Understanding Support Vector Machines (SVMs)

Jacob Casey *

August 2023

Abstract

The classification of multiple, linearly separable classes through Support Vector Machines (SVMs) poses a unique challenge that is addressed through the One-Against-All (OAA) approach. This paper covers design the construction of a OAA SVM multi-class classifier, on linearly seperable data with 3 distinct classes. It demystifies the mathematics and voting strategy behind a multi-class SVM that picks the hyperplane decision boundary that gives the highest confidence score for accurate classification. The paper then elucidiates how to construct a hyperplane using lagrange multipliers, showing the underpinnings of SVM classifiers. Fianlly, the paper illustrates the process of classifying three new test samples, providing an insight into OAA SVM multi-class classifiers. The work and methodologies shared in this paper serve as a valuable introductory resource for understanding how they work and how SVM are designed within the field of pattern recognition.

1 Linearly Seperable, Multi-Class Data

Support Vector Machines [1] are a supervised machine learning technique that can be used to classify data points into k different classes [2] that are not labelled. It finds a hyperplane that maximizes the margin between the closest points of different classes, projecting the feature space into higher dimensions so that the data can be seperated by a linear discriminant. The margin is defined as the geometric distance between the decision boundary (hyperplane) and the nearest data points from each class, or the support

*M.Sc. Artificial Intelligence, King's College London, Department of Informatics.
For supporting source-code please view the supplementary GitHub repository at https://github.com/jacobmcasey/MultiClass-SVM-Lagrange-Hyperplane-Construction-Paper.

vectors. The SVM algorithm finds the hyperplane that has the largest margin, which is expected to generalise well to new data and seeks to achieve better generalisation performance, i.e., the ability to accurately classify new, unseen data.

The dataset $\mathcal{D}$ in Table 1 is comprised of three classes (Class 1, Class 2 and Class 3). Each data point $\mathbf{x_i}$ is a feature vector $\mathbf{x}_i = (x_{i,1}, x_{i,2}) \in \mathbb{R}^2$ with corresponding class label $y_i \in$ (Class 1, Class 2, Class 3).

Class 1	Class 2	Class 3
(1.7974, 1.0628)	(4.8911, 2.0221)	(2.9769, 2.9397)
(2.3158, 1.1534)	(5.2931, 1.9548)	(3.1476, 3.0342)
(1.9073, 0.9068)	(4.9973, 1.7884)	(3.0662, 3.1951)
(2.0483, 0.6173)	(5.0417, 1.6080)	(3.2062, 3.1862)
(1.8183, 0.7175)	(5.1645, 1.7558)	(3.0687, 2.6473)
(1.6550, 0.8875)	(4.7698, 2.0751)	(2.8321, 2.9381)
(1.9531, 0.9531)	(4.8798, 1.9416)	(3.0648, 2.9229)
(1.9061, 1.1085)	(4.7343, 2.0393)	(2.9078, 3.2114)
(2.0993, 0.9723)	(5.0135, 1.7150)	(2.8646, 3.1223)
(2.1295, 1.3046)	(4.8796, 2.3704)	(2.7042, 2.8560)

Table 1: One-hot encoded class labels and (x, y) coordinates.

2 Linearly Seperable Data

A dataset is said to be linearly separable if there exists a hyperplane that can perfectly separate the data points of different classes. In other words, there exists a hyperplane that separates the points of one class from those of the other two classes. This notion of linear separability has been fundamental in the development of support vector machines and other classification algorithms, with seminal work done by Vapnik and Chervonenkis [3]

