Operations Research Letters 50 (2022) 423–429

Contents lists available at ScienceDirect

Operations Research Letters

www.elsevier.com/locate/orl

Support vector machines as Bayes' classifiers

Peter L. Jackson

Singapore University of Technology and Design, 8 Somapah Road, 487372 Singapore

ARTICLE INFO

Article history:
Received 16 November 2021
Received in revised form 4 May 2022
Accepted 14 June 2022
Available online 20 June 2022

Keywords:
Support vector machines
Naïve Bayes' classifier
Multi-class classification

ABSTRACT

We reformulate the problem of determining support vectors directly as an application of Bayes' classifiers rather than as the dual program to a binary geometric separation problem. The primary purpose of the reformulation is to create a simpler exposition of the support vector machines technique. A secondary advantage is that it immediately and naturally applies to multi-class classification problems where the kernel function can be normalized as a density.

1. Introduction

Traditional expositions of the support vector machines (SVM) technique (for example, [1], [2]) start with the geometric problem of separating two non-overlapping vector sets by choosing a hyper-plane with maximum margin. This is then extended to allow overlapping vector sets or soft margins. A cost parameter is introduced to control the tradeoff between misclassification and margin maximization. The dual program is then derived. The kernel trick is applied and multiple kernel functions are considered. This development is mathematically appealing: it provides geometric intuition, a nice application of duality theory, and it reveals the power of the kernel trick. What is missing from these traditional expositions is an interpretation of the resulting dual program. Also, from a pedagogical viewpoint, the kernel trick complicates the geometric intuition of separating hyper-planes and demands a much higher level of spatial reasoning to visualize. Also, because the SVM is only a binary classifier, the technique must be run in multiple combinations (one against one or one against many) to handle multi-class problems. Finally, another layer of modeling is required to extract classification probabilities from the paired comparisons. This paper can be considered as an alternative to traditional expositions by offering a Bayes' classifier interpretation of the SVM dual program. The equivalence we establish is restricted to kernel functions which can be normalized as kernel density functions, such as Gaussian radial basis functions. The reasoning required from a pedagogical viewpoint is reduced to visualization of empirical probability density functions based on kernel densities, manipulation of quantities from Bayes' Identity, and basic tools of optimization including Lagrangean relaxation. What emerges as a by-product of this exposition is the natural extension of the Two Class SVM, for kernel density functions, to a true Multi-Class SVM with classification probabilities. We do not claim superiority of these probability estimates over existing methods but present them only as natural extensions of the multi-class formulation.

2. Review: the naïve Bayes' classifier

We consider a sample data set of (input, label) pairs, (x_i, y_i) for observations $i \in I$, where input $x_i \in \mathbb{R}^d$ and label y_i is taken to be a one-hot vector $y_i = (y_{ic})_{c\in C}$ where C is the set of classes. By a one-hot vector, we mean $y_{ic} \in \{0, 1\}$ for each $c \in C$ and $\sum_{c\in C} y_{ic} = 1$. We use the superscript 1 to indicate the conversion of a class index, $c \in C$, to a one hot-hot vector. Hence, $y_i = c^1$ if and only if $y_{ic} = 1$. The data set is assumed to be balanced in terms of the number of labels of each class and mis-classifications of each class are assumed to be equally important.

A kernel density function $k(\cdot, \cdot)$ is a mapping $\mathbb{R}^d \times \mathbb{R}^d$ into $[0, 1]$, which is symmetric, non-negative, and integrates to 1 over either argument. It is typically parameterized by a bandwidth, h, which we suppress from the notation. We consider an empirical density function for a random input X as the average of kernel densities centered at the observed inputs.

$$P_I\{X \in (x)_{dx}\} = \frac{1}{|I|}\sum_{i\in I} k(x, x_i),$$

where $|I|$ denotes the number of elements in set I. We use the notation $(x)_{dx}$ to suggest an infinitesimally small region surrounding a point x. We also subscript the probability with the set I to suggest that this is an empirical estimate of the true probability based

E-mail address: peter_jackson@sutd.edu.sg.

https://doi.org/10.1016/j.orl.2022.06.003
0167-6377/

on the observations in set I. Similarly, we estimate a conditional density function for inputs belonging to each class using

$$P_I\{X \in (x)_{dx} | Y = c^1\} = \frac{1}{\sum_{i\in I} y_{ic}} \sum_{i\in I} y_{ic} k(x, x_i).$$

Finally, we estimate the marginal probability mass function for the labels as

$$P_I\{Y = c^1\} = \frac{1}{|I|} \sum_{i\in I} y_{ic}.$$

Bayes' Identity can be written as

$$P_I\{Y = c^1 | X \in (x)_{dx}\} = \frac{P_I\{X \in (x)_{dx} | Y = c^1\} P_I\{Y = c^1\}}{P_I\{X \in (x)_{dx}\}}.$$

Substituting with the empirical density functions and canceling terms, we get

$$P_I\{Y = c^1 | X \in (x)_{dx}\} = \frac{\sum_{i\in I} y_{ic} k(x, x_i)}{\sum_{i\in I} k(x, x_i)}.$$

Using $argmax^1()$ to represent the one-hot version of the $argmax()$ function, the naïve Bayes' Classifier could then be described as:

$$\begin{aligned} C(x) &= argmax^1_{c\in C} \left(P_I\{Y = c^1 | X \in (x)_{dx}\} \right) \\ &= argmax^1_{c\in C} \left(\frac{\sum_{i\in I} y_{ic} k(x, x_i)}{\sum_{i\in I} k(x, x_i)} \right) \\ &= argmax^1_{c\in C} \left(\sum_{i\in I} y_{ic} k(x, x_i) \right) \end{aligned}$$

since the common denominator does not affect the $argmax()$ function.

