Kernelized Bayes Classifier

Ruye Wang

Department of Engineering, Harvey Mudd College
Claremont, CA 91711, USA

***Abstract*—This paper presents a new binary classification algorithm based on two existing methods, the classical naive Bayes classification and the kernel method. Specifically, the naive Bayes classification algorithm is reformulated so that it can be kernelized in such a way that all data points are mapped into a higher dimensional space in which the classes not separable in the original feature space become either linearly or quadratically separable. In comparison to the method of kernel based support vector machine, the kernelized Bayes classifier has two main advantages: first, like the naive Bayes method, it is a closed form algorithm and can therefore be implemented efficiently without iteration; second, it almost always outperforms the support vector machine in terms of the classification accuracy. Therefore the presented method can be used wherever the support vector machine is used, with lower computational costs as well as lower error rates.**

Keywords: binary classification, kernel method, naive Bayes classifier, support vector machine

I. Review of Bayes Classifier

The naive Bayes classifier [1] is a classical supervised classification algorithm, which, when trained by a set of N samples $\mathbf{X}=[\mathbf{x}_1,\cdots,\mathbf{x}_N]$ each labeled to be one of the K classes $\{C_1,\cdots,C_K\}$, classifies any unlabeled sample $\mathbf{x}$ into one of the K classes based on its posterior probability:

$$\text{If } P(C_l|\mathbf{x})=\max_{1\le k\le K}P(C_k|\mathbf{x}),\quad \text{then } \mathbf{x}\in C_l \tag{1}$$

Based on Bayes' theorem, this posterior can be obtained as

$$P(C_k|\mathbf{x})=\frac{p(\mathbf{x}|C_k)P(C_k)}{p(\mathbf{x})},\qquad (k=1,\cdots,K) \tag{2}$$

where

- $p(\mathbf{x}|C_k)$ is the likelihood of class C_k,
- $P(C_k)$ is the prior probability, which can be estimated as N_k/N with N_k being the number of training samples in C_k, $N=N_1+\cdots+N_K$,
- $p(\mathbf{x})$ is the marginal likelihood, the weighted sum of all likelihoods $p(\mathbf{x}|C_1),\cdots,P(\mathbf{x}|C_K)$:

$$p(\mathbf{x})=\sum_{k=1}^{K}p(\mathbf{x}|C_k)\,P(C_k) \tag{3}$$

The likelihood is typically assumed to be Gaussian (if there is no evidence for any other distribution):

$$\begin{aligned} p(\mathbf{x}|C_k) &= N(\mathbf{x},\mathbf{m}_k,\boldsymbol{\Sigma}_k)\\ &= \frac{1}{(2\pi)^{N/2}\,|\boldsymbol{\Sigma}_k|^{1/2}}\exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{m}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\mathbf{m}_k)\right] \end{aligned} \tag{4}$$

in terms of the mean vector $\mathbf{m}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$ of class C_k, which can be estimated based on the N_k training samples of C_k:

$$\begin{aligned} \mathbf{m}_k &= \frac{1}{N_k}\sum_{\mathbf{x}\in C_k}\mathbf{x}\\ \boldsymbol{\Sigma}_k &= \frac{1}{N_k}\sum_{\mathbf{x}\in C_k}(\mathbf{x}-\mathbf{m}_k)(\mathbf{x}-\mathbf{m}_k)^T \end{aligned} \tag{5}$$

As the classification is based on the relative comparison of the posterior $P(C_k|\mathbf{x})$, any monotonic mapping of the posterior, such as the logarithmic function, can be equivalently used to simplify the computation:

$$\begin{aligned} \log P(C_k|\mathbf{x}) &= \log\left[p(\mathbf{x}|C_k)P(C_k)/p(\mathbf{x})\right]\\ &= \log p(\mathbf{x}|C_k)+\log P(C_k)-\log p(\mathbf{x})\\ &= -\frac{1}{2}(\mathbf{x}-\mathbf{m}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\mathbf{m}_k)-\frac{N}{2}\log(2\pi)\\ &\quad -\frac{1}{2}\log|\boldsymbol{\Sigma}_k|+\log P(C_k)-\log p(\mathbf{x}) \end{aligned} \tag{6}$$

All terms independent of the index k can be dropped as they play no role in the comparison, and the resulting terms form the *quadratic discriminant function*:

$$D_k(\mathbf{x})=-\frac{1}{2}(\mathbf{x}-\mathbf{m}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\mathbf{m}_k)-\frac{1}{2}\log|\boldsymbol{\Sigma}_k|+\log P(C_k) \tag{7}$$