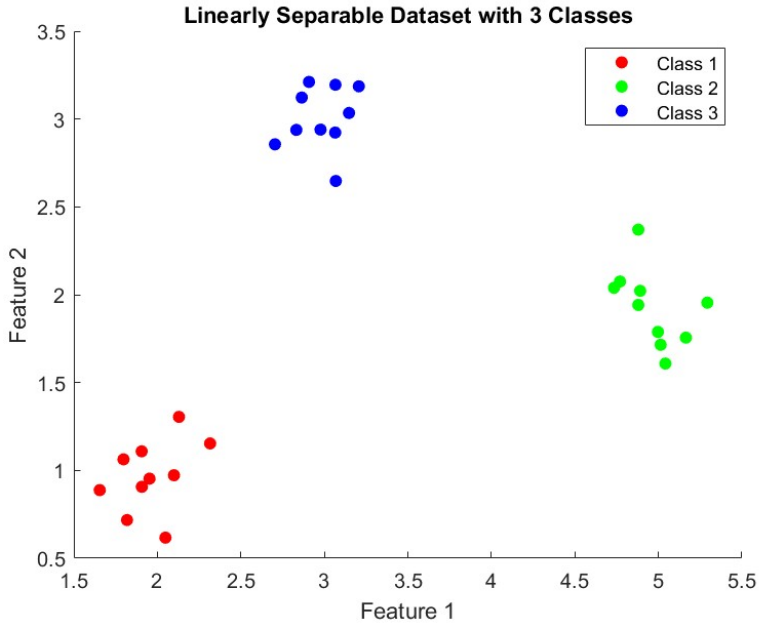

Figure 1: The dataset $\mathcal{D}$ plotted in its 2 dimensional feature space.

In the case of three classes, you can use an OAA approach, where you train three binary SVM classifiers, one for each class versus the other two combined. Each binary classifier finds a hyperplane that separates one class from the other two. Then, to classify a new data point, you apply each binary classifier to it and choose the class that has the highest output value, or majority voting, depending on the design. The output value is the sign of the distance from the hyperplane to the data point, which therefore can be positive or negative.

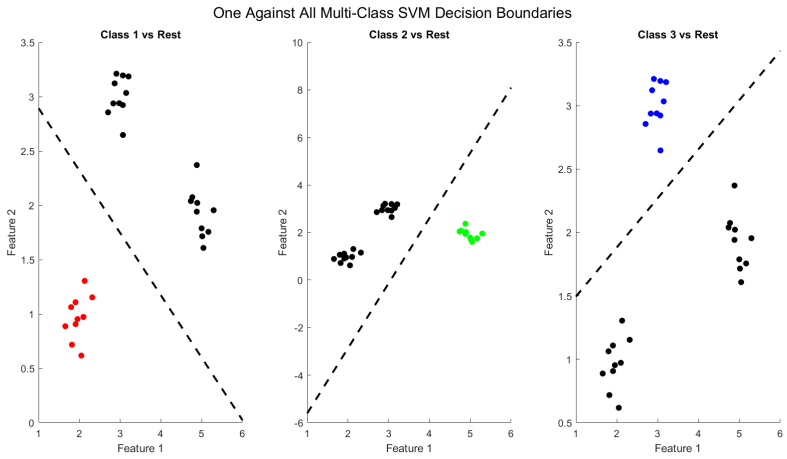

Figure 2: Image showing the OAA approach with hyperplanes for each round for each of the three classes in $\mathcal{D}$. Sidenote: the decision boundaries are infact the optimal with largest for OAA in this feature space. This was done in MATLAB using fitcsvm.

3 Design of a SVM

This design therefore, involves using three linear SVMs to predict one of three class labels based on a 2D input feature vector defined earlier. The predicted class label is assigned based on the classifier that provides the highest confidence score, and each binary SVM is trained to separate a specific class from the other two. We then define the number of inputs, outputs, SVMs, and the type of class label assignement.

Number of inputs: The input feature vector has 2 dimensions (x_1, x_2).
Number of outputs: The output is the predicted class label (Class 1, Class 2, or Class 3).
Number of SVMs used: We use 3 linear SVMs, one for each class in the dataset.
Class label assignment: The predicted class label is assigned based on the classifier that provides the highest confidence score.