3. The multi-class support vector machine

In the sequel, we use a subset of observations, the *support*, to estimate these classification probabilities. Let $S \subseteq I$ denote the support set. The only requirement we impose on the support set is that it be representative of the full set I in the sense of maintaining the balance of classes represented. Since we assume the full set I is perfectly balanced, for each $c \in C$, we require:

$$\sum_{i\in S} y_{ic} = \frac{|S|}{|C|}$$

where, for the time being, we assume $|S|$ is an integer multiple of $|C|$.

We now subscript the empirical probability functions by S to remind ourselves that we are dealing with less complete information. By analysis similar to the previous section we arrive at the following empirical conditional probability:

$$P_S\{Y = c^1 | X \in (x)_{dx}\} = \frac{\sum_{i\in S} y_{ic} k(x, x_i)}{\sum_{i\in S} k(x, x_i)}.$$

The naïve Bayes' Classifier is to predict the class based on the largest such conditional probability. The support-restricted Bayes' classifier function can be written as:

$$\begin{aligned} C_S(x) &= argmax^1_{c\in C} \left\{ P_S\{Y = c^1 | X \in (x)_{dx}\} \right\} \\ &= argmax^1_{c\in C} \left\{ \sum_{i\in S} y_{ic} k(x, x_i) \right\}. \end{aligned}$$

3.1. A probabilistic decision rule

It is difficult to apply calculus with an $argmax^1(\cdot)$ decision rule. For the purpose of analysis we use a probabilistic decision rule. Let $p_c(x) = P_S\{Y = c^1 | X \in (x)_{dx}\}$ and let $F(c, x) = \sum_{c'=1}^{c} p_{c'}(x)$, the cumulative distribution function for Y given X. Then, letting U denote a Uniform(0,1) random variable, it is well known that the random variable generated by $\tilde{C} = F^{-1}(U, x)$ will have $F(c, x)$ as its cumulative distribution function. Hence, we consider a probabilistic decision rule, $\tilde{C}_S(x)$, a one-hot random vector chosen such that

$$P_S\{\tilde{C}_S(x) = c^1\} = P_S\{Y = c^1 | X \in (x)_{dx}\}.$$

When applied to observation $i \in I$, the probability that this decision rule results in a correct classification is given by

$$\begin{aligned} P_S\{\tilde{C}_S(x_i) = y_i\} &= \sum_{c\in C} y_{ic} P_S\{\tilde{C}_S(x_i) = c^1\} \\ &= \sum_{c\in C} y_{ic} P_S\{Y = c^1 | X \in (x_i)_{dx}\} \\ &= \sum_{c\in C} y_{ic} \frac{\sum_{j\in S} y_{jc} k(x_i, x_j)}{\sum_{j\in S} k(x_i, x_j)} \\ &= \frac{\sum_{j\in S} \sum_{c\in C} y_{ic} y_{jc} k(x_i, x_j)}{\sum_{j\in S} k(x_i, x_j)} \\ &= \frac{\sum_{j\in S} y_i \cdot y_j k(x_i, x_j)}{\sum_{j\in S} k(x_i, x_j)} \end{aligned}$$

where $y_i \cdot y_j$ is the dot product of the label vectors for the i^{th} and j^{th} observations. Henceforth, we abbreviate $k(x_i, x_j)$ to k_{ij}. The matrix of these quantities, called the *kernel matrix*, $K = [k_{ij}]$, is computed only once, at the beginning of the algorithm.

Using this approach, we could formulate a wide range of optimization problems depending on the characteristics of the solution we seek. In particular, inspired by the dual to the SVM, we consider the following metric:

$$R(S) = \sum_{i\in S} \left(P_S\{\tilde{C}(x_i) \neq y_i\} - P_S\{\tilde{C}(x_i) = y_i\} \right) P_S\{X \in (x_i)_{dx}\}$$

Support sets which score highly on the $R(S)$ metric will be characterized by high probabilities of misclassifications $P_S\{\tilde{C}(x_i) \neq y_i | X \in (x_i)_{dx}\}$ and low probabilities of correct classifications ($P_S\{\tilde{C}(x_i) = y_i | X \in (x_i)_{dx}\}$). Heuristically speaking, vectors which have high probabilities of being classified correctly under the current support do not need to be in the support set. Only the most difficult vectors to classify should be in the support. But this is moderated by the factor $P_S\{X \in (x_i)_{dx}\}$ which is the likelihood of observing a vector such as x_i. That is, outliers are discouraged from belonging to the support.