Specially, in the case of binary classification with $K=2$ classes C_+ and C_-, the feature space is partitioned into two regions corresponding to the two classes by a hyperquadratic decision boundary represented by the equation $D_+(\mathbf{x})=D_-(\mathbf{x})$, i.e.,

$$\begin{aligned} &-\frac{1}{2}(\mathbf{x}-\mathbf{m}_+)^T\boldsymbol{\Sigma}_+^{-1}(\mathbf{x}-\mathbf{m}_+)-\frac{1}{2}\,log\,|\boldsymbol{\Sigma}_+|+log\,P(C_+)\\ =\;& -\frac{1}{2}(\mathbf{x}-\mathbf{m}_-)^T\boldsymbol{\Sigma}_-^{-1}(\mathbf{x}-\mathbf{m}_-)-\frac{1}{2}\,log\,|\boldsymbol{\Sigma}_-|+log\,P(C_-) \end{aligned} \tag{8}$$

which can be rewritten in the following form:

$$f(\mathbf{x})=\mathbf{x}^T\mathbf{W}\mathbf{x}+\mathbf{w}^T\mathbf{x}+w=0 \tag{9}$$

where

$$\begin{aligned} \mathbf{W} &= -\frac{1}{2}(\boldsymbol{\Sigma}_+^{-1}-\boldsymbol{\Sigma}_-^{-1})\\ \mathbf{w} &= \boldsymbol{\Sigma}_+^{-1}\mathbf{m}_+-\boldsymbol{\Sigma}_-^{-1}\mathbf{m}_-\\ w &= -\frac{1}{2}(\mathbf{m}_+^T\boldsymbol{\Sigma}_+^{-1}\mathbf{m}_+-\mathbf{m}_-^T\boldsymbol{\Sigma}_-^{-1}\mathbf{m}_-)\\ &\quad -\frac{1}{2}log\,\frac{|\boldsymbol{\Sigma}_+|}{|\boldsymbol{\Sigma}_-|}+log\,\frac{P(C_+)}{P(C_-)} \end{aligned} \tag{10}$$

Any unlabeled sample $\mathbf{x}$ is classified into either of the two classes depending on whether its decision function below is greater or smaller than zero:

$$\text{If } f(\mathbf{x}) = \mathbf{x}^T\mathbf{W}\mathbf{x} + \mathbf{w}^T\mathbf{x} + w \begin{cases} >0 \\ <0 \end{cases}, \text{ then } \begin{cases} \mathbf{x} \in C_+ \\ \mathbf{x} \in C_- \end{cases} \tag{11}$$

Specially, if $\mathbf{\Sigma}_+ = \mathbf{\Sigma}_-$ and therefore $\mathbf{W} = \mathbf{0}$, the decision function becomes linear $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w$, and the decision boundary becomes a hyperplane.

II. Review of the Support Vector Machine

The support vector machine (SVM) [2], [3] is a binary classifier based on a set of training samples $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$, labeled respectively by $y_1, \cdots, y_N$, where $y_n = 1$ if $\mathbf{x}_n \in C_+$ and $y_n = -1$ if $\mathbf{x}_n \in C_-$. The SVM partitions the feature space by a hyperplane $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0$ into two regions for the two classes C_+ and C_-, where $\mathbf{w}$ is the normal vector of the hyperplane which can be obtained as

$$\begin{aligned} \mathbf{w} &= \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n = \sum_{n \in sv} \alpha_n y_n \mathbf{x}_n \\ &= \sum_{\mathbf{x}_n \in C_+, n \in sv} \alpha_n \mathbf{x}_n - \sum_{\mathbf{x}_n \in C_-, n \in sv} \alpha_n \mathbf{x}_n \end{aligned} \tag{12}$$

and b is a constant offset (bias) which can be found as

$$b = y_m - \mathbf{w}^T\mathbf{x}_m = y_m - \sum_{n \in sv} \alpha_n y_n \mathbf{x}_n^T \mathbf{x}_m, \quad m \in sv \tag{13}$$

Here $\alpha_1, \cdots, \alpha_N$ in the equations above are a set of N non-negative parameters that can be obtained by solving the following linearly constrained quadratic programming (QP) problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m \left(\mathbf{x}_n^T \mathbf{x}_m\right) \\ \text{subject to} \quad & \sum_{n=1}^{N} \alpha_n y_n = 0, \quad \alpha_n \geq 0 \quad (n = 1, \cdots, N) \end{aligned} \tag{14}$$

Those training vectors $\mathbf{x}_n$ corresponding to $\alpha_n > 0$ are the support vectors that determine $\mathbf{w}$ and b for the decision hyperplane, while all other vectors corresponding to $\alpha_n = 0$ play no role in the classification. In Eqs. (12) and (13), sv represents the set of all indices corresponding to the support vectors. Any unlabeled sample $\mathbf{x}$ is classified into either C_+ or C_-, depending on whether the following decision function is greater or smaller than zero:

$$\begin{aligned} &\text{If} \qquad f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{n \in sv} \alpha_n y_n \mathbf{x}_n^T \mathbf{x} + b \begin{cases} >0 \\ <0 \end{cases}, \\ &\text{then} \qquad \begin{cases} \mathbf{x} \in C_+ \\ \mathbf{x} \in C_- \end{cases} \end{aligned} \tag{15}$$

This SVM is in the same form as the binary Bayes classifier in Eq. (11) when $\mathbf{W} = \mathbf{0}$.