The multi-class classifier works as follows. For each class, train a binary linear SVM classifier to separate that class from the other two classes.

Linear SVM 1: Class 1 vs (Class 2 and Class 3)

Linear SVM 2: Class 2 vs (Class 1 and Class 3)
Linear SVM 3: Class 3 vs (Class 1 and Class 2)

For a given input feature vector (x_1, x_2), pass it through all 3 SVM classifiers and obtain the decision function values using

$$f_i(\mathbf{x}) = \mathbf{w_i}^T \mathbf{x} + b_i, \tag{1}$$

where $\mathbf{w_i}$ is the weight vector and b_i is the bias term for the i^{th} classifier. Then, simply determine the classifier that gives the highest decision function value (confidence score) and assign the class label corresponding to the winning classifier as the predicted class label for the input feature vector.

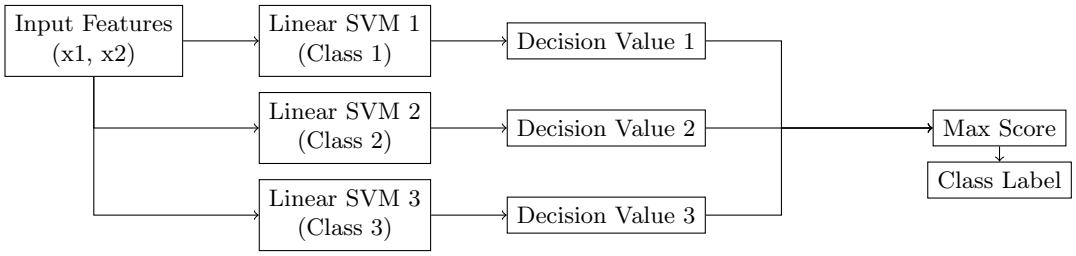

Figure 3: Block diagram of the multi-class classifier using the one-against-all strategy with linear SVMs.

4 But How Do we Find w_i and b_i?

For example, we pick the two support vectors for Class 2 vs Rest (Class 1, Class 3) by inspection, which are the two closest datapoints between the two classes. For illustration I show the process for Class 2 vs Rest as this allows for the most obvious hyperplane construction based on the margin via inspection on the dataset above, with the minimal amount of support vectors.

$$\mathbf{v}_1 = \begin{pmatrix} 3.0687, & 2.6474 \end{pmatrix} \quad \text{(Rest)} \tag{2}$$

$$\mathbf{v}_2 = \begin{pmatrix} 4.7344, & 2.0394 \end{pmatrix} \quad \text{(Class 2)} \tag{3}$$

$$y_1 \text{ for } \mathbf{v}_1 \text{ is } 1 \tag{4}$$

$$y_2 \text{ for } \mathbf{v}_2 \text{ is } -1 \tag{5}$$

Next, we need to find the values of the Lagrange multipliers λ_i that correspond to the support vectors. From the Karush–Kuhn–Tucker [4] (KKT) conditions, we know that $\lambda_i > 0$ for support vectors and $\lambda_i = 0$ for non-support vectors. Therefore, we can write:

$$\mathbf{w} = \lambda_1 \mathbf{v}_1 - \lambda_2 \mathbf{v}_2 \tag{6}$$

$$\mathbf{w} = \lambda_1 \begin{pmatrix} 3.0687, & 2.6474 \end{pmatrix} - \lambda_2 \begin{pmatrix} 4.7344, & 2.0394 \end{pmatrix} \tag{7}$$

