Context-sensitive Bayesian Classifiers and Application to Mouse Pressure Pattern Classification

Yuan Qi and Rosalind W. Picard
Media Lab, MIT, Cambridge, MA, 02139

Abstract

In this paper, we propose a new context-sensitive Bayesian learning algorithm. By modeling the distributions of data locations by a mixture of Gaussians, the new algorithm can utilize different classifier complexities for different contexts/locations and, at the same time, keep the optimality of Bayesian solutions. This algorithm is also an online learning algorithm, efficient in training, and easy for incorporating new knowledge from data sets available in the future. We apply this algorithm to detecting computer-user mouse pressure patterns during episodes likely to be frustrating to the user. By modeling user identity as hidden context, this algorithm achieves on average 10.6% *user-independent test error rate.*

1 Introduction

By approximating the Bayesian average, Bayes Classifiers achieve good generalization performance [3, 6]. A Bayesian linear classifier can be easily converted to a non-linear classifier by using feature expansions or kernel methods as does the Support Vector Machine (SVM).

On the other hand, prediction for real world applications is often complicated by some changing context. For example, a person may change his accent when speaking at the office vs. when speaking in his hometown. Modeling such a hidden context should result in better performance. In this paper, we present context-sensitive Bayesian classifiers that switch Bayesian classifiers corresponding to different contexts. We train each component classifier by an efficient training algorithm, expectation propagation [6].

Similar to classical context learning algorithms, for example, the Splice algorithm [2], the context-sensitive Bayesian classifier can be viewed as an approximation to a mixture of experts [4, 9] with an easier training procedure. Our algorithm is also an online learning technique: after training on current data sets, if there are more data sets possibly containing different contexts in the future, we only need to train some new local classifiers based on new data sets. By contrast, classical global learning techniques, e.g., SVM, will re-train a global classifier over all the data sets to get a new optimal solution. Thus, when there are mulitple data sets, our approach requires less computer memory size and has a faster training speed than classical global learning techniques.

One of the main difference between our algorithm and many other context learning algorithms, such as Splice, is that we keep the context-sensitive classifiers' optimality in the Bayesian sense by using data posterior probabilities to combine different component classifiers even after incremental training.

In a previous paper [7], we used a different Bayesian approach in an attempt to classify mouse pressure signals from subjects into two categories: pressure patterns that arise when all is going well, and those that arise following potentially frustrating events, like delays or usability bugs. The method was not context-dependent, and was run in a user-dependent way, obtaining on average 88% classification accuracy. The results in [7], plus some additional analysis from the human subjects in the experiment, confirmed that such mouse patterns do evince differences under the two conditions [8]. However, it is desirable to build one person-independent classifier. The problem is difficult in that the generalization performance of the classifier deteriorates when it is trained and tested on different computer users who have different mouse-usage behaviors. In this paper, context-sensitive Bayesian classifiers treat the identity of computer user as context information, softly switching the classifiers based on extracted context. Our experimental results demonstrate the effectiveness of this approach.

2 Context-Sensitive Bayesian Classifier

Given a training set $D = \{D_1, \ldots, D_L\}$ that has L subsets

$$D_1 = \{(\mathbf{x}_1^1, t_1^1), \ldots, (\mathbf{x}_{N_1}^1, t_{N_1}^1)\} \quad (1)$$

$$\vdots$$

$$D_L = \{(\mathbf{x}_1^L, t_1^L), \ldots, (\mathbf{x}_{N_L}^L, t_{N_L}^L)\}, \quad (2)$$

a test data point $\tilde{\mathbf{x}}$ can be classified as follows:

$$p(\tilde{t}|\tilde{\mathbf{x}}, D) = \sum_i p(\tilde{t}|\tilde{\mathbf{x}}, D_i) P(i|\tilde{\mathbf{x}}, D) \quad (3)$$

where i means choosing the subset D_i and different subsets D_i's correspond to different contexts. In the following sections, we first propose a new way to compute the classification confidence $p(\tilde{t}|\tilde{\mathbf{x}}, D_i)$ based on expectation propagation [6], and then apply a Gaussian mixture model to compute $P(\tilde{\mathbf{x}}|D_i)$. For a finite number of contexts, after computing $p(\tilde{t}|\tilde{\mathbf{x}}, D_i)$ and $P(\tilde{\mathbf{x}}|D_i)$, it is easy to obtain $p(\tilde{t}|\tilde{\mathbf{x}})$ using equation (3).