III. Kernelized Support Vector Machine

Two classes not linearly separable cannot be classified by the SVM as a linear classifier. This problem can be overcome by the kernel method [4], by which all data vectors $\mathbf{x}_n$ in the original feature space are mapped into another vector $\phi(\mathbf{x}_n)$ in a much higher dimensional space, where the data points can be linearly separated. The use of the kernel method requires that all data points in the algorithm are in the form of an inner product $\mathbf{x}_i^T\mathbf{x}_j$, which is then replaced by a scalar kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{z}_j)$. A major advantage of the kernel method is that the mapping $\phi(\mathbf{x})$ does not actually need to be carried out explicitly. Popular kernel functions include, among others, the linear kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j \tag{16}$$

corresponding to an identity kernel mapping $\phi(\mathbf{x}) = \mathbf{x}$, and the Gaussian or radial basis function (RBF) kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma ||\mathbf{x}_i - \mathbf{x}_j||} \tag{17}$$

corresponding to an implicit mapping $\phi(\mathbf{x})$ to an infinite dimensional space.

To apply the kernel method, we replace all inner products in Eqs. (13) and (15) by the corresponding kernel functions to get

$$b = y_m - \sum_{n \in sv} \alpha_n y_n \, K(\mathbf{x}_n, \mathbf{x}_m) \tag{18}$$

and

$$\text{If } f(\mathbf{x}) = \sum_{n \in sv} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + b \begin{cases} >0 \\ <0 \end{cases}, \text{ then } \begin{cases} \mathbf{x} \in C_+ \\ \mathbf{x} \in C_- \end{cases} \tag{19}$$

IV. Kernelized Binary Bayes Classifier

As the Bayes classifier discussed above is based on a quadratic decision function $f(\mathbf{x}) = \mathbf{x}^T\mathbf{W}\mathbf{x} + \mathbf{w}^T\mathbf{x} + w$, it is unable to separate classes that are not quadratically separable. Again, like in the SVM case, the kernel method can be used to overcome this problem, if the algorithm can be reformulated in such a way that all data points appear in the form of an inner product. To do so, we first consider the eigendecompositions of the three matrices $\mathbf{W} = -(\mathbf{\Sigma}_+^{-1} - \mathbf{\Sigma}_-^{-1})/2$, $\mathbf{\Sigma}_+$ and $\mathbf{\Sigma}_-$. Let $\mathbf{\Lambda}$, $\mathbf{\Lambda}_+$ and $\mathbf{\Lambda}_-$ be their eigenvalue matrices and $\mathbf{V}$, $\mathbf{V}_+$, and $\mathbf{V}_-$ the corresponding eigenvector matrices, then we can write these matrices as

$$\begin{aligned} \mathbf{W} &= \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T = \mathbf{U}\mathbf{U}^T \\ \mathbf{\Sigma}_+^{-1} &= \mathbf{V}_+\mathbf{\Lambda}_+\mathbf{V}_+^T = \mathbf{U}_+\mathbf{U}_+^T \\ \mathbf{\Sigma}_-^{-1} &= \mathbf{V}_-\mathbf{\Lambda}_-\mathbf{V}_-^T = \mathbf{U}_-\mathbf{U}_-^T \end{aligned} \tag{20}$$

where $\mathbf{U} = \mathbf{V}\mathbf{\Lambda}^{1/2}$, $\mathbf{U}_+ = \mathbf{V}_+\mathbf{\Lambda}_+^{1/2}$ and $\mathbf{U}_- = \mathbf{V}_-\mathbf{\Lambda}_-^{1/2}$. Now vector $\mathbf{w}$ can be written as:

$$\begin{aligned} \mathbf{w} &= \mathbf{\Sigma}_+^{-1}\mathbf{m}_+ - \mathbf{\Sigma}_-^{-1}\mathbf{m}_- \\ &= \mathbf{U}_+\mathbf{U}_+^T \frac{1}{N_+} \sum_{\mathbf{x} \in C_+} \mathbf{x} - \mathbf{U}_-\mathbf{U}_-^T \frac{1}{N_-} \sum_{\mathbf{x} \in C_-} \mathbf{x} \end{aligned} \tag{21}$$

and the quadratic decision function in Eq. (11) of any unlabeled sample $\mathbf{x}'$ can be written as:

$$\begin{aligned} & f(\mathbf{x}') = \mathbf{x}'^T\mathbf{W}\mathbf{x}' + \mathbf{w}^T\mathbf{x}' + w \\ = \; & \mathbf{x}'^T\mathbf{U}\mathbf{U}^T\mathbf{x}' + \left(\frac{1}{N_+}\sum_{\mathbf{x}\in C_+}\mathbf{x}^T\mathbf{U}_+\mathbf{U}_+^T\right)\mathbf{x}' \\ & - \left(\frac{1}{N_-}\sum_{\mathbf{x}\in C_-}\mathbf{x}^T\mathbf{U}_-\mathbf{U}_-^T\right)\mathbf{x}' + w \\ = \; & \mathbf{z}'^T\mathbf{z}' + \frac{1}{N_+}\sum_{\mathbf{z}_+\in C_+}\left(\mathbf{z}_+^T\mathbf{z}'_+\right) - \frac{1}{N_-}\sum_{\mathbf{z}_-\in C_-}\left(\mathbf{z}_-^T\mathbf{z}'_-\right) \end{aligned} \tag{22}$$

where

$$\left\{\begin{array}{l} \mathbf{z}_+ = \mathbf{U}_+^T\mathbf{x} \;\; (\mathbf{x}\in C_+) \\ \mathbf{z}_- = \mathbf{U}_-^T\mathbf{x} \;\; (\mathbf{x}\in C_-) \end{array}\right. , \quad \left\{\begin{array}{l} \mathbf{z}' = \mathbf{U}^T\mathbf{x}' \\ \mathbf{z}'_+ = \mathbf{U}_+^T\mathbf{x}' \\ \mathbf{z}'_- = \mathbf{U}_-^T\mathbf{x}' \end{array}\right. \tag{23}$$

As all data points appear in the form of an inner product, we can use the kernel method by replacing all inner products in the decision function with the corresponding kernel functions:

$$\begin{aligned} f(\mathbf{x}') &= K(\mathbf{z}',\mathbf{z}') + \frac{1}{N_+}\sum_{\mathbf{z}_+\in C_+}K(\mathbf{z}_+,\mathbf{z}'_+) \\ &\quad -\frac{1}{N_-}\sum_{\mathbf{z}_-\in C_-}K(\mathbf{z}_-,\mathbf{z}'_-) + b \\ &= p(\mathbf{x}') + b \end{aligned} \tag{24}$$

This is the decision function in the new higher dimensional space, where $p(\mathbf{x}')$ is defined as a function composed of all terms in $f(\mathbf{x}')$ except the scalar constant term b, which, as the offset in the new space, is to replace w in the original space. To find this b, we first map all training samples into a 1-D space $x_n = p(\mathbf{x}_n),\ (n = 1,\cdots,N)$ and sort them, together with their corresponding labelings $y_1,\cdots,y_N$, into $x_1 \le \cdots \le x_N$, and then search through all $N-1$ possible ways to partition them into two groups indexed respectively by $1,\cdots,l$ and $l+1,\cdots,N$ to find the optimal k corresponding to the maximum labeling consistency:

$$k = \underset{1\le l\le N-1}{\arg\max}\left(\left|\sum_{n=1}^{l} y_n\right| + \left|\sum_{n=l+1}^{N} y_n\right|\right) \tag{25}$$

The middle point $(x_k + x_{k+1})/2$ between x_k and x_{k+1} can be used as the optimal threshold to separate the training samples into two classes in the 1-D space, i.e., the offset is $b = -(x_k + x_{k+1})/2$. Now the unlabeled sample $\mathbf{x}'$ can be classified into either of the two classes C_+ and C_-:

$$\text{If } f(\mathbf{x}') = p(\mathbf{x}') + b \left\{\begin{array}{l} >0 \\ <0 \end{array}\right. , \text{ then } \left\{\begin{array}{l} \mathbf{x}'\in C_+ \\ \mathbf{x}'\in C_- \end{array}\right. \tag{26}$$

In general, when all data points are kernel mapped into a higher dimensional space, the two classes can be more easily separated even by a hyperplane corresponding to the linear part of the decision function without the second order term $K(\mathbf{z}',\mathbf{z})$ in Eq. (24). This justifies the assumption that the two classes have the same covariance matrix so that $\mathbf{W} = -(\mathbf{\Sigma}_+ - \mathbf{\Sigma}_-)/2 = \mathbf{0}$ and the second order term is dropped. We can therefore consider the following two special cases:

- Both covariance matrices $\mathbf{\Sigma}_+$ and $\mathbf{\Sigma}_-$ are approximated by their average $\mathbf{\Sigma} = (\mathbf{\Sigma}_+ + \mathbf{\Sigma}_-)/2$, then $\mathbf{W} = \mathbf{0}$ and

$$\mathbf{w} = \mathbf{\Sigma}^{-1}(\mathbf{m}_+ - \mathbf{m}_-) = \mathbf{U}\mathbf{U}^T\left(\frac{1}{N_+}\sum_{\mathbf{x}\in C_+}\mathbf{x} - \frac{1}{N_-}\sum_{\mathbf{x}\in C_-}\mathbf{x}\right) \tag{27}$$

  where $\mathbf{U}\mathbf{U}^T = \mathbf{\Sigma}^{-1}$ is the eigendecomposition of $\mathbf{\Sigma}^{-1}$. The decision function of an unlabeled sample $\mathbf{x}'$ becomes

$$\begin{aligned} f(\mathbf{x}') &= \mathbf{w}^T\mathbf{x}' + w \\ &= \frac{1}{N_+}\sum_{\mathbf{x}\in C_+}\mathbf{x}^T\mathbf{U}\mathbf{U}^T\mathbf{x}' - \frac{1}{N_-}\sum_{\mathbf{x}\in C_-}\mathbf{x}^T\mathbf{U}\mathbf{U}^T\mathbf{x}' + w \\ &= \frac{1}{N_+}\sum_{\mathbf{z}\in C_+}\mathbf{z}^T\mathbf{z}' - \frac{1}{N_-}\sum_{\mathbf{z}\in C_-}\mathbf{z}^T\mathbf{z}' + w \end{aligned} \tag{28}$$

  where $\mathbf{z} = \mathbf{U}^T\mathbf{x}$ and $\mathbf{z}' = \mathbf{U}^T\mathbf{x}'$. Replacing the inner products by the corresponding kernel functions we get:

$$f(\mathbf{x}') = \frac{1}{N_+}\sum_{\mathbf{z}\in C_+}K(\mathbf{z},\mathbf{z}') - \frac{1}{N_-}\sum_{\mathbf{z}\in C_-}K(\mathbf{z},\mathbf{z}') + b \tag{29}$$

- More specially, $\mathbf{\Sigma}_+ = \mathbf{\Sigma}_- = \mathbf{I}$, then $\mathbf{W} = \mathbf{0}$ and

$$\mathbf{w} = \mathbf{m}_+ - \mathbf{m}_- = \frac{1}{N_+}\sum_{\mathbf{x}\in C_+}\mathbf{x} - \frac{1}{N_-}\sum_{\mathbf{x}\in C_-}\mathbf{x} \tag{30}$$

  Comparing this with Eq. (12) for the SVM, we see that in both cases the normal vector $\mathbf{w}$ of the decision plane is a vector between some weighted mean vectors of the two classes. The decision function becomes

$$f(\mathbf{x}') = \mathbf{w}^T\mathbf{x}' + w = \frac{1}{N_+}\sum_{\mathbf{x}\in C_+}\mathbf{x}^T\mathbf{x}' - \frac{1}{N_-}\sum_{\mathbf{x}\in C_-}\mathbf{x}^T\mathbf{x}' + w \tag{31}$$

  Applying the kernel method, we have:

$$f(\mathbf{x}') = \frac{1}{N_+}\sum_{\mathbf{x}\in C_+}K(\mathbf{x},\mathbf{x}') - \frac{1}{N_-}\sum_{\mathbf{x}\in C_-}K(\mathbf{x}^T\mathbf{x}') + b \tag{32}$$

We make a special note that when the kernel method is used to replace the inner products in the decision function by a kernel function $K(\mathbf{x},\mathbf{x}') = \phi(\mathbf{x})^T\phi(\mathbf{x}')$ (e.g,, in Eqs. (28) and (31)), we need to map all data points to the higher dimensional space $\phi(\mathbf{x})$, instead of only mapping certain weighted sum of these data points such as $\phi(\mathbf{w})$ or $\phi(\mathbf{m}_+) - \phi(\mathbf{m}_-)$. This is because the mapping of the sum of data points is not equal to the sum of the mapping of these points if the kernel is not linear $K(\mathbf{x}_i,\mathbf{x}_j) \ne \mathbf{x}_i^T\mathbf{x}_j$:

$$\begin{aligned} & \phi(\mathbf{w}) \ne \phi(\mathbf{m}_+) - \phi(\mathbf{m}_-) \\ & \phi(\mathbf{m}_\pm) = \phi\left(\frac{1}{n_\pm}\sum_{\mathbf{x}_n\in C_\pm}\mathbf{x}_n\right) \ne \frac{1}{n_\pm}\sum_{\mathbf{x}_n\in C_\pm}\phi(\mathbf{x}_n) \end{aligned} \tag{33}$$

V. Testing Results

The three variations of the kernelized Bayes method, including the two special cases based on the decision functions in Eqs. (32) and (29), as well as the general case in Eqs. (24), hereby referred to as Type I, Type II and Type III, respectively, are tested based on a set of eight 2-D simulated datasets, and their performances are compared with that of the SVM method, implemented by the Matlab function:

fitcsvm(X',y,'Standardize',true,'KernelFunction','linear','Kernel

Both the linear kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$ and the RBF kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||)$ are used in the tests. The value of the parameter γ used in the may vary in a wide range (e.g., $1 < \gamma < 9$) to make proper trade-off between low error rates and avoiding overfitting.