We are led to consider a support set optimization problem of the following form:

$$\begin{aligned} &\text{maximize}_{S\subseteq I} R(S) \\ &\quad \text{subject to:} \\ &\qquad |S| = \bar{n}_S, \\ &\qquad |S_c| = \frac{1}{|C|}|S|, \text{ for each } c \in C, \end{aligned}$$

where $\bar{\eta}_S$ is a target value for the support set count.

3.2. Reformulations and relaxations

Since $|S|$ is fixed in our optimization ($|S| = \bar{n}_S$), the result is not affected by scaling the objective by $|S|$, changing it to $Z(S) = R(S)|S|$. We can then exploit the following simplification.

Lemma 1.

$$Z(S) = R(S)|S| = \sum_{i\in S}\sum_{j\in S}(1 - 2y_i \cdot y_j)k_{ij}$$

Proof.

$$\begin{aligned}
Z(S) &= R(S)|S| \\
&= \sum_{i\in S}\big(P_S\{\tilde{C}(x_i) \neq y_i\} - P_S\{\tilde{C}(x_i) = y_i\}\big)P_S\{X \in (x_i)_{dx}\}|S| \\
&= \sum_{i\in S}\big(1 - 2P_S\{\tilde{C}(x_i) = y_i\}\big)P_S\{X \in (x_i)_{dx}\}|S| \\
&= \sum_{i\in S}\big(1 - 2\frac{\sum_{j\in S} y_i \cdot y_j k_{ij}}{\sum_{j\in S} k_{ij}}\big)\frac{1}{|S|}\sum_{j\in S} k_{ij}|S| \\
&= \sum_{i\in S}\big(\sum_{j\in S} k_{ij} - 2\sum_{j\in S} y_i \cdot y_j k_{ij}\big) \\
&= \sum_{i\in S}\sum_{j\in S}(1 - 2y_i \cdot y_j)k_{ij}. \quad \square
\end{aligned}$$

Remark. The quantity $1 - 2y_i \cdot y_j$ has the following interpretation:

$$1 - 2y_i \cdot y_j = \begin{cases} +1, & \text{if } y_i \neq y_j; \\ -1, & \text{if } y_i = y_j. \end{cases} \tag{1}$$

Let $\alpha_i \in \{0, 1\}$ denote membership of the i^{th} vector in the support set. Then $S = \{i \in I; \alpha_i > 0\}$ and we re-express support vector optimization as an integer program:

$$\text{maximize}_\alpha\, Z(\alpha) = \sum_{i\in I}\sum_{j\in I}\alpha_i\alpha_j(1 - 2y_i \cdot y_j)k_{ij}$$

subject to:

$$\sum_{j\in I}\alpha_j = \bar{n}_S,$$

$$\sum_{j\in I}\alpha_j y_{jc} = \frac{1}{|C|}\sum_{j\in I}\alpha_j, \text{ for each } c \in C,$$

$$\alpha_j \in \{0, 1\} \text{ for all } j \in I.$$

Next we relax the integer constraints, requiring only $0 \leq \alpha_j \leq 1$ for each $j \in I$. The *support set* S is then the set of observations for which $\alpha_j > 0$. We also apply Lagrangian relaxation to the first constraint using Lagrange multiplier θ.

$$\text{maximize}_\alpha \sum_{i\in I}\sum_{j\in I}\alpha_i\alpha_j(1 - 2y_i \cdot y_j)k_{ij} - \theta(\bar{n}_S - \sum_{j\in I}\alpha_j)$$

subject to:

$$\sum_{j\in I}\alpha_j y_{jc} = \frac{1}{|C|}\sum_{j\in I}\alpha_j, \text{ for each } c \in C,$$

$$0 \leq \alpha_j \leq 1 \text{ for all } j \in I.$$

Note that once we have relaxed the integrality constraints we can no longer expect $|S|$ to be an integer multiple of $|C|$.

3.3. The multi-class SVM with kernel densities

The final step is to drop the parameter $\bar{n}_S$ leaving θ as the user parameter to control the solution. We call this form of the support set optimization *The Multi-Class SVM with Kernel Densities*.

$$\text{maximize}_\alpha \sum_{i\in I}\sum_{j\in I}\alpha_i\alpha_j(1 - 2y_i \cdot y_j)k_{ij} + \theta\sum_{j\in I}\alpha_j$$

subject to:

$$\sum_{j\in I}\alpha_j\big(y_{jc} - \frac{1}{|C|}\big) = 0, \text{ for each } c \in C,$$

$$0 \leq \alpha_j \leq 1 \text{ for all } j \in I.$$

This problem can be solved using gradient descent-based methods. Let $A = [a_{ij}]$ where $a_{ij} = (1 - 2y_i \cdot y_j)k_{ij}$ and let $\pi = [\pi_c]_{c\in C}$ denote a vector of Lagrange multipliers for the balance constraints. Form the Lagrangian function $\mathcal{L}(\alpha, \pi; \theta)$ as

$$\mathcal{L}(\alpha, \pi; \theta) = \alpha^T A\alpha + \theta\mathbf{e}^T\alpha + \pi^T\big(\bar{Y} - \frac{1}{|C|}\big)^T\alpha,$$

where $\mathbf{e}$ is a vector of 1's of length $|I|$ and $\bar{Y}$ is the $|I| \times |C|$ matrix formed from the one-hot labels y_i taken as row vectors. Note that A is a symmetric matrix. Setting the gradient $\nabla\mathcal{L}(\alpha, \pi; \theta) = 0$, we seek a solution to the system of equations:

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial\alpha_i}(\alpha, \pi; \theta) &= 2A_{i\cdot}\alpha + \theta + \big(y_i - \frac{1}{|C|}\big)\pi = 0, \\
\frac{\partial\mathcal{L}}{\partial\pi_c}(\alpha, \pi; \theta) &= \big(\bar{Y}_{\cdot c} - \frac{1}{|C|}\big)^T\alpha = 0,
\end{aligned}$$

for all $i \in I$ and $c \in C$. Boundary conditions can be handled easily. Denote the solution (α^*, π^*).

For any input, x, we use the following as classification probabilities: for each $c \in C$,

$$P_{\alpha^*}\{Y = c^1 | X \in (x)_{dx}\} = \frac{\sum_{j\in I}\alpha_j^* y_{jc}k(x, x_j)}{\sum_{j\in I}\alpha_j^* k(x, x_j)}.$$

The fact that $\sum_{c\in C} y_{jc} = 1$ implies that these probabilities sum to 1 even though we have relaxed the integer constraints on the α_j's. Also note that the normalizing constant for the kernel density function need not be computed explicitly as it would cancel out in this expression. The predictor (or decision rule) for the Multi-Class SVM with Kernel Densities is to set:

$$\hat{y} = \text{argmax}^1_{c\in C} P_{\alpha^*}\{Y = c^1 | X \in (x)_{dx}\}.$$

This completes our description of the multi-class problem and solution technique. In practice, the user may choose to perform a grid search on θ and choose the value which optimizes the tradeoff between accuracy and overfitting of the predictor on the training set. We demonstrate this in the application section below.

4. The two-class SVM with kernel densities

A special case of the Multi-Class SVM with Kernel Densities is the binary classification problem, i.e. $|C| = 2$. In this section we show that it is equivalent to the classical binary SVM, restricted to kernel density functions (compare, for example, with (7.32-7.34) in [1] or (4.11-4.13) in [2]). This is accomplished with two changes of variable.

Proposition 1. *After the change of variable $\alpha'_j = \kappa\alpha_j$ and $y'_j = y_{j1} - y_{j2}$, for all $j \in I$, the Two Class SVM with Kernel Densities is equivalent to the SVM dual given by:*

$$\text{maximize}_{\alpha'} Z'(\alpha') = \sum_{j\in I}\alpha'_j - \frac{1}{2}\sum_{i\in I}\sum_{j\in I}\alpha'_i\alpha'_j y'_i y'_j k_{ij}$$

subject to:

$$\sum_{j\in I}\alpha'_j y'_j = 0,$$

$$0 \le \alpha'_j \le \kappa \text{ for all } j \in I,$$

where $\kappa = \frac{2}{\theta}$.

Proof. After effecting the first change of variable ($\alpha'_j = \kappa\alpha_j$, where $\kappa = \frac{2}{\theta}$,) the Two Class SVM is:

$$\text{maximize}_{\alpha}\frac{1}{\kappa^2}\sum_{i\in I}\sum_{j\in I}\alpha'_i\alpha'_j\sum_{c\in C}(1-2y_{ic})y_{jc}k_{ij} + \frac{2}{\kappa^2}\sum_{j\in I}\alpha'_j$$

subject to:

$$\sum_{j\in I}\alpha'_j y_{jc} = \frac{1}{|C|}\sum_{j\in I}\alpha'_j, \text{ for each } c \in C$$

$$0 \le \alpha'_j \le \kappa \text{ for all } j \in I.$$

Clearly, the factor $\frac{1}{\kappa^2}$ in the objective has no effect on the solution. It will be removed as part of the next step. The parameter κ now appears as an upper bound on the selector variables but still plays the role of controlling the size of the support set in place of θ.

The next change of variable relates to the coding of the observation labels. By setting the scalar variable $y'_i = y_{i1} - y_{i2}$ we have that $y'_i = 1$ if the observation is in class 1 and $y'_i = -1$ if it is in class 2. This is the classical coding scheme for the SVM. Taking advantage of the fact that $y_{i1} + y_{i2} = 1$, the expression $y'_i y'_j$ is given by:

$$\begin{aligned}
y'_i y'_j &= (y_{i1} - y_{i2})(y_{j1} - y_{j2}) \\
&= y_{i1}y_{j1} + y_{i2}y_{j2} - y_{i1}y_{j2} - y_{i2}y_{j1} \\
&= y_{i1}y_{j1} + y_{i2}y_{j2} - y_{i1}(1 - y_{j1}) - y_{i2}(1 - y_{j2}) \\
&= 2y_{i1}y_{j1} + 2y_{i2}y_{j2} - (y_{i1} + y_{i2}) \\
&= 2(y_{i1}y_{j1} + y_{i2}y_{j2}) - 1 \\
&= 2y_i \cdot y_j - 1
\end{aligned}$$

which matches the negative of (1).