2.1 Training Bayesian Classifiers

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

$$p(\mathbf{t}|\mathbf{w}, X) = \prod_i p(t_i|\mathbf{x}_i, \mathbf{w}) = \prod_i \Psi(t_i \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_i)) \quad (4)$$

where $\mathbf{t} = \{t_i\}_{i=1}^N$, $X = \{\mathbf{x}_i\}_{i=1}^N$, and $\Psi(a)$ is a step function, i.e., $\Psi(a) = 1$ if $a > 0$ and $\Psi(a) = -1$ if $a \leq 0$. We can also use the logistic function or probit model as $\Psi(\cdot)$. In this section, we drop the data set index i in (1) to (3) for simplicity.

Given a new input $\mathbf{x_{N+1}}$, we approximate the predictive distribution:

$$p(t_{N+1}|\mathbf{x}_{N+1}, \mathbf{t}) = \int p(t_{N+1}|\mathbf{x}_{N+1}, \mathbf{w}) p(\mathbf{w}|\mathbf{t}) d\mathbf{w} \quad (5)$$

$$\approx P(t_{N+1}|\mathbf{x}_{N+1}, \langle \mathbf{w} \rangle) \quad (6)$$

where $\langle \mathbf{w} \rangle$ denotes the posterior mean of the weights, called the Bayes Point. Instead of using popular Monte-Carlo methods, we apply efficient Expectation Propagation to compute $\langle \mathbf{w} \rangle$ and $p(t_{N+1}|\mathbf{x}_{N+1}, \mathbf{t})$.

2.1.1 Expectation Propagation

Expectation Propagation (EP) [6] 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.

Expectation Propagation can also be viewed as a powerful extension of assumed-density filtering (ADF) [5]. The ADF method is a sequential technique for approximating a posterior distribution that can be used in stochastic process modeling and online learning. Expectation Propagation extends ADF by using iterative batch-version refinements; this enables EP to utilize the information from the whole data sequence and to greatly improve the approximation quality.

First, a Gaussian prior distribution is assigned for $\mathbf{w}$

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_i \mathcal{N}(w_i|0, \alpha_i^{-1}) \quad (7)$$

where $\boldsymbol{\alpha} = \{\alpha_i\}$ is a hyperparameter vector. Later, we assign $\alpha_i = 1$ for all i.

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

$$p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}) \propto p(\mathbf{w}|\boldsymbol{\alpha}) \prod_i p(t_i|\mathbf{w}) = p(\mathbf{w}|\boldsymbol{\alpha}) \prod_i g_i(\mathbf{w}) \quad (8)$$

$$\approx p(\mathbf{w}|\boldsymbol{\alpha}) \prod_i \tilde{g}_i(\mathbf{w}) \quad (9)$$

For the Bayesian linear classifier, the approximate terms are chosen to be Gaussian, parameterized by (m_i, v_i, s_i):

$$\tilde{g}_i = s_i \exp(-\frac{1}{2v_i}(t_i \boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - m_i)^2). \quad (10)$$

This makes the approximate posterior distribution also Gaussian:

$$p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}) \approx q(\mathbf{w}) = \mathcal{N}(\mathbf{m}_w, \mathbf{V}_w). \quad (11)$$

To find the best term approximations we proceed as follows: (to save notation, $t_i\boldsymbol{\phi}(\mathbf{x}_i)$ 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 the prior: $\mathbf{m}_w = \mathbf{0}$, $\mathbf{V}_w = \text{diag}(\boldsymbol{\alpha})$, $\alpha_i = 1$ for all i.