The last five datasets are generated by the Matlab code provided on this page to demonstrate the performance of Matlab's implementation of the SVM method: https://www.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html.

The correct rates of the four methods are listed in the table below:

	SVM	Bayes I	Bayes II	Bayes III
Test 1	93.0%	88.0%	94.0%	94.0%
Test 2	73.0%	80.0%	79.5%	96.5%
Test 3	98.5%	100%	100%	100%
Test 4	58.75%	60.0%	60.25%	97.0%
Test 5	97.75%	98.50%	98.50%	99.50%
Test 6	98.0%	100%	100%	100%
Test 7	96.0%	98.5%	95.5%	97.0%
Test 8	96.0%	97.0%	91.0%	93.0%

and the actual partitionings of the 2-D feature space are shown in the following figures. In each figure, the results of the corresponding test of the four methods, the SVM, Types I, II, and III of the kernelized Bayes method, are shown in the top-left, top-right, bottom-left, and bottom-right panels, respectively.

- Test 1: Two classes with identical covariance matrices, linear kernel
  The kernel Bayes methods II and III both achieve the best result (94%), slightly better than that of the SVM (93%). But the kernel Bayes type I performs poorly (84.7%), as it is based only on the means of the two classes while their covariances representing the distribution of the data points are ignored.
- Test 2: Two classes linearly inseparable, linear kernel
  The Bayes type III achieves the best result (96.5%), due to its quadratic term in the decision function, Bayes I and II perform much more poorly, but still slightly better than the SVM method.
- Test 3: The same dataset as in the previous test, RBF kernel
  While the SVM performs well (98.5%), all three versions of the kernelized Bayes method (with $\gamma = 5$) achieve perfect result with all points of the two classes correctly classified. However, the partitioning of the space by Bayes III is fragmented, showing a sign of overfitting.
- Test 4: Exclusive OR, linear kernel
  As the two classes are not linearly separable, all methods based on linear kernel perform poorly except Bayes III, which successfully partitions the space into two regions for the XOR data by a pair of hyperbola, due to its second order decision function.
- Test 5: Exclusive OR, RBF kernel
  All methods perform well, but the three variations of the kernelized Bayes method achieve slightly higher correct rates than the SVM.
- Test 6: Concentric ring dataset, RBF kernel
  All three types of the Bayes method achieve perfect classification, while the SVM has 2% error. Interestingly, the Type III of the Bayes method