After scaling the objective by $\kappa^2/2$, the Two-Class SVM objective becomes:

$$\begin{aligned}
Z &= \frac{1}{2}\sum_{i\in I}\sum_{j\in I}\alpha'_i\alpha'_j\sum_{c\in C}(1-2y_{ic})y_{jc}k_{ij} + \sum_{j\in I}\alpha'_j \\
&= \frac{1}{2}\sum_{i\in I}\sum_{j\in I}\alpha'_i\alpha'_j\left(\sum_{c\in C}(1-2y_{ic})y_{jc}\right)k_{ij} + \sum_{j\in I}\alpha'_j \\
&= \frac{1}{2}\sum_{i\in I}\sum_{j\in I}\alpha'_i\alpha'_j\left(\sum_{c\in C}y_{jc} - 2\sum_{c\in C}y_{ic}y_{jc}\right)k_{ij} + \sum_{j\in I}\alpha'_j \\
&= \frac{1}{2}\sum_{i\in I}\sum_{j\in I}\alpha'_i\alpha'_j\left(1 - 2\sum_{c\in C}y_{ic}y_{jc}\right)k_{ij} + \sum_{j\in I}\alpha'_j \\
&= \frac{1}{2}\sum_{i\in I}\sum_{j\in I}\alpha'_i\alpha'_j\left(-y'_i y'_j\right)k_{ij} + \sum_{j\in I}\alpha'_j \\
&= \sum_{j\in I}\alpha'_j - \frac{1}{2}\sum_{i\in I}\sum_{j\in I}\alpha'_i\alpha'_j y'_i y'_j k_{ij}.
\end{aligned}$$

Applying the change of variable to the constraint, for $c = 1, 2$:

$$\sum_{j\in I}\alpha'_j y_{jc} = \frac{1}{2}\sum_{j\in I}\alpha'_j$$

Equivalently,

$$\sum_{j\in I}\alpha'_j y_{j1} = \sum_{j\in I}\alpha'_j y_{j2}$$

and, hence,

$$\sum_{j\in I}\alpha'_j\left(y_{j1} - y_{j2}\right) = 0.$$

But, $y'_j = y_{j1} - y_{j2}$. So after the change of variable, the constraint is:

$$\sum_{j\in I}\alpha'_j y'_j = 0.$$

The result follows. □

We have achieved the objective of explaining the dual program of the support vector machine technique in terms of Bayes' classifiers, at least for kernel functions which can be normalized as kernel density functions, such as the radial basis function. This approach avoids the complexities of the traditional exposition such as the need to introduce Karush-Kuhn-Tucker conditions and the kernel trick. It also leads immediately to a multi-class classification formulation with a probability-based decision rule and easily computed probabilities. We close the paper with an example suitable for classroom use.

5. Example: the iris classification problem

For illustration, we use the iris classification problem since that data set is readily available in both R and Python. The iris data set has 150 observations of flowers in the iris genus. Each flower is characterized by four measurements: lengths and widths for both petal and sepal. Each flower is classified into one of three species (setosa, versicolor, and virginica), with 50 observations in each species. We take a random training sample of 40 observations of each species. The classification challenge is to correctly classify an iris flower by species using only the petal and sepal measurement data.

For visualization purposes, we project the measurement data, after normalization, down to a two-dimensional space using the first two principal components, called "PC1" and "PC2" in the figures. Fig. 1 illustrates the data set before and after projection.

A radial basis kernel function (using bandwidth 1) is used to generate empirical probability density functions. Fig. 2 illustrates the conditional density functions by species. These correspond to the quantities $P_I\{X \in (x)_{dx} | Y = c^1\}$ for each species, c. A similar figure, not shown, can be displayed for the probability density function, $P_I\{X \in (x)_{dx}\}$ derived using the same kernel function.

By construction of the training set, the marginal probability mass function for species is given by $P_I\{Y = c^1\} = \frac{1}{3}$. The conditional probability, $P_I\{Y = c^1 | X \in (x)_{dx}\}$ for each species c is shown in Fig. 3. These three surfaces sum to 1 at every point in the domain, but it is clear that over many regions of the domain, a single species dominates.