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

   $$\mathbf{V}_w^{\setminus i} = \mathbf{V}_w + \frac{(\mathbf{V}_w\boldsymbol{\phi}_i)(\mathbf{V}_w\boldsymbol{\phi}_i)^T}{v_i - \boldsymbol{\phi}_i^T\mathbf{V}_w\boldsymbol{\phi}_i} \quad (12)$$

   $$\mathbf{m}_w^{\setminus i} = \mathbf{m}_w + (\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i)v_i^{-1}(\boldsymbol{\phi}_i^T\mathbf{m}_w - m_i) \quad (13)$$

   (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 $KL(\hat{p}(\mathbf{w}) \,||\, q(\mathbf{w}))$. Let Z_i be the normalizing factor.

$$\mathbf{m}_w = \mathbf{m}_w^{\setminus i} + \mathbf{V}_w^{\setminus i}\rho_i\boldsymbol{\phi}_i \tag{14}$$

$$\mathbf{V}_w = \mathbf{V}_w^{\setminus i} - (\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i)\big(\frac{\rho_i\boldsymbol{\phi}_i^T\mathbf{m}_w}{\boldsymbol{\phi}_i^T\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i}\big)(\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i)^T \tag{15}$$

$$Z_i = \int_{\mathbf{w}} g_i(\mathbf{w})q^{\setminus i}(\mathbf{w})\mathrm{d}\mathbf{w} = \Psi(z_i) \tag{16}$$

$$\text{where } z_i = \frac{(\mathbf{m}_w^{\setminus i})^T\boldsymbol{\phi}_i}{\sqrt{\boldsymbol{\phi}_i^T\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i}} \tag{17}$$

$$\rho_i = \frac{1}{\sqrt{\boldsymbol{\phi}_i^T\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i}}\frac{\mathcal{N}(z_i;0,1)}{\Psi(z_i)} \tag{18}$$

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

$$v_i = \boldsymbol{\phi}_i^T\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i\Big(\frac{1}{\rho_i\boldsymbol{\phi}_i^T\mathbf{m}_w} - 1\Big) \tag{19}$$

$$m_i = \boldsymbol{\phi}_i^T\mathbf{m}_w^{\setminus i} + (v_i + \boldsymbol{\phi}_i^T\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i)\rho_i \tag{20}$$

$$s_i = Z_i\sqrt{1 + v_i^{-1}\boldsymbol{\phi}_i^T\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i}\exp\big(\frac{\rho_i}{2}\frac{\boldsymbol{\phi}_i^T\mathbf{V}_w^{\setminus i}\boldsymbol{\phi}_i}{\boldsymbol{\phi}_i^T\mathbf{m}_w}\big) \tag{21}$$

2.2 Computing $p(\tilde{t}|\tilde{\mathbf{x}}, D_i)$

Given the training results for data set D_i, we can compute $p(\tilde{t}|\tilde{\mathbf{x}}, D_i)$ as follows:

$$p(\tilde{t}|\tilde{\mathbf{x}}, D_i) = \int p(\tilde{t}|\tilde{\mathbf{x}}, \mathbf{w})p(\mathbf{w}|D_i)\mathrm{d}\mathbf{w} = \Psi(\tilde{z}) \tag{22}$$

$$\text{where } \tilde{z} = \frac{(\tilde{t}\mathbf{m}_w)^T\boldsymbol{\phi}(\tilde{\mathbf{x}})}{\sqrt{\boldsymbol{\phi}^T(\tilde{\mathbf{x}})\mathbf{V}_w\boldsymbol{\phi}(\tilde{\mathbf{x}})}} \tag{23}$$

By contrast, in mixture of experts [9], logistic regression is used instead of Bayesian classifiers.