Next, we know that $y_i(\mathbf{w}^T \mathbf{v}_i + w_0) = 1$ when $\mathbf{v}_i$ is a support vector. Then the two equations with the corresponding y_i values for the two support vectors we have are:

$$(\lambda_1 \mathbf{v}_1 - \lambda_2 \mathbf{v}_2)^T \mathbf{v}_1 + w_0 = 1 \tag{8}$$

$$(\lambda_1 \begin{pmatrix} 3.0687, & 2.6474 \end{pmatrix} - \lambda_2 \begin{pmatrix} 4.7344, & 2.0394 \end{pmatrix})^T \begin{pmatrix} 3.0687, & 2.6474 \end{pmatrix} + w_0 = 1 \tag{9}$$

$$-1(\lambda_1 \mathbf{v}_1 - \lambda_2 \mathbf{v}_2)^T \mathbf{v}_2 + w_0 = 1 \tag{10}$$

$$-1(\lambda_1 \begin{pmatrix} 3.0687, & 2.6474 \end{pmatrix} - \lambda_2 \begin{pmatrix} 4.7344, & 2.0394 \end{pmatrix})^T \begin{pmatrix} 4.7344, & 2.0394 \end{pmatrix} + w_0 = 1 \tag{11}$$

Expanding Equations 9 and 11, we get the following linear equations:

$$16.4256\lambda_1 - 9.0518\lambda_2 + w_0 = 1 \tag{12}$$

$$-19.9276\lambda_1 + 26.5737\lambda_2 + w_0 = 1 \tag{13}$$

Simplify and solve for λ and w_0 by subsituting, will give us the weights for the hyperplane:

$$w_1 = 1.06 \tag{14}$$

$$w_2 = -0.39 \tag{15}$$

$$w_0 = -3.23 \tag{16}$$

Finally, the hyperplane equation is:

$$1.06x_1 - 0.39x_2 - 3.23 = 0 \tag{17}$$

We can use this hyperplane to classify new data points into class 2 versus classes 1 and 3 as outlined in the previous chapter. The decision boundary is given by:

$$\mathbf{w}^\top \mathbf{x} + b = 0 \tag{18}$$

where $\mathbf{x}$ is the input feature vector. If $\mathbf{w}^\top \mathbf{x} + b > 0$, then $\mathbf{x}$ is classified as class 1 or 3, and if $\mathbf{w}^\top \mathbf{x} + b < 0$, then $\mathbf{x}$ is classified as class 2.
Finally, this is repeated for the other two OAA rounds.

5 Classification

Table 2: SVM Classification Results

Test Sample	**SVM 1**	**SVM 2**	**SVM 3**	**Classification**
$(2.6984, 1.1452)$	0.3931	-0.1743	-0.0591	Class 1
$(3.9188, 1.5804)$	0.3882	0.0049	-0.2231	Class 1
$(3.0470, 3.0369)$	-0.0032	-0.5855	0.9566	Class 3

Using the SVM models and test sample $(2.6984, 1.1452)$, we can compute the distances and confidence scores as follows:

For Class 1 vs Rest:
Hyperplane equation: $-0.61x_1 - 1.06x_2 + 3.69 = 0$
Distance from sample to hyperplane: 2.0049
Confidence score: 0.4072

For Class 2 vs Rest:
Hyperplane equation: $1.06x_1 - 0.39x_2 - 3.23 = 0$
Distance from sample to hyperplane: 3.2562
Confidence score: 0.2072

For Class 3 vs Rest:
Hyperplane equation: $-0.62x_1 + 1.59x_2 - 1.76 = 0$
Distance from sample to hyperplane: 1.1578
Confidence score: 0.3856

Therefore, the predicted class for the test sample is Class 1, since it has the highest confidence score of 0.4072. This is repeated for all other SVMs and classes to get the most accurate classifications as shown in Table 2.

References

[1] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[2] J. Weston and C. Watkins, "Multi-Class support vector machine," *Proc. Europ. Symp. Artificial Neural Networks*, Mar. 1999.

[3] V. Vapnik and A. Chervonenkis, "A note on one class of perceptrons," in *Automation and Remote Control*, vol. 25, 1963, pp. 821–837.

[4] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press, Jan. 1951, vol. 2, pp. 481–493.