant		Not Relevant		Total	
# docs retrieved by perceptron classifier proper	815	(732)	378	(345)	1193	(1077)
# docs retrieved by information gain (or random strategy)	313	(0)	885	(1456)	1198	(1456)
Total	1128	(732)	1263	(1801)	2391	(2533)

5. CONCLUSIONS AND FURTHER WORK

We have implemented and tested Bayesian online perceptron and Gaussian processes on the text classification problem, and have shown that their performance is comparable to that of SVM, one of the best learning algorithms on text classification in the published literature. We have also demonstrated the effectiveness of online learning with information gain on the TREC-9 batch-adaptive filtering task.

Our results on text classification suggest that one can use Bayesian classifiers for common categories, and maximum margin classifiers for rare categories. The partitioning of the categories into common and rare ones in an optimal way is an interesting problem.

SVM has been employed to use relevance feedback by Drucker *et al*[4], where the retrieval is in groups of 10 documents. In essence, this is a form of adaptive routing. It would be instructive to see how Bayesian classifiers perform here, without storing too many previously seen documents.

It would also be interesting to compare the merits of incremental SVM[21, 1] with the Bayesian online classifiers.

Acknowledgments

We would like to thank Lehel Csató for providing details on the implementation of the Gaussian process, Wee Meng Soon for assisting in the data preparation, Yiming Yang for clarifying the representation used in [27], and Loo Nin Teow for proof-reading the manuscript. We would also like to thank the reviewers for their many helpful comments in improving the paper.

6. REFERENCES

[1] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *NIPS 2000*, volume 13. The MIT Press, 2001.

[2] D. Cox and E. Snell. *Analysis of Binary Data*. Chapman & Hall, London, 2nd edition, 1989.

[3] L. Csató and M. Opper. Sparse representation for Gaussian process models. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *NIPS 2000*, volume 13. The MIT Press, 2001.

[4] H. Drucker, B. Shahrary, and D. C. Gibbon. Relevance feedback using support vector machines. In *Proceedings of the 2001 International Conference on Machine Learning*, 2001.

[5] T. E. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.

[6] W. Hersh, C. Buckley, T. Leone, and D. Hickam. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 192–201, 1994.

[7] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 137–142, 1998.

[8] T. Joachims. Making large-scale SVM learning practical. In B. Schólkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, chapter 11. The MIT Press, 1999.

[9] D. D. Lewis. *Representation and Learning in Information Retrieval.* PhD thesis, Department of Computer and Information Science, University of Massachusetts at Amherst, 1992.

[10] D. D. Lewis. Evaluating and optimizing automomous text classification systems. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 246–254, 1995.

[11] D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 298–306, 1996.

[12] D. J. Mackay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1991.

[13] D. J. Mackay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992.

[14] R. M. Neal. Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical Report CRG-TR-97-2, Department of Computer Science, University of Toronto, January 1997.

[15] H. T. Ng, W. B. Goh, and K. L. Low. Feature selection, perceptron learning, and a usability case study for text categorization. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 67–73, 1997.

[16] M. Opper. Online versus offline learning from random examples: General results. *Physical Review Letters*, 77:4671–4674, 1996.

[17] M. Opper. A Bayesian approach to online learning. In D. Saad, editor, *On-Line Learning in Neural Networks*. Combridge University Press, 1998.

[18] S. Robertson and D. A. Hull. The TREC-9 filtering track final report. In *Proceedings of the 9th Text REtrieval Conference (TREC-9)*, pages 25–40, 2001.

[19] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[20] S. A. Solla and O. Winther. Optimal perceptron learning: an online Bayesian approach. In D. Saad, editor, *On-Line Learning in Neural Networks*. Combridge University Press, 1998.

[21] N. A. Syed, H. Liu, and K. K. Sung. Incremental learning with support vector machines. In *Proceedings of the Workshop on Support Vector Machines at the International Joint Conference on Artificial Intelligence (IJCAI-99)*, 1999.

[22] C. van Rijsbergen. *Information Retrieval.* Butterworths, London, 1979.

[23] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.

[24] C. K. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *NIPS 2000*, volume 13. The MIT Press, 2001.

[25] O. Winther. *Bayesian Mean Field Algorithms for Neural Networks and Gaussian Processes.* PhD thesis, University of Copenhagen, CONNECT, The Niels Bohr Institute, 1998.

[26] Y. Yang. A study on thresholding strategies for text categorization. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 137–145, 2001.

[27] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 42–49, 1999.

APPENDIX

A. ON THE CHOICE OF PARAMETERS

A.1 Likelihood model

MacKay[12] has suggested the evidence framework for model selection. Here, we calculate the evidence on the training data using the final posterior for $\mathbf{a}$:

$$p(\mathcal{D}_m) = \prod_{t=1}^{m} \langle p(y_t|\mathbf{x}_t, \mathbf{a}) \rangle_m .$$

Table 6: Micro-/Macro-avg F_1 (`MaxF1` thresholds) and Avg log-evidence on Reuters-21578 for different likelihood models, using Bayesian online perceptron.

	Micro-/Macro-avg F_1	Avg log-evidence
Logit	86.48 / 52.75	-45.02
Probit	86.69 / 52.16	-34.32
Flip	85.94 / 53.00	-368.8

Table 7: Micro-/Macro-avg F_1 (`MaxF1` thresholds) and Avg log-evidence on Reuters-21578 for different passes over the training data, using Bayesian online perceptron.

Passes	Micro-/Macro-avg F_1	Avg log-evidence
1	87.08 / 52.87	-35.56
2	86.92 / 52.63	-34.36
3	86.69 / 52.16	-34.32
4	86.62 / 52.75	-34.54
5	85.22 / 46.93	-34.69

Table 6 illustrates this for selecting the likelihood measure for the text classification task, using the Bayesian online perceptron. In the table, the probit model follows the formulation in section 3.1 with $\sigma_0 = 0.5$, logit model is estimated by the probit model with $\sigma_0 = 1.6474$[2], and the flip noise model is as described in section 4.2. Although their F_1 averages are similar, the evidences show that the probit model with $\sigma_0 = 0.5$ is a more likely model. The small evidence for the flip noise model is because much information is lost through the threshold function Θ.

A.2 Effects of multiple passes over data

Using the evidence measure defined in section A.1, Table 7 illustrates the effects of different number of passes over training data for Bayesian online perceptron. Treating the number of passes as a parameter for the algorithm, we see that having 3 passes over the data gives the highest average evidence, although there is no significant difference between 2, 3, or 4 passes. Similar results hold for the Gaussian process for the 3 different kernels. Hence, in section 4.1, we choose to use 3 passes for all the Bayesian algorithms.

A.3 Jitter term

The addition of the jitter term $10^{-4}\delta_{ij}$ (where $\delta_{ij} = 1$ if $i = j$, and 0 otherwise) for Gaussian process for classification is recommended by Neal[14]. This term improves the conditioning of the matrix computations while having a small effect on the model. From our preliminary experiments, without the jitter term, the matrix operations in Bayesian online Gaussian process become ill-conditioned.

A.4 Sizes of the basis vectors sets

The sizes of the sets of basis vectors for GP in section 4.1 are limited to less than or equal to the number of features selected. This is because, as noted by Csató & Opper[3], for a feature space of finite dimension M, no more than M basis vectors are needed, due to linear dependence.