2.3 Computing $P(i|\tilde{\mathbf{x}}, D)$

Denote $\mathbf{x}^i = \{\mathbf{x}_1^i, \ldots, \mathbf{x}_{N_i}^i\}$, $\mathbf{t}^i = \{\mathbf{t}_1^i, \ldots, \mathbf{t}_{N_i}^i\}$, and $D_i = \{\mathbf{x}^i, \mathbf{t}^i\}$. We model $\mathbf{x}^i$ by a Gaussian mixture model H_i, i.e.,

$$p(\mathbf{x}^i) = \sum_{j=1}^{J}\beta_j\mathcal{N}(\boldsymbol{\mu}_j, \Sigma_j).$$

where H_i is parameterized by $\{\beta_j, \boldsymbol{\mu}_j, \Sigma_j\}_{j=1}^J$. Classical expectation maximization technique can be applied to estimating the parameters $\{\beta_j, \boldsymbol{\mu}_j, \Sigma_j\}_{j=1}^J$.

We assume that the probability $p(\tilde{\mathbf{x}}|D_i)$ does not depend on the labeling of the training set, $\mathbf{t}^i$, and approximate $p(\tilde{\mathbf{x}}|D_i)$ by $p(\tilde{\mathbf{x}}|\hat{H}_i)$, where $\hat{H}^i$ denotes the estimated parameters by EM. Using a uniform prior for the subset choice i, we have

$$P(i|\tilde{\mathbf{x}}, D) \approx \frac{p(\tilde{\mathbf{x}}|\hat{H}_i)}{\sum_i p(\tilde{\mathbf{x}}|\hat{H}_i)} \tag{24}$$

Inserting equations (22) and (24) into (3), we obtain $p(\tilde{t}|\tilde{\mathbf{x}}, D_1, \ldots, D_L)$. If $p(\tilde{t} = 1|\tilde{\mathbf{x}}, D_1, \ldots, D_L) > p(\tilde{t} = -1|\tilde{\mathbf{x}}, D_1, \ldots, D_L)$, then $\tilde{t} = 1$; otherwise, $\tilde{t} = -1$.

Note that equation (24) can be viewed as a gating function as in mixture of experts [9]. Xu et al's approach actually models each subset's data by a Gaussian, while our gating function is more general.

3 Application: Mouse Pressure Pattern Detection

In this section, we apply the context-sensitive Bayesian classifier to mouse pressure data gathered from subjects filling out a multiple-page web form to put their resume online at a job site. After filling out a very long page and clicking to next web page, the subject receives an error message alerting her that the date format in the previous page is wrong. When the user goes back to the page, she sees that the page lost all of the data just entered. This event has a tendency to increase the user's frustration level, which is confirmed by our post-experiment questionnaire investigation and by some changes in mouse pressure patterns. The pressure signals are collected by a custom-made mouse equipped with eight pressure sensors [8]. The unprocessed mouse data is 8 dimensions of 8-bit analog data captured at 60 Hz.

Visually examining the pressure signals, we find that subjects have different patterns with respect to handling the mouse, and that almost all of them seem to have more activity right after the error notification/data loss event. For classification, the signal is labeled as -1 during the page preceding the error event and 1 during the page immediately following the event. Thus the two episodes being classified involve the exact same web task: filling out the same form; however, the state of the user may differ (and perhaps signal more frustration) on the second round. More details about the data collection experiment are in our previous paper [7]. For classification, we extract from the signals two features, the mean and the variance of the pressure signals over a half-second window (i.e., 30 data points because the signals are sampled at 60 Hz).

Five subjects filled out the forms using the mouse. For the i^{th} subject's data subset, we choose the nonlinear basis expansion $\boldsymbol{\phi}(\mathbf{x})$ in equation (4) over the data points as

Table 1. CS-EP, the Context Sensitive Expectation Propagation-trained Bayesian classifier uses less computation, memory, and time, and has slightly lower error rates than SVM on the average. Choosing different kernel widths improves the test performance of CS-EP.