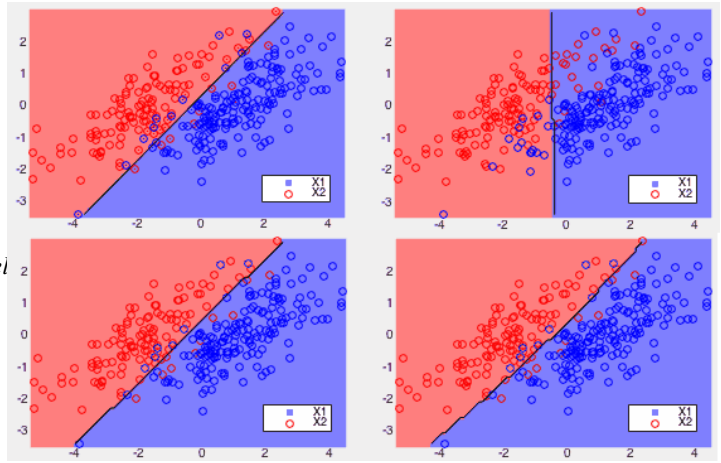

Fig. 1. Test 1: Classes of Identical covariance, linear kernel

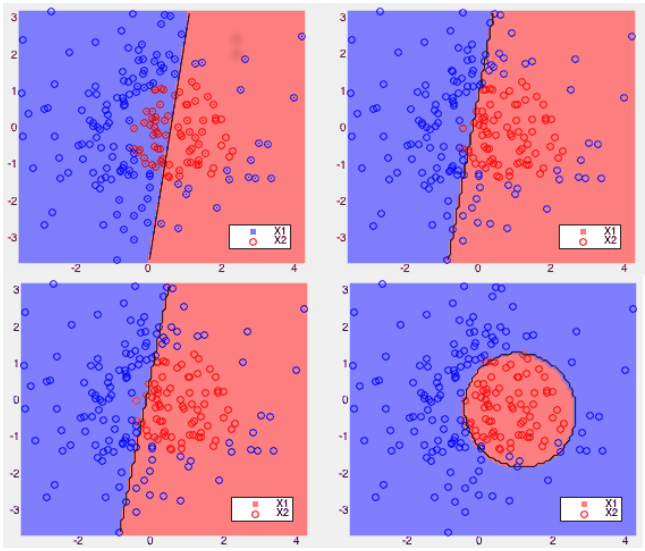

Fig. 2. Test 2: Linearly inseparable classes, linear kernel

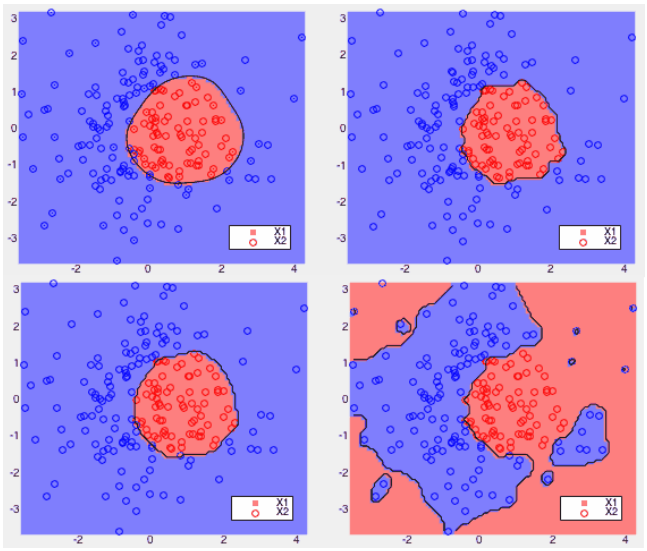

Fig. 3. Test 3: Linearly inseparable classes, RBF kernel

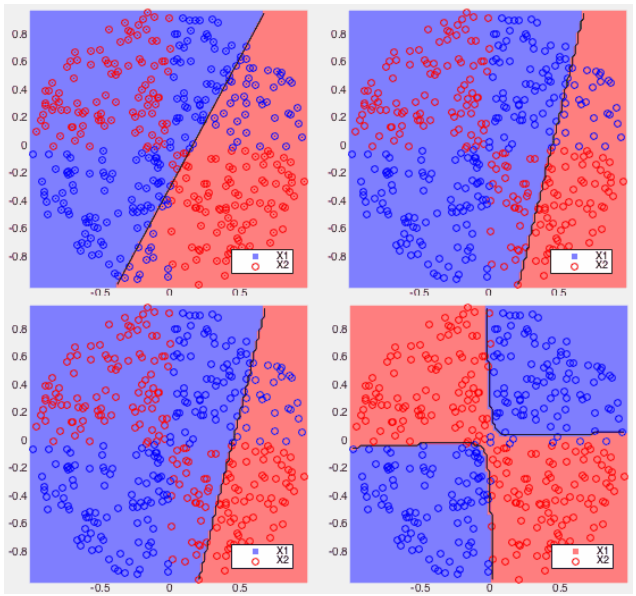

Fig. 4. Test 4: XOR dataset, linear kernel

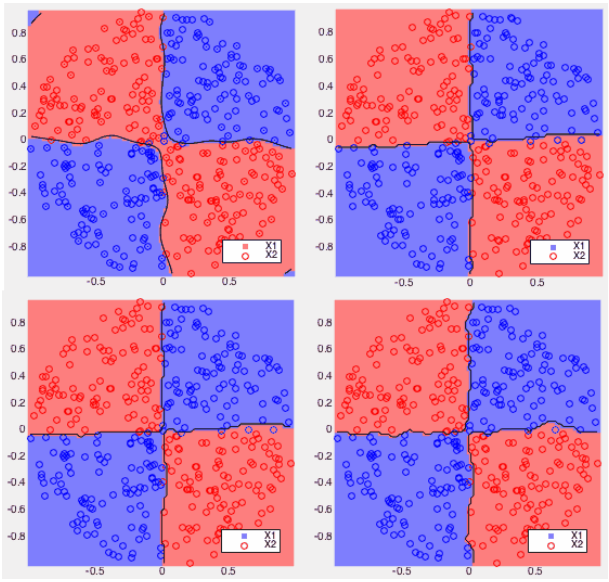

Fig. 5. Test 5: XOR dataset, RBF kernel

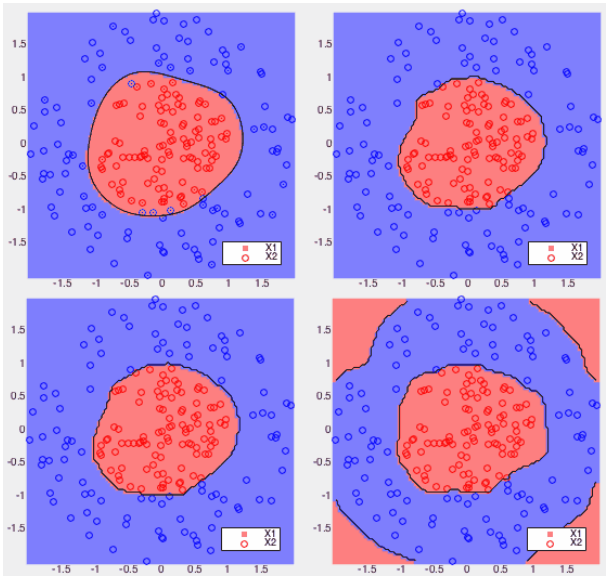

Fig. 6. Test 6: Concentric ring dataset, RBF kernel

- Test 7: Multi-cluster dataset, RBF kernel ($\gamma = 5$)
  Types I and III achieve slightly better results than the SVM and Type II of Bayes, with the price of more fragmented partitioning.
- Test 8: Multi-cluster dataset, RBF kernel ($\gamma = 3$)
  Types I achieves slightly better result than the SVM, but both Types II and III of Bayes perform more poorly than SVM. However, the partitionings are less fragmented. Comparing these results with those in the previous test, we see that by changing the parameter γ, different trade-offs can be made between error rate and fragmentation.

VI. Conclusion

As a classical classification method, the naive Bayes classifier is widely used because it is a closed form algorithm that is simple to implement. However, it is limited to data that are quadratically separable. Based on the eigendecomposition of the covariance matrices of the two classes used in the algorithm, the kernelized Bayes classifier presented in this paper significantly improves the performance of the naive Bayes algorithm as a binary classifier by reformulating it in such a way that all data appear in the algorithm in the form of an inner product so that the kernel method can be used.