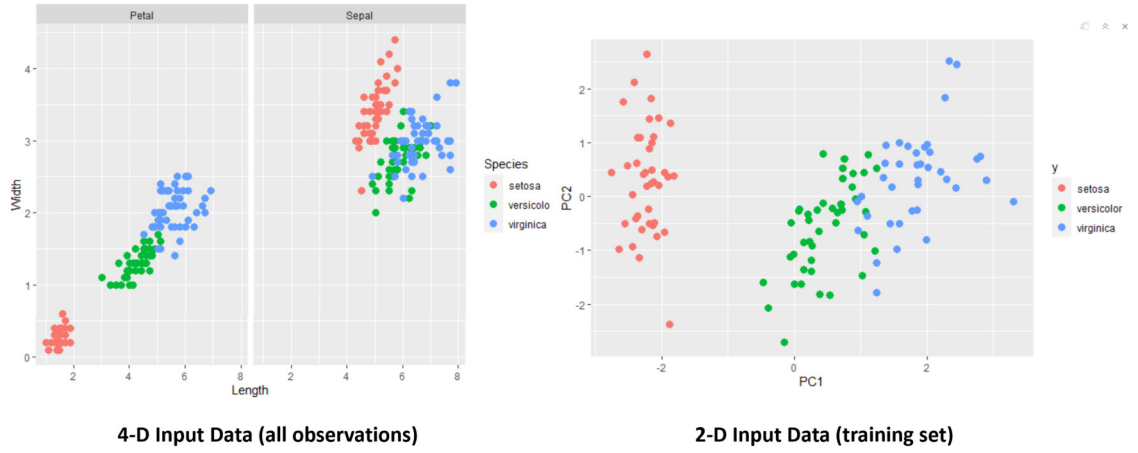

Fig. 1. Iris Classification Projection to Two Dimensions. The first two principal components are labeled PC1 and PC2. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

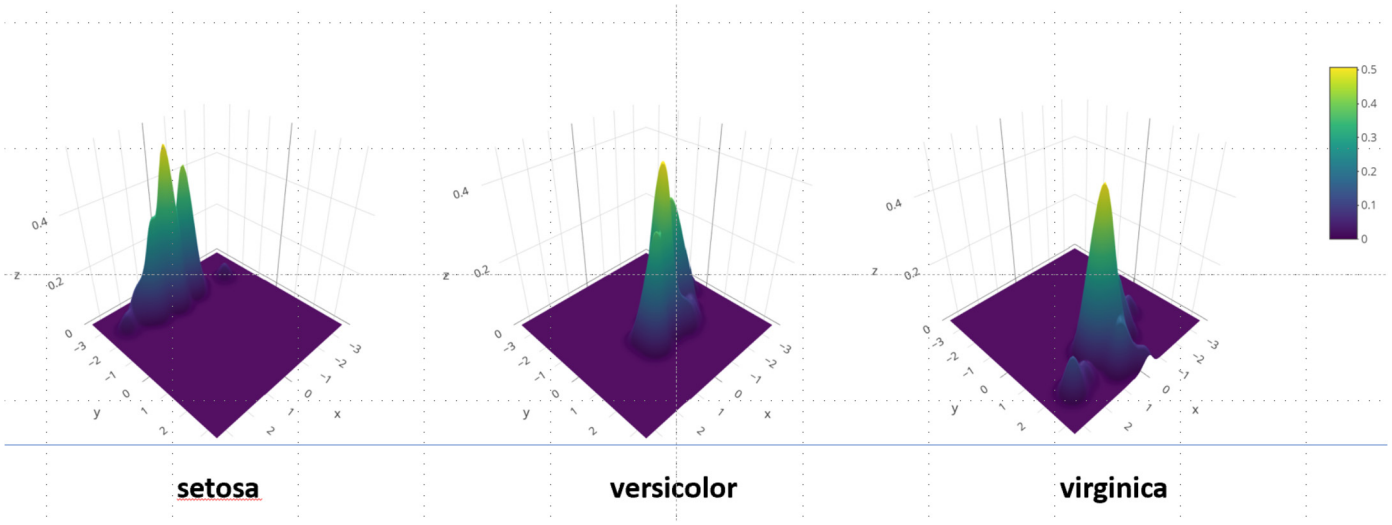

Fig. 2. Gaussian Kernel Densities of Inputs by Species. These represent the quantities $P_I\{X \in (x)_{dx} | Y = c^1\}$ for each species c.