Test Error Rate (%)	Test 1	Test 2	Test 3	Test 4	Test 5	Average Error	**Kernel Width**	**SVM's Error Rate (%)**
CS-EP	7.1	8.0	12.3	11.8	28.7	13.6	0.1	14.3
CS-EP	8.0	10.0	9.0	5.9	23.0	11.2	0.5	11.4
CS-EP	7.1	8.0	9.0	5.9	23.0	10.6	0.1 & 0.5	-

follows:

$$\boldsymbol{\phi}(\mathbf{x}) = [K(\mathbf{x}, \mathbf{x}_1), \cdots, K(\mathbf{x}^i, \mathbf{x}^i_j), \cdots, K(\mathbf{x}^i, \mathbf{x}^i_{N_i})]^T$$

where $K(\mathbf{x}, \mathbf{x}_i)$ is a basis function, which is chosen to be a Gaussian. Each data subset is randomly split into a training subset and a test set, in the ratio $60\% : 40\%$. We first train a Bayesian component classifiers on each training subset and model each subset by a mixture of two Gaussians, whose parameters are estimated by the EM algorithm. We choose 0.1 Gaussian kernel width for the first two component classifiers and 0.5 kernel width for the others. In the test phase, we pretend that we do not know which test data set is from which subject and let our algorithm adapt to the test data automatically. The test error rate on each test set and the average test error rate are shown in table 1.

For comparison, we train two SVMs with Gaussian kernel widths of 0.1 and 0.5 repectively, using all the data sets. The test results are also summarized in table 1. Although having comparable test performance to the context-sensitive classifiers, the SVMs compute much larger kernel matrices, require larger memory sizes and a longer training time. Choosing working sets can reduce the required memory size for SVMs, but many training iterations are needed over the split working sets for convergence. On the other hand, context-sensitive classifiers do not have to be iterated over different training sets. Also, our algorithm can use different kernel widths for different component classifiers to optimize the performance while the SVMs cannot.

4 Conclusion and Future Work

In this paper, we present a new context-sensitive Bayesian learning algorithm. This algorithm is trained efficiently and easily incorporates new knowledge. In addition, the algorithm can utilize different kernels for different component classifiers, i.e., different classifier complexities in different contexts/locations, to handle unstationary data distributions and at the same time keep the optimality of the Bayesian solution. By contrast, global learning algorithms, for example, SVM, and classical local learning algorithms like k-nearest neighbours cannot achieve the best compromise between locality and capacity [1].

We apply the context-sensitive Bayesian classifier to detect user-independent mouse patterns related to frustration and achieved significant classification results. This suggests the algorithm might be useful as part of an adaptive system that aims to better respond to users' frustration.

Future work includes improved modeling of the context with a full Bayesian treatment and detecting the hidden context in the training procedure if we do not know it during training. Also, in our algorithm there is no information sharing between the training of different component classifiers. A hierarchical data model may solve this problem. Finally, we will test our algorithm on more real world data sets.

References

[1] L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4(6):888–900, 1992.

[2] M. Harries and K. Horn. Learning stable concepts in domains with hidden changes in context. In *13th ICML, workshop on Learning in Context Sensitive Domains*, Bari, Italy, July 1996.

[3] R. Herbrich, T. Graepel, and C. Campbell. Bayes point machine: Estimating the Bayes point in kernel space. In *IJCAI Workshop SVMs*, pages 23–27, 1999.

[4] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

[5] H. J. Kushner and A. S. Budhiraja. A nonlinear filtering algorithm based on an approximation of the conditional distribution. *IEEE Transaction Automatic Control*, 45:580–585, 2000.

[6] T. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, MIT, Jan. 2001. www.media.mit.edu/˜tpminka/papers/learning.html.

[7] Y. Qi, C. Reynolds, and R. W. Picard. The Bayes point machine for computer-user frustration detection via pressuremouse. In *the 2001 Workshop on Perceptive User Interfaces*, Orlando, Florida, Nov. 2001.

[8] C. Reynolds. The sensing and measurement of frustration with computers. Master's thesis, MIT, 2001.

[9] L. Xu, M. I. Jordan, and G. E. Hinton. An alternative model for mixtures of experts. In G. Tesauro, D. Touretzky, and T. Leen, editors, *NIPS*, volume 7, pages 633–640. The MIT Press, 1995.