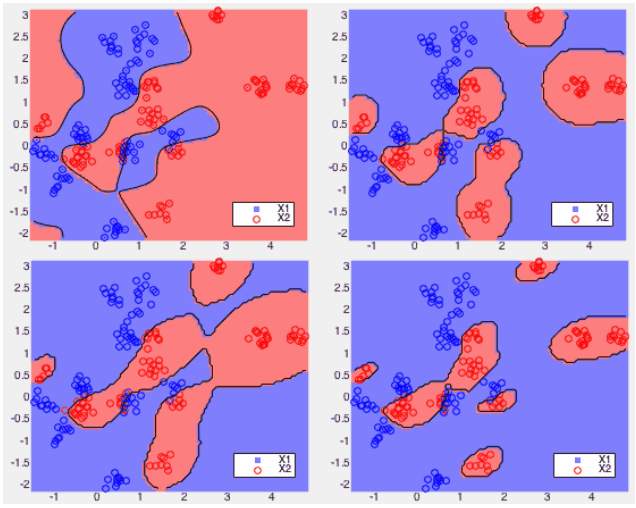

Fig. 7. Test 7: Multi-cluster dataset, RBF kernel ($\gamma = 5$)

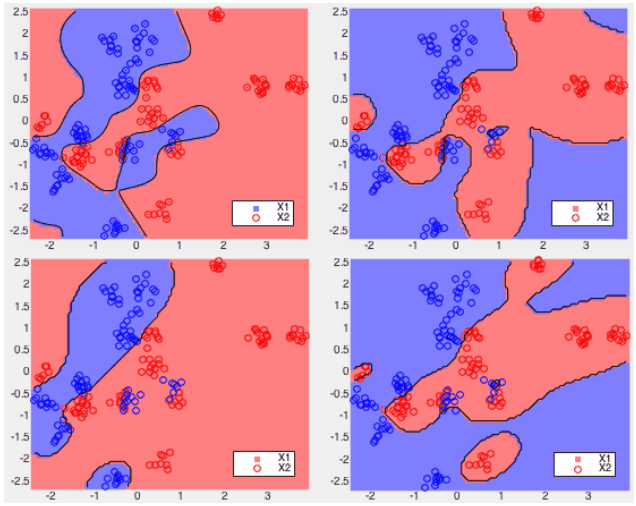

Fig. 8. Test 8: Multi-cluster dataset, RBF kernel ($\gamma = 3$)

In comparison with the SVM also as a binary classifier, the kernelized Bayes method has two main advantages. First, it is a closed form algorithm and therefore can be conveniently and efficiently carried out without any iteration, while the SVM requires solving a quadratic programming problem iteratively by either the interior point method or the sequential minimization optimization (SMO) method. Second, the performance of the kernelized Bayes method is very much comparable with, often better than, that of the SVM, in terms of the classification error rate, as shown in the examples above. We can therefore conclude that the kernelized Bayes method can be used wherever the SVM is used, with lower computational costs as well as lower error rates.

REFERENCES

[1] Duda, R.; Hart, P. *Pattern Classification*, (1973). Publisher: Wiley.

[2] Boser, B. E.; Guyon, I. M.; Vapnik, V. N. (1992). "A training algorithm for optimal margin classifiers". Proceedings of the fifth annual workshop on Computational learning theory, COLT 1992. p. 144

[3] Cortes, C.; Vapnik, V. (1995). "Support-Vector Networks". Journal of Machine Learning, Vol. 20 (3), p. 273

[4] Shawe-Taylor, J.; Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*, publisher: Cambridge University Press.