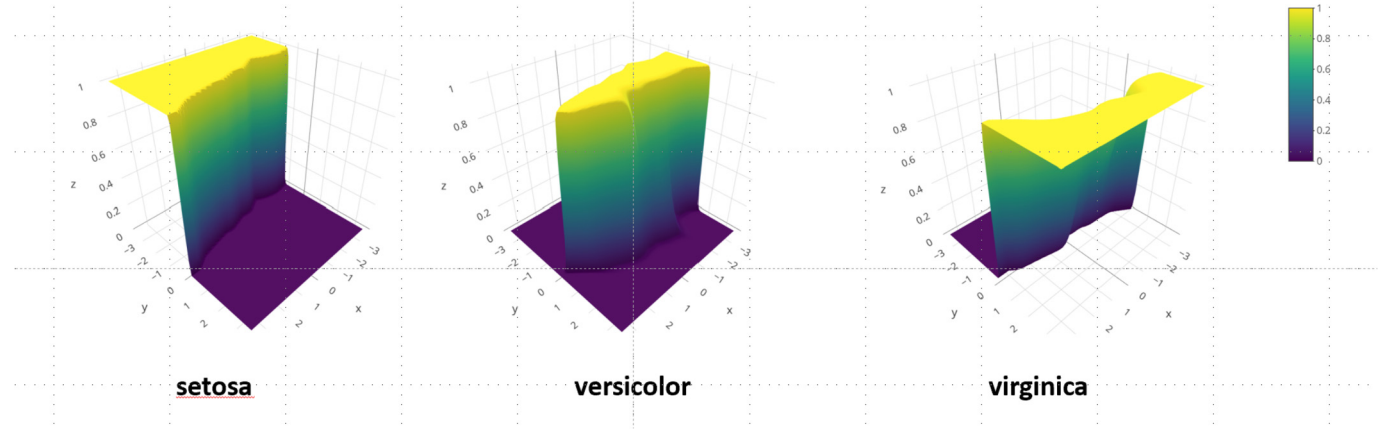

Fig. 3. Conditional Probability Density Functions of Label Given Input. These represent the quantities $P_I\{Y = c | X \in (x)_{dx}\}$ for species c.

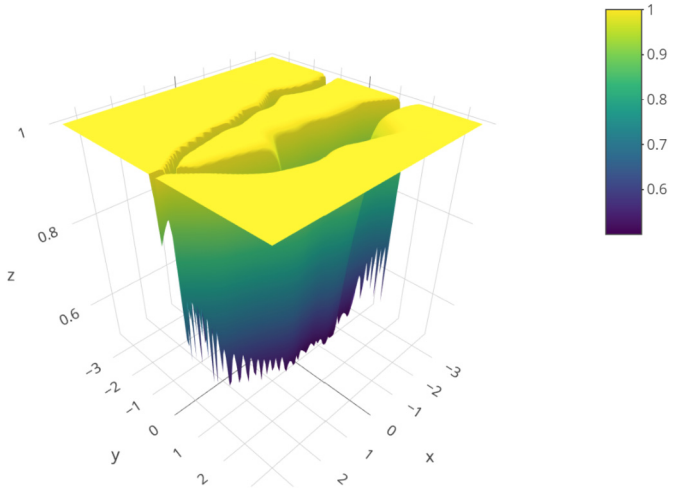

Fig. 4. Maximum of Classification Probability Density Functions. This represents the quantity $\max_{c\in C} P_I\{Y = c^1 | X \in (x)_{dx}\}$.

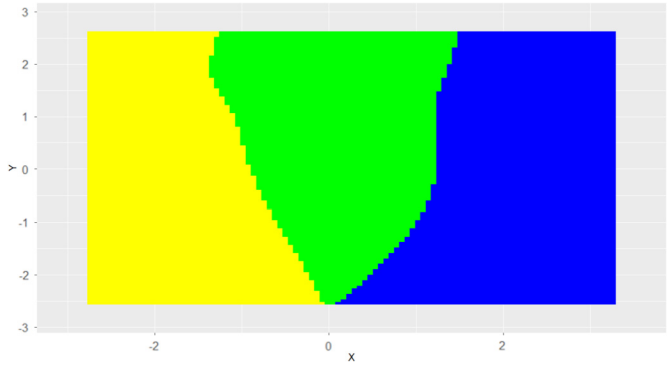

Fig. 5. Bayes' Classifier Rule. This represents the quantity $C_I(x) = \text{argmax}_{c\in C} P_I\{Y = c^1 | X \in (x)_{dx}\}$. (The axis orientation differs from the surface plots.)

Though the conditional probabilities sum to 1, the quantity of interest is the maximum conditional probability at each point, $\max_{c\in C} P_I\{Y = c^1 | X \in (x)_{dx}\}$. This surface is shown in Fig. 4. The crevasses of this surface point to regions of uncertainty where more than one species has a substantial probability of existing.

The Naïve Bayes' Classifier decision rule is given by $C_I(x) = argmax^1_{c\in C} P_I\{Y = c^1 | X \in (x)_{dx}\}$. Fig. 5 illustrates. The orientation of axes differ between the 2D and 3D plots.

In Fig. 6, we superimpose the entire training set on the decision rule surface. Some misclassifications are evident even when using the full training set as support. The accuracy score is computed as $\sum_{i\in I} 1_{\{C_I(x_i)=y_i\}}/|I|$. The accuracy of the Bayes' Classifier defined on I is 93% for this training set.

To tune the value of θ, we perform a 10-fold cross-validation, dividing up the training data according to 10 different partitions. For each value of θ, we find the α^* which optimizes the Multi-Class SVM on the training portion of the partition (80 observations) and then use that vector of α^* to predict the classifications in the testing portion of the partition (40 observations). The resulting accuracies, both training and testing, are then averaged over the 10 partitions. Two interesting values of θ emerge from a grid search: one which maximizes accuracy on the training data folds ($\theta = 0.3$) and the other which maximizes accuracy on the validation data folds ($\theta = 3.0$). Table 1 shows the tuning results for these two parameter values.

To illustrate the problem of overfitting, we solve the Multi-Class SVM on the unpartitioned training set (120 observations) with $\theta = 0.3$ and display the solution in Fig. 7. There are only

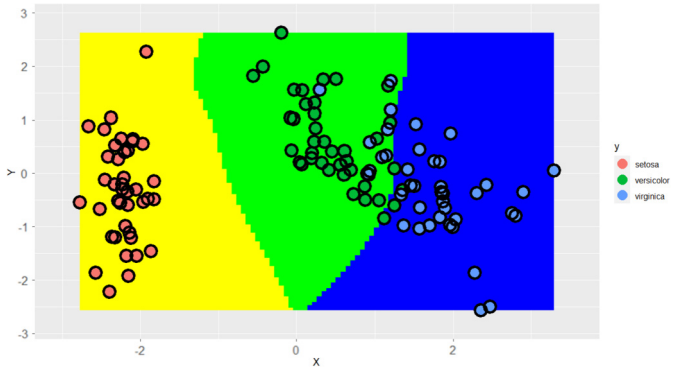

Fig. 6. Bayes' Classifier with Training Set. The accuracy score is 0.93.

Table 1
Parameter Tuning on 10-fold Cross-Validation.

Support bonus, θ	Average support count	Average accuracy on training data (%)	Average accuracy on validation data (%)
0.3	42.9	95.83	85.00
3.0	60.4	93.24	91.67

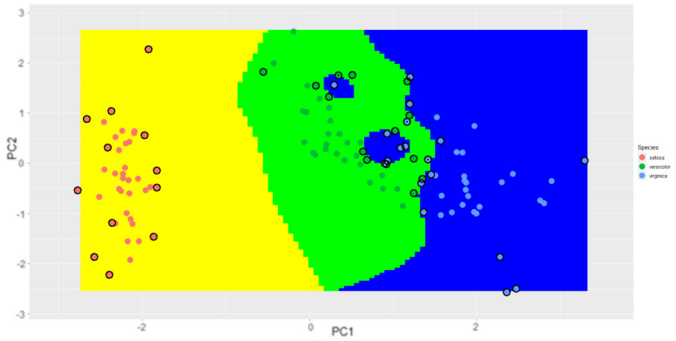

Fig. 7. Highest Training Accuracy. For $\theta = 0.3$, chosen to maximize accuracy on training data, the support count is 44 and the accuracy is 98%.

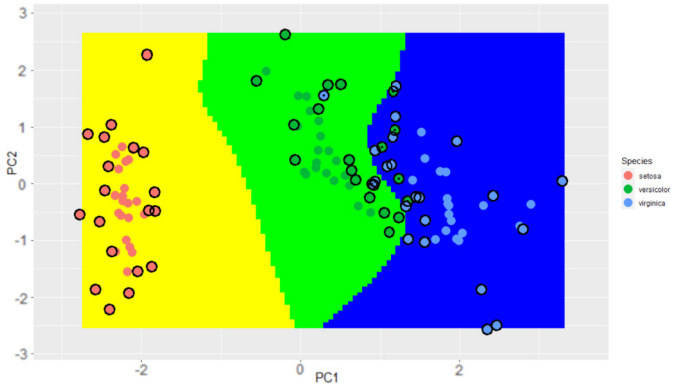

Fig. 8. Highest Validation Accuracy. For $\theta = 3.0$, chosen to maximize accuracy on validation data, the support count is 61 and the accuracy is 93%.

three misclassifications (shown with black dots at their centers) leading to an accuracy of 97.5%. Only 44 observations belong in the support set. The role of support vectors in shaping the decision boundaries is clearly apparent in this solution, with vectors from alternate species lining up along the contentious boundaries in close proximity to one another. However, when this solution is applied to the testing data (30 observations), it scores only 76.6% accuracy.

In contrast, we display the full training solution for $\theta = 3.0$ in Fig. 8. Its accuracy on the training data is 93.3% and it uses 61 support vectors. However, when applied to the testing data it scores 90.0% accuracy which is much superior to the $\theta = 0.3$ solution. This demonstrates that parameter tuning for the Multi-Class SVM should be conducted with cross-validation to prevent overfitting.

6. Further research

We mentioned that the probabilistic decision rule could be used to formulate a wide range of optimization problems. We suggest starting with simply choosing a support set S to maximize expected accuracy ($\sum_{i\in I} P_S\{\tilde{C}(x_i) = y_i\}$) on the set I, subject to size and balance constraints on the support set. One could also consider weighted accuracy metrics to value the correct classification of some classes over others. Also, we have not derived the metric $R(S)$ from first principles: it has been developed solely as an interpretation of the SVM dual objective. A more fundamental justification of this metric would be helpful. It would also be interesting to see if the dual of the Multi-Class SVM has a geometric interpretation when $|C| > 2$.

Data availability

The iris data set is available in R and Python.

Acknowledgement

This research is supported in part by grant RS-CAASI-00009 from the Civil Aviation Authority of Singapore (CAAS).

References

[1] Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[2] A. Nefedov, Support vector machines: a simple tutorial, https://sustech-cs-courses.github.io/IDA/materials/Classification/SVM_tutorial.pdf, 